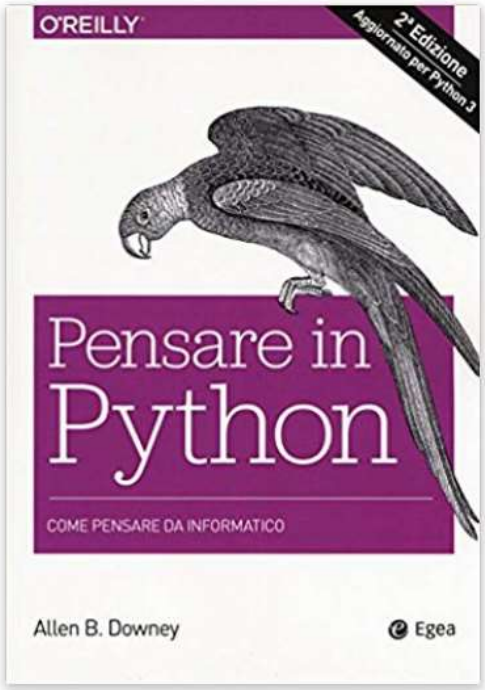
Fondamenti di Informatica

AA 2019/2020

Eng. Ph.D. Michela Paolucci

DISIT Lab http://www.disit.dinfo.unifi.it
Department of Information Engineering, DINFO
University of Florence
Via S. Marta 3, 50139, Firenze, Italy
tel: +39-055-2758515, fax: +39-055-2758570
michela.paolucci@unifi.it

Libro di Testo



- **Titolo del Libro: Pensare in Python**
- **Autore : Allen Downey**
- **Editore: EGEA**
- **Data di Pubblicazione: 2018**
- **Genere: libro. elaborazione dati**
- **Argomento : Python, linguaggio**
- **ISBN-10: 8823822645**
- **ISBN-13: 9788823822641**

Bibliografia –

- Concetti di informatica e fondamenti di Python, Cay Horstmann, Rance D. Necaise, Apogeo
- Pensare in Python - Come pensare da Informatico, Allen Downey, Green Tea Press
- Pensare da informatico - Imparare con Python, Allen Downey Jeffrey Elkner Chris Meyers
- …

Programma del Corso

Anno Accademico 2019-20

Laurea Triennale (DM 270/04) - INGEGNERIA GESTIONALE

Programma del corso - Cognomi A-H

- Introduzione al linguaggio python
  - o Tipi, variabili e costanti
  - o Operatori ed espressioni
  - o Istruzioni
- Rappresentazione dei dati
  - o Numeri
  - o Interi
  - o Caratteri e stringhe

Elementi di sintassi di un linguaggio

Esecuzione di programmi e ambienti: notebooks, IDE, console

- Il linguaggio python
  - o tipi mutabili e immutabili
  - o operatori ed espressioni
  - o istruzioni
  - o funzioni
  - o cicli while e for
  - o esecuzione condizionale
- Strutture dati e algoritmi elementari: Liste, Dizionari, Insiemi, iterazioni su strutture dati

Costo di esecuzione e complessità

Il modello di costo

Cenni sulla complessità di un algoritmo:

- Algoritmi di ordinamento su vettori
  - o Sequential-sort
- Cenni sugli alberi
  - o Alberi
  - o Alberi binari di ricerca: i) Visita in forma ricorsiva; ii) Ricerca; iii) Inserimento ordinato

Cenni su analisi dei dati, lettura e scrittura di file in forma tabulare, grafici.

Pagina del Corso

http://www.disit.org/drupal/?q=node/7020

Qui trovate:

- AVVISI
- Slide del corso
- Approfondimenti

Distributed Systems and Internet Technologies Lab
Distributed Data Intelligence and Technologies Lab
Department of Information Engineering (DINFO)
University of Florence

http://www.disit.dinfo.unifi.it

HOME ABOUT RESEARCH INNOVATION EDUCATION AND COURSES HOWTO EVENTS

CORSO DI FONDAMENTI DI INFORMATICA, TRIENNALE, GESTIONALE E MECCANICA A-L, AA 2018/2019

AVVISI

ATTENZIONE: l'esame orale relativo all'appello del 23 Gennaio si terrà in data 30 Gennaio:
- aula 007, viale morgagni ore 9:00

LIBRO DI TESTO AA 2019/2020

- Allen Downey. Pensare in Python. EGEA

ORARIO DEL CORSO AA 2019/2020

L'orario è consultabile a questo link

SLIDE DEL CORSO AA 2019/2020

Si ricorda che le slide del corso NON sono in alcun modo sostitutive del libro di testo.

Modalità d’esame – alcune linee guida -

L'esame si compone di una prova scritta e una orale.

La prova scritta è svolta su carta A4.

Si accede alla prova orale solo se la parte di programmazione è corretta e funzionante

La prova orale può essere sostenuta a partire dalla settimana seguente alla prova scritta, non oltre la prova scritta successiva.

La prova orale inizia con la discussione dell'elaborato, e prosegue con l'approfondimento di tutti i contenuti del programma del corso.

Orario del Corso e Ricevimento

Docente: PAOLUCCI MICHELA

Ingegneria FIRENZE - A.A. 2019/2020 - 2° periodo

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
08:15						
09:15						
10:15						
11:15	FonDiInf(A-L) (Auditorium B - C.D.M.) FonDiInf(A-L) (001 - C.D.M.)			FonDiInf(A-L) (Auditorium A - C.D.M.) FonDiInf(A-L) (001 - C.D.M.)		
12:15	FonDiInf(A-L) (Auditorium B - C.D.M.) FonDiInf(A-L) (001 - C.D.M.)			FonDiInf(A-L) (Auditorium A - C.D.M.) FonDiInf(A-L) (001 - C.D.M.)		
14:00				FonDiInf(A-L) (Auditorium A - C.D.M.) FonDiInf(A-L) (001 - C.D.M.)		
15:00						
16:00						
17:00						
18:00						

Legenda: C.D.M.: Centro Didattico Morgagni

- Il ricevimento si svolge su appuntamento contattando la docente via e-mail:
  - michela.paolucci@unifi.it

Python: Compilatori e Console

Esecuzione di programmi e ambienti: notebooks, IDE, console
- Il linguaggio python
  - tipi mutabili e immutabili
  - operatori ed espressioni
  - istruzioni
  - funzioni
  - cicli while e for
  - esecuzione condizionale
- Strutture dati e algoritmi elementari: Liste, Dizionari, Insiemi, iterazioni su strutture dati

Costo di esecuzione e complessità
Il modello di costo
Cenni sulla complessità di un algoritmo:
- Algoritmi di ordinamento su vettori
  - Sequential-sort
- Cenni sugli alberi
  - Alberi
  - Alberi binari di ricerca: i) Visita in forma ricorsiva; ii) Ricerca; iii) Inserimento ordinato

Cenni su analisi dei dati, lettura e scrittura di file in forma tabulare, grafici.

Liste

NOTA: manca integrazione con ‘concetti di informatica e fondamenti di Python’

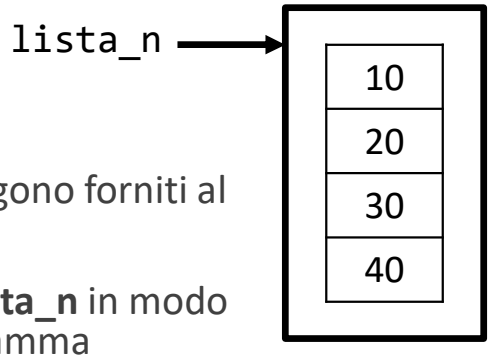
Liste (1)

- E’ un tipo predefinito di Python
- Una lista è una sequenza ordinata di valori, ognuno identificato da un indice
- Una lista è un contenitore che memorizza una raccolta o collezione di elementi, disposti linearmente o, con un sinonimo, in ordine sequenziale
- I valori che fanno parte della lista sono chiamati **elementi**
- Le liste sono simili alle stringhe (che sono insiemi ordinati di caratteri)
- Mentre in una stringa i valori sono dei caratteri, in una lista possono essere di qualsiasi tipo
- Liste e stringhe (e altri tipi di dati che si comportano da insiemi ordinati) sono chiamate sequenze
- Le liste sono in grado di aumentare automaticamente la propria dimensione quando serve per accogliere i nuovi elementi che vi vengono aggiunti o che vengono rimossi

Liste (2)

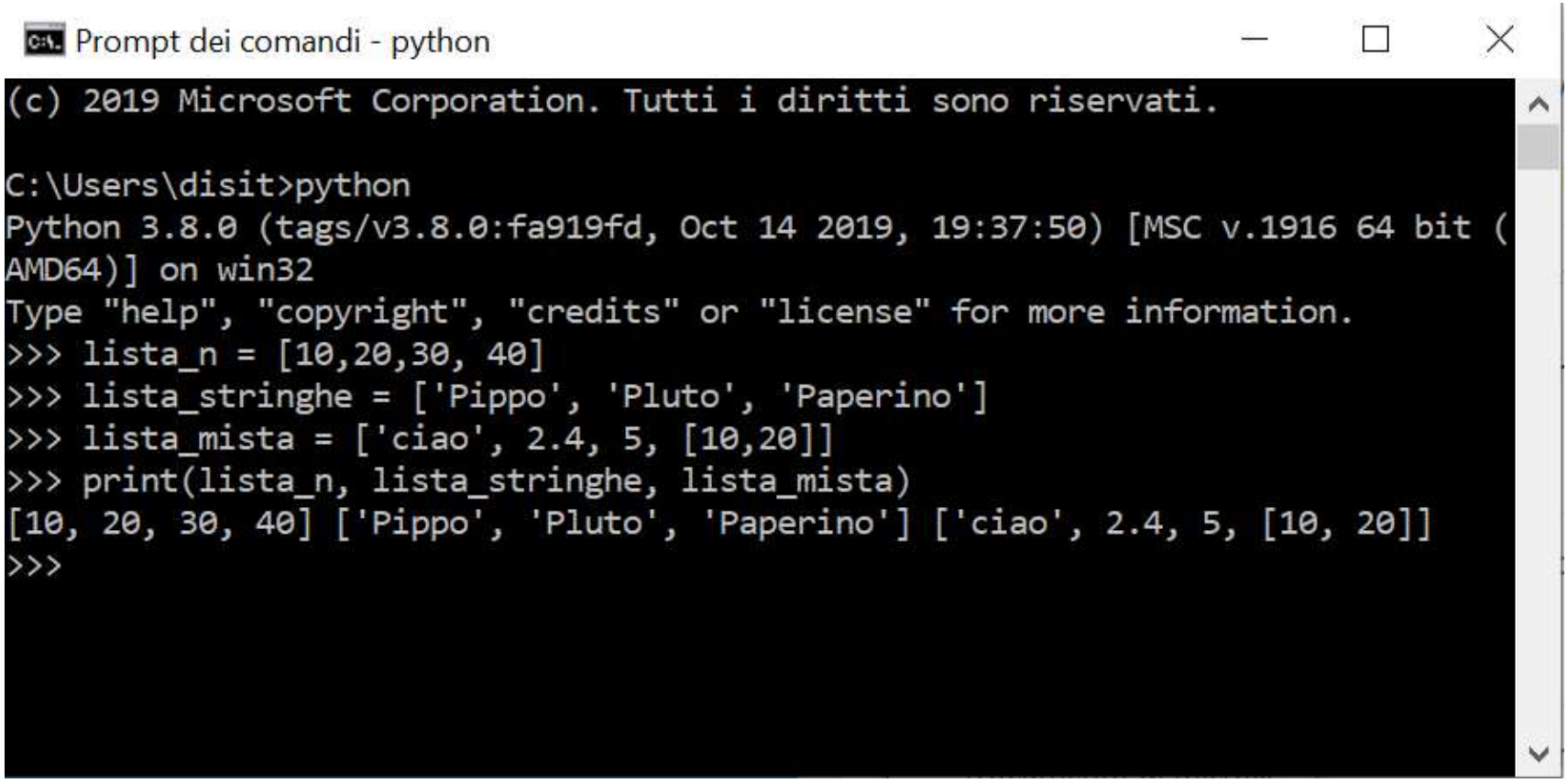
- Ci sono parecchi modi di creare una lista nuova, e quello più semplice è racchiudere i suoi elementi tra parentesi quadrate:

```
lista_n = [10, 20, 30, 40]   # si assegna una lista di 4 interi alla
                                                    # variabile lista_n

#lista di 3 stringhe:
lista_stringhe = ["Pippo", "Pluto", "Paperino"]
#lista con numeri, stringhe, liste
lista_mista = ["ciao", 2.0, 5, [10, 20]]
```



- Gli elementi di una lista vengono memorizzati nello stesso ordine in cui vengono forniti al momento della creazione della lista
- Nell'esempio gli elementi della lista sono stati memorizzati nella variabile **lista_n** in modo da poter accedere agli elementi anche successivamente nel corso del programma
- Una lista all'interno di un'altra lista è detta lista annidata o nidificata
- Una lista che non contiene elementi è detta lista vuota: []

Liste (3)


Prompt dei comandi - python
(c) 2019 Microsoft Corporation. Tutti i diritti sono riservati.
C:\Users\disit>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> lista_n = [10,20,30, 40]
>>> lista_stringhe = ['Pippo', 'Pluto', 'Paperino']
>>> lista_mista = ['ciao', 2.4, 5, [10,20]]
>>> print(lista_n, lista_stringhe, lista_mista)
[10, 20, 30, 40] ['Pippo', 'Pluto', 'Paperino'] ['ciao', 2.4, 5, [10, 20]]
>>>

Accesso agli elementi di una lista (1)

- Una lista è una sequenza di elementi, ciascuno dei quali è associato ad una posizione o indice (che è un numero intero)
- Per accedere ad un elemento di una lista:
  - Si specifica quale indice si vuole usare usando l'operatore di indicizzazione ([])

```
lista_n = [10, 20, 30, 40]

print(lista_n[0])
```

- Si ricorda che l'indice inizia da 0
- Il meccanismo è lo stesso di quello viso per accedere ai singoli caratteri (elementi) di una stringa
- Questo perché sia le liste che le stringhe sono sequenze e l'operatore [] può essere usato per accedere ad u elemento di una qualunque sequenza

Differenze tra Liste e Stringhe (1)

- Tipi gestiti:
  - Le stringhe sono sequenze di caratteri
  - Le liste possono memorizzare valori di qualsiasi tipo
- Mutabili o immutabili?
  - Le stringhe sono immutabili, cioè non è possibile modificare i caratteri di una sequenza
  - Le liste sono mutabili, ovvero è possibile sostituire l'elemento di una lista con un altro elemento
- Si può sostituire un elemento della lista nel modo seguente:

```
lista_n[0] = 100
```

- A questo punto l'elemento di indice 0 contiene il valore 100

Liste: sintassi

Sintassi:

Per creare una lista: [valore1, valore2, …., valoreN]

Per accedere ad un elemento riferimentoLista(nomelista)[indice]

Esempio:

Nome di variabile di tipo lista

```
lista_vuota = []
lista_n = [10, 20, 30, 40]
```

Crea una lista vuota

Crea una lista con valori iniziali

Per accedere ad un elemento si usano le parentesi quadre:

```
lista_n[i] = valore
element = lista_n[i]
```

Differenze tra Liste e Stringhe (2)

```
#Liste
lista_n = [10, 20, 30, 40]
print('valore iniziale di lista_n[0]:', lista_n[0])
lista_n[0] = 100
print(f'valore dopo la mdifica di lista_n[0]:{lista_n[0]}')
#racchiudere la variabile all'interno del {}
# per visualizzarne il valore in uscita

#Stringhe
stringa = 'Hello World!'
print(stringa)
print('primo carattere:',stringa[0])
stringa[0] = 'h'
```

- Tipi gestiti:
  - Le stringhe sono sequenze di caratteri
  - Le liste possono memorizzare valori di qualsiasi tipo
- Mutabili o immutabili?
  - Le stringhe sono immutabili, cioè non è possibile modificare i caratteri di una sequenza
  - Le liste sono mutabili, ovvero è possibile sostituire l'elemento di una lista con un altro elemento

```
valore iniziale di lista_n[0]: 10
valore dopo la mdifica di lista_n[0]: 100
Hello World!
primo carattere: H
Traceback (most recent call last):
  File "main.py", line 11, in <module>
    stringa[0] = 'h'
TypeError: 'str' object does not support item
 assignment
>
```

Accesso e Sostituzione ad/di un elemento

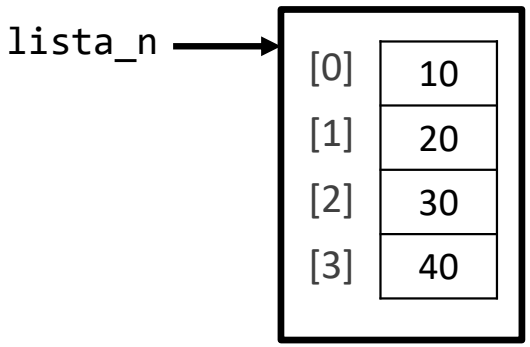
- Accesso ad un elemento della lista:

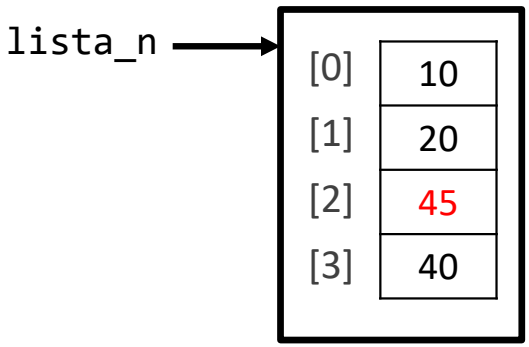
```
lista_n[0], primo elemento
lista_n[1], secondo elemento
lista_n[2], terzo elemento
lista_n[3], quarto elemento
```

- Sostituzione del terzo elemento della lista:

```
lista_n[2] = 45
```

Nota: in questa lista gli indici validi sono quelli che vanno da 0 a 3





Visita di una lista ed errore di intervallo (1)

- Data una lista di N elementi, sappiamo che gli indici per identificare gli elementi partono da 0 e quindi stanno nell'intervallo: [0, N-1] (compresi)
- ATTENZIONE! Il tentativo di accesso ad un elemento con indice che sta al di fuori di quello consentito genera un **errore di intervallo**
- In una lista di dieci elementi:

Dispari = [1,3,5,7,11,13,15,17,21, 23]

- NON esiste Dispari[0]

```
Dispari = [1,3,5,7,11,13,15,17,21, 23]
print(Dispari[10])
```

```
Traceback (most recent call last):
  File "main.py", line 2, in <module>
    print(Dispari[10])
IndexError: list index out of range
```

Visita di una lista

- Per visitare una lista evitando gli errori di indice, conviene sapere sempre il numero di elementi che appartengono alla lista PRIMA di visitarla
- Per ottener la lunghezza di una lista, cioè il numero dei suoi elementi, si può usare la funzione len:

```
num_elem = len(Dispari)
```

- Ci sono due modi di utilizzo delle parentesi quadre quando si parla di Liste:
  - SE le parentesi quadre seguono il nome (o riferimento) di una variabile, sono interpretate come operatore di indicizzazione e servono per accedere ai singoli elementi della lista

```
Dispari[9]
```

  - SE le parentesi quadre NON seguono il nome di una variabile, creano una Lista

```
Dispari = [9] #assegna a Dispari la lista che contiene il solo valore 9
```

Scansione di una lista (1)

- Ci sono due modi per visitare tutti gli elementi di una lista
  - CASO1: Eseguire un ciclo che usa tutti i valori validi di un indice all'interno della lista, ispezionando ciascun elemento tramite tali indici
  - CASO2: Eseguire un ciclo che prenda direttamente in esame tutti gli elementi
- CASO1: Si vuole fare in modo che l'indice i assuma tutti i valori possibili

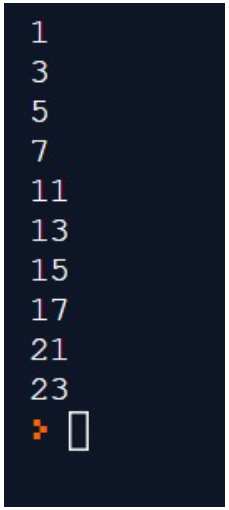
```
Dispari = [1,3,5,7,11,13,15,17,21, 23]
for i in range(len(Dispari)):
    print('indice', i,'---> valore', Dispari[i])
```

```
indice 0 ---> valore 1
indice 1 ---> valore 3
indice 2 ---> valore 5
indice 3 ---> valore 7
indice 4 ---> valore 11
indice 5 ---> valore 13
indice 6 ---> valore 15
indice 7 ---> valore 17
indice 8 ---> valore 21
indice 9 ---> valore 23
```

Scansione di una lista (2)

- Ci sono due modi per visitare tutti gli elementi di una lista
  - CASO1: Eseguire un ciclo che usa tutti i valori validi di un indice all'interno della lista, ispezionando ciascun elemento tramite tali indici
  - CASO2: Eseguire un ciclo che prenda direttamente in esame tutti gli elementi
- CASO2: Scrivere un ciclo che scansioni direttamente gli elementi

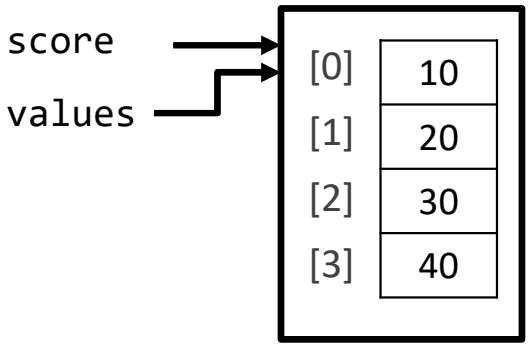
```
Dispari = [1,3,5,7,11,13,15,17,21, 23]
for element in Dispari:
    print(element)
```



Riferimenti ad una Lista

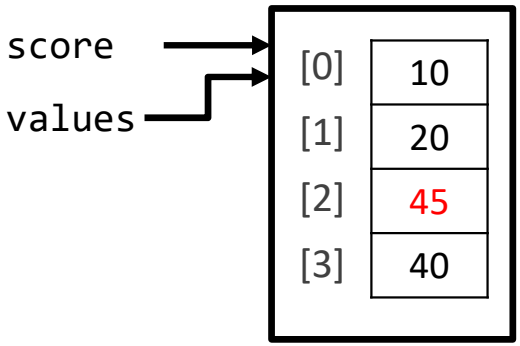
- **Assegnazione**(di struttura):

```
score = [10,20,30,40]
values = score
```



- Assegnazione (di elementi di una struttura):

```
score[2] = 45
print(values[2])#stampa 45
```



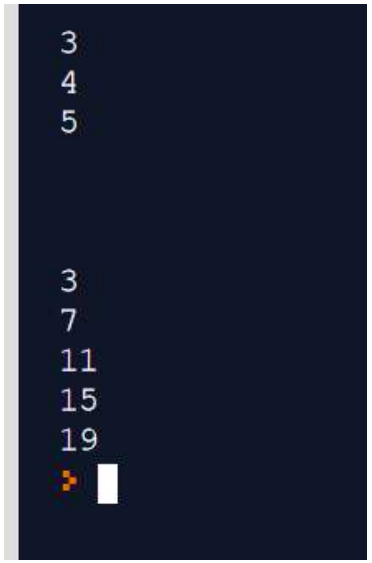
- Guardando in figura, si nota che la variabile values NON memorizza nessuno dei numeri della collezione
- La lista è memorizzata altrove e la variabile values contiene un riferimento alla lista
- DOPO aver copiato una variabile di tipo lista in un'altra ENTRAMBE fanno riferimento alla stessa lista. A questo punto si può modificare o fare riferimento al contenuto di una lista usando indifferentemente una delle due variabili

Liste: range(…) (1)

- Le liste che contengono numeri interi consecutivi sono così comuni che Python fornisce un modo semplice per crearle:
- **range**(*start, stop, step*)
  - start è il valore di inizio
  - stop, valore di fine (NON compreso)
  - step, differenza tra un valore ed il successivo

  (default value = 1)

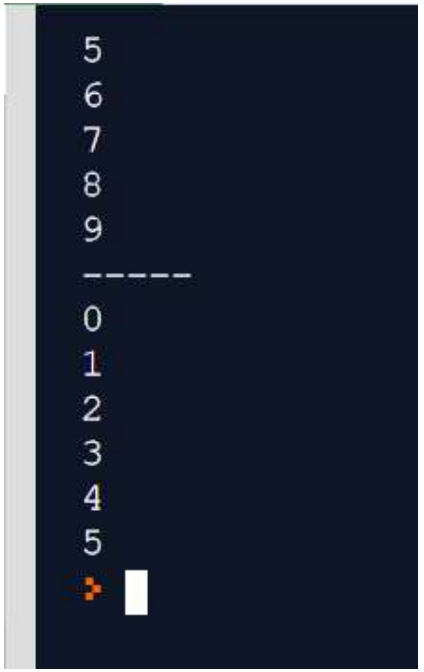
```
x = range(3, 6) #step=1 per default
for n in x:
     print(n)

print('\n\n’)

x = range(3, 20, 4) #step = 4
for n in x:
     print(n)
```



Liste: range (…) (2)

- range(), con solo uno o due parametri in ingresso:

```
x = range(5,10) # range(), con due parametri in ingresso:
                # step = 1 di default
for n in x:
  print(n)

print('-----')

x = range(6)  # range(), con solo un parametro in ingresso
              # start=1 di default
              # step = 1 di default,
for n in x:
  print(n)
```



Accesso agli elementi di una lista (1)

- Esiste una lista speciale che non contiene alcun elemento ed è chiamata lista vuota (indicata da [])
- La sintassi per l'accesso agli elementi di una lista è la stessa che abbiamo già visto per i caratteri di una stringa: anche in questo caso facciamo uso dell'operatore porzione ([])
- L'espressione tra parentesi quadrate specifica l'indice dell'elemento (non dimenticare che gli indici partono da 0!)

```
x = range(5,10) #step = 1 di default
for n in x:
    print(n)

print("\n")
print(x[2]) #stampo il terzo elemento della lista
print(x[7-3]) #la differenza da' numero intero -> stampo il quinto elemento
print(x[-2]) #il conteggio parte dalla fine, x[-1] è l'ultimo elemento della lista

print(x[10]) #da' errore 'out of range' #perchè la lista contiene SOLO 5 elementi!
```

Accesso agli elementi di una lista (2)

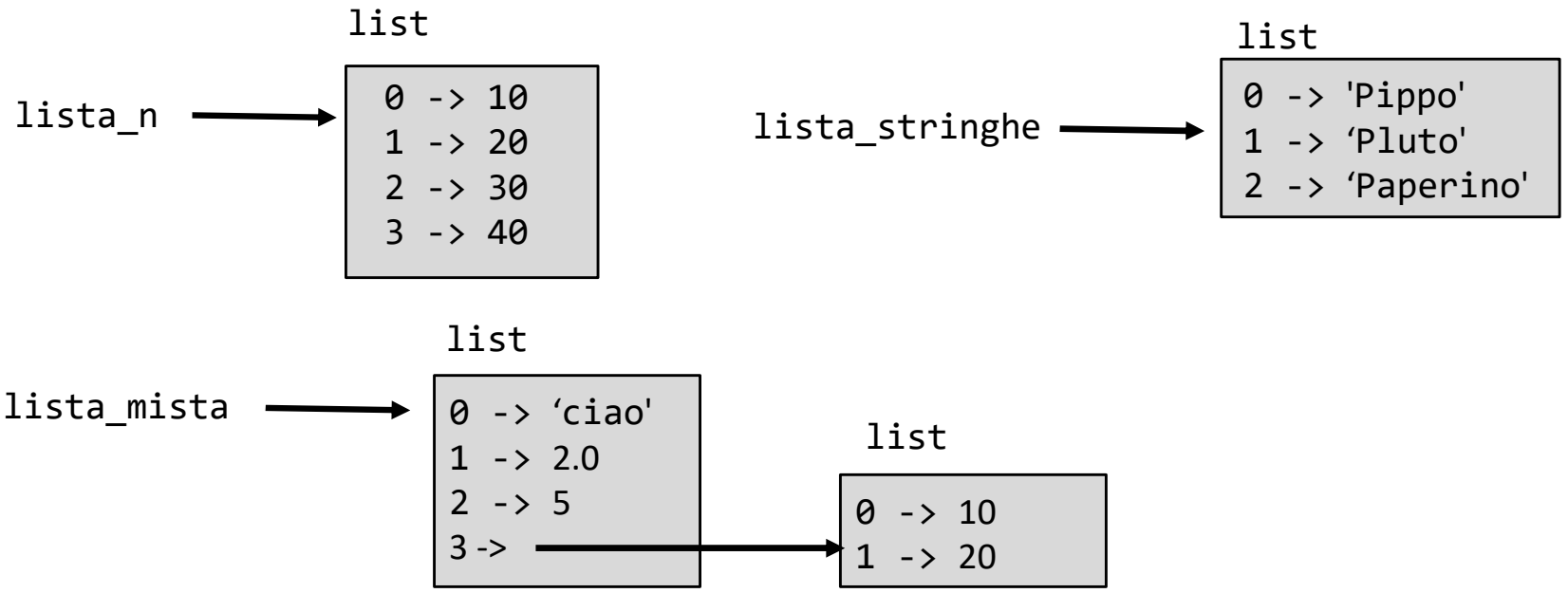
main.py saved

```
1   x = range(5,10) #step = 1 di default
2   for n in x:
3     print(n)
4
5   print("\n")
6   print(x[2]) #stampo il terzo elemento della lista
7   print(x[7-3]) #la differenza da' numero intero
8
9   print(x[-2]) #il conteggio parte dalla fine
10
11  #print(x[15]) #da' errore 'out of range'
12  #perchè la lista contiene SOLO 5 elementi!
```

```
5
6
7
8
9


7
9
8
```

Liste (6): Diagrammi di stato

```
lista_n = [10, 20, 30, 40]
lista_stringhe = ['Pippo', 'Pluto', 'Paperino']
lista_mista = ['ciao', 2.0, 5, [10, 20]]
```



Attraversamento di una lista (1)

- E' comune usare una variabile di ciclo come indice di una lista:

```
Squadre = ["Fiorentina", "Juventus", "Inter", "Milan", "Roma"]
i = 0
while i < 5:
    print(Squadre[i])
    i = i + 1
#altro metodo
for i in Squadre:
    print(i)
```

- In ogni ciclo la variabile i è usata come indice della lista: questo tipo di elaborazione è chiamata **elaborazione trasversale di una lista** o **attraversamento di una lista**

Attraversamento di una lista (2)

main.py saved

```
1   Squadre = ["Fiorentina","Juventus", "Inter", "Milan", "Roma"]
2   i = 0
3   while i < 5:
4       print(Squadre[i])
5       i = i + 1
6   print("\n")
7   for i in Squadre:
8       print(i)
9
10
```

```
Fiorentina
Juventus
Inter
Milan
Roma


Fiorentina
Juventus
Inter
Milan
Roma
```

Le liste sono mutabili (1)

- A differenza delle stringhe, le liste sono **mutabili**, ovvero gli elementi possono essere modificati

```
lista_n = [10, 20, 30, 40]
lista_n [2] = 67
```

- Usando l'operatore **porzione** nella parte sinistra dell'**assegnazione** possiamo aggiornare un elemento:

```
Squadre = ["Fiorentina", "Juventus", "Inter", "Milan", "Roma"]
i = 0
while i < 5:
    print(Squadre[i])
    i = i + 1
print("\n")
Squadre[2] = "Napoli"
for i in Squadre:
    print(i)
```

```
Fiorentina
Juventus
Inter
Milan
Roma


Fiorentina
Juventus
Napoli
Milan
Roma
```

Le liste sono mutabili (2)

```
lista = ["a", "b", "c", "d", "e", "f", "g", "h"]
i = 0
while i < 8:
   print(lista[i])
   i = i + 1
print("\n")
lista[2:4] = [ "C", "D", "E"] #modifica di più elementi alla volta
i=0
for i in lista:
   print(i)
print("\n")
lista[5:6] = [] #rimozione di uno (o più) elementi dalla lista
lista[6:6] = ["t"] #aggiunta di un elemento nella
#sezione vuota
for i in lista:
  print(i)
```

- Con l'operatore ***porzione ':'*** (Si parla anche di '***SLICING***' di una lista) possiamo:
  - modificare più elementi alla volta
  - rimuovere più elementi alla volta
  - Aggiungere più elementi alla volta nelle sezioni vuote della lista

Le liste sono mutabili (3)

main.py saved

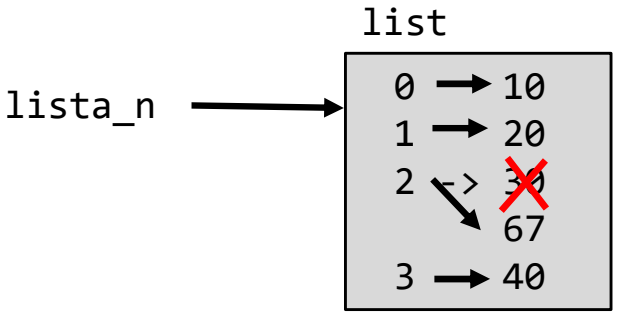
```
1   lista = ["a", "b", "c", "d", "e", "f", "g", "h"]
2   i = 0
3   while i < 8:
4       print(lista[i])
5       i = i + 1
6   print("\n")
7   lista[2:4] = [ "C", "D", "E"] #modifica di più elementi
    alla volta
8   i=0
9   for i in lista:
10      print(i)
11  print("\n")
12  lista[5:6] = [] #rimozione di uno (o più) elementi
    dalla lista
13  lista[6:6] = ["t"] #aggiunta di un elemento nella
14  #sezione vuota
15  for i in lista:
16      print(i)
```

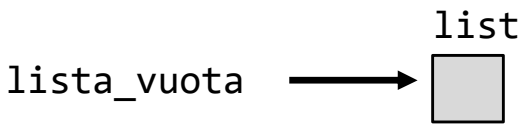
```
a
b
c
d
e
f
g
h

a
b
C
D
E
e
f
g
h
```

```
a
b
C
D
E
f
t
g
h
```

Liste (7): Diagrammi di stato

```
lista_n = [10, 20, 30, 40]
lista_n [2] = 67  #variazione del terzo elemento, si assegna al terzo elemento
                  #il nuovo valore 67
Lista_vuota = []
```





Attraversamento di una lista (1)

- Il modo che viene usato più frequentemente per attraversare una lista è il ciclo for
- La sintassi è la stessa usata per le stringhe:

```
numeri = [10, 20, 30, 40]
for numero in numeri:
    print(numero)
```

- Il metodo visto sopra funziona bene per leggere gli elementi di una lista
- Per scrivere o per aggiornare gli elementi è utile usare gli indici
- Un modo per farlo è combinare range e len:

```
for i in range(len(numeri)):
    numeri[i] = numeri[i]*2
    print(numeri[i])
```

Attraversamento di una lista (2)

- Per scrivere o per aggiornare gli elementi è utile usare gli indici si combina range e len:

```
for i in range(len(numeri)):
    numeri[i] = numeri[i]*2
    print(numeri[i])
```

- Si noti che:
  - len, restituisce il numero degli elementi di una lista
  - range, restituisca la lista di indici da 0 a n-1, con n = lunghezza lista
- Ad ogni ripetizione del ciclo, i assume il valore dell'indice dell'elemento successivo
- L'istruzione di assegnazione che si trova nel corpo della funzione, usa i per leggere il vecchio valore dell'elemento e assegnare quello nuovo

Attraversamento di una lista (3)

main.py saved

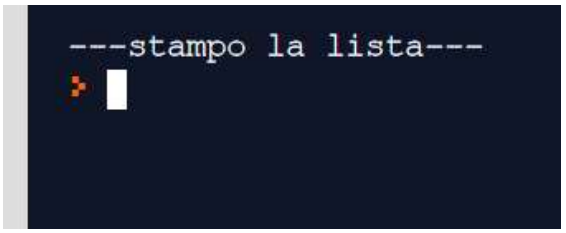
```
1   numeri = [10, 20, 30, 40]
2   print('---stampo la lista---')
3   for numero in numeri:
4       print(numero)
5
6   print('---modifico la lista
    ---\n---e stampo la lista
    mofìdificata:---')
7   for i in range(len(numeri)):
8       numeri[i] = numeri[i]*2
9       print(numeri[i])
```

```
---stampo la lista---
10
20
30
40
---modifico la lista ---
---e stampo la lista mofìdificata:---
20
40
60
80
```

Attraversamento di una lista (4)

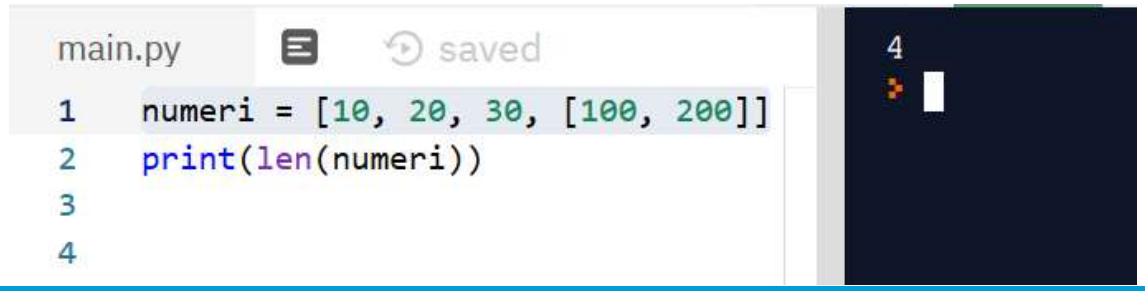
- Un ciclo for su una lista vuota NON esegue mai il corpo:

```
numeri = []
print('---stampo la lista---')
for numero in numeri:
    print('Non entra mai!’)
```



- Si noti inoltre che, nel caso di liste annidate, la lista nidificata conta sempre come un singolo elemento. La lunghezza edlla lista seguente vale 4:

```
numeri = [10, 20, 30, [100, 200]]
```



Operazioni sulle Liste

- Concatenazione: si usano gli operatori '+' e '*'
- Uso dell'operatore '+':

```
a = [1,2,3,4]
b = [0,20]
c = a+b
print(c)
```

```
[1, 2, 3, 4, 0, 20]
>
```

- Uso dell'operatore '*'

…

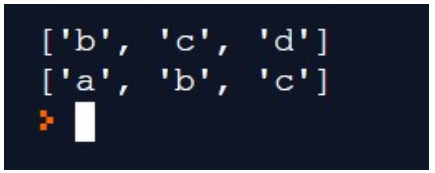
```
d = b*3
e = [9]*4
print(d)
print(e)
```

```
[0, 20, 0, 20, 0, 20]
[9, 9, 9, 9]
>
```

Slicing delle liste (1)

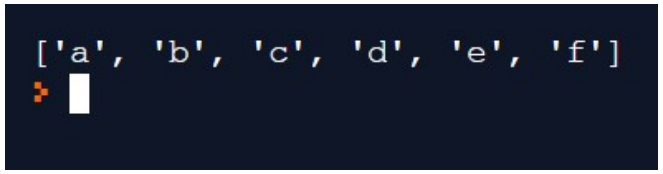
- Operatore di Slicing, Sintassi : nome_lista[indice_1:indice_2]
- L'operazione di Slicing funziona anche sulle liste:

```
t = ['a','b','c','d','e','f']
print(t[1:4])
print(t[:3])
```



- Se si omette il primo indice, lo slicing comincia dall'inizio (se il primo indice manca, allora il suo valore di default è 0)
- Se si omettono entrambi gli indici, lo slicing è una copia della intera lista

```
print(t[:])
```



- Sapendo che le liste sono MUTABILI, spesso è utile fare una copia delle liste, prima di eseguire operazioni su di esse che le modifichino

Slicing delle liste (2)

- Sempre ricordando che le liste sono MUTABILI, l'operatore di Slicing permette anche di modificare una lista SE sta alla sinistra dell'operatore di assegnazione

```
t = ['a','b','c','d','e','f']
print(t[1:4])
t[1:4] = ['ecco','sovrascrivo','adesso']
print(t)
```

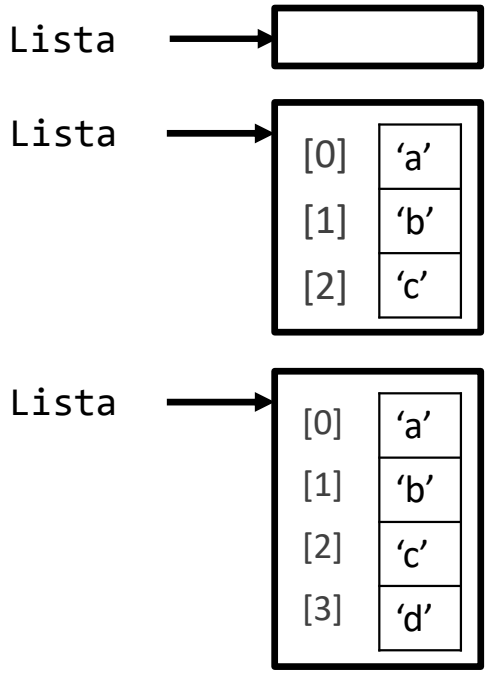
- In questo caso infatti NON stiamo copiando la lista ma la stiamo MODIFICANDO, stiamo sovrascrivendo /Aggiornando i valori

Metodi delle liste: append e extend (1)

- Python fornisce dei metodi che operano sulle liste
- **append,** aggiunge un nuovo elemento in CODA alla lista:

```
#metodi delle liste
Lista = []
Lista = ['a', 'b', 'c']
Lista.append('d')
print(Lista)
```

```
['a', 'b', 'c', 'd']
>
```



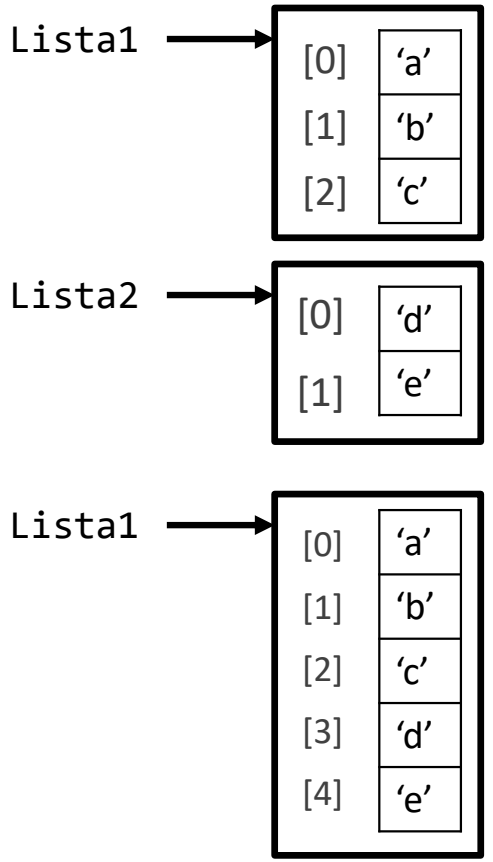
Metodi delle liste: append e extend (2)

- Python fornisce dei metodi che operano sulle liste
- **extend**, concatena due liste

```
#metodi delle liste; extend
Lista1 = ['a', 'b', 'c']
Lista2 = ['d', 'e']
Lista1.extend(Lista2)
print(Lista1)
```

```
['a', 'b', 'c', 'd', 'e']
>
```

- ATTENZIONE! L'operazione di extend MODIFICA la Lista (Lista1) a cui vengono concatenati gli elementi della seconda lista che invece resta invariata (Lista2)



Inserire un elemento (1)

Metodi delle liste: extend

```
Lista1 = ['a', 'b', 'c']
Lista2 = ['d', 'e']
print('Stampo Lista1')
print(Lista1)
print('Stampo Lista2')
print(Lista2)
Lista1.extend(Lista2)
print('Stampo Lista1')
print(Lista1)
print('Stampo Lista2')
print(Lista2)
```

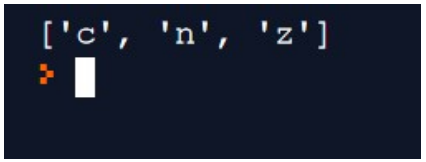
- ATTENZIONE! L'operazione extend MODIFICA la Lista (Lista1) a cui vengono concatenati gli elementi della seconda lista che invece resta invariata (Lista2)

```
Stampo Lista1
['a', 'b', 'c']
Stampo Lista2
['d', 'e']
Stampo Lista1
['a', 'b', 'c', 'd', 'e']
Stampo Lista2
['d', 'e']
```

Metodi delle liste: sort (1)

sort, dispone gli elementi della lista in ordine decrescente, sia per stringhe:

```
#metodi delle liste: sort
Lista = ['z', 'n', 'c']
Lista.sort()
print(Lista)
```



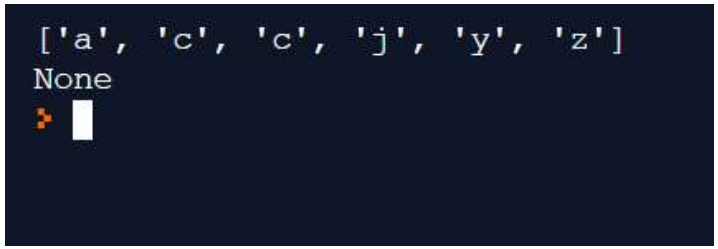
…che per numeri:

```
#metodi delle liste: sort
Lista = [23, 3, 67]
Lista.sort()
print(Lista)
```

Metodi delle Liste: sort (2)

- Questo metodo serve per disporre gli elementi di una lista in modo ordinato

```
Lista1 = ['a', 'z', 'c', 'y', 'c', 'j' ]

Lista2 = Lista1.sort() #sort() è un metodo vuoto!

print(Lista1)

print(Lista2)
```



- ATTENZIONE sort() è un metodo vuoto ovvero senza output, si occupa solo di modificare la lista su cui gli viene chiesto di effettuare l'ordinamento, NON rende niente in output, e infatti... 'None'

Mappare, Filtrare e Ridurre (1) - riduzione

- Per sommare tutti i numeri in una lista, si può ad esempio usare la seguente funzione:

```
#Sommare tutti i numeri in una lista
#concetto di RIDUZIONE (da N elementi a 1 elemento)
def somma_tutti(t):
    totale = 0 #inizializzato a zero
    for x in t: #ciclo sui valori di t, x è il valore di t che si aggiorna
                ad ogni ciclo, il primo valore è t[0], il secondo t[1],
                etc.. Finché ci sono elementi in t
        totale += x #ad ogni ciclo aggiungo il valore di x a totale
    return totale
#main…
Lista = [23, 3, 67, 100, 2]
print('Ecco la lista: ', Lista)
print(somma_tutti(Lista))
#usando funzione predefinita sum
print(sum(Lista))
```

```
Ecco la lista:  [23, 3, 67, 100, 2]
195
195
```

- Una operazione come questa, che compatta una serie di dati in un unico dato, è detta **riduzione**
- Sommare gli elementi di una lista è una operazione comune e Pyhton fornisce una apposita funzione predefinita 'sum'

Mappare, Filtrare e Ridurre (2) - mappa

- A volte è necessario attraversare una lista per costruirne un'altra contemporaneamente:

```
def tutte_maiuscole(t):
    res = [] #res è inizializzata come lista vuota
    for s in t:
            res.append(s.capitalize())    #ad ogni ciclo, si accoda un
elemento a res
    return(res) #NOTA: stringa.capitalize() rende i caratteri di stringa
Maiuscoli
#main
Lista = ['angelo', 'b', 'carta']
Maiscole = tutte_maiuscole(Lista) #ho due liste diverse e le stampo
print('Stampo Lista:')
print(Lista)
print('Stampo Maiuscole:')
print(Maiscole)
```

- Una operazione come questa, che applica una funzione su ciascun elemento di una sequenza (Lista), è detta **mappa. Questa è una operazione effettuata a livello globale e non sul singolo elemento.**

Mappare, Filtrare e Ridurre (3) - filtro

- Un'altra operazione frequente è quella di selezionare alcuni elementi di una lista per formare una **sottolista**

```
def solo_maiuscole(t):
    res = []
    for s in t:
        if s.isupper(): #vero se tutti i caratteri di s sono maiuscoli
            res.append(s)
    return res
#main
Lista = ['Ab', 'b', 'C', 'DADO', 'Ciclone']
filtro = solo_maiuscole(Lista)
print ('Lista iniziale: ', Lista)
print('Stampo solo le stringhe maiuscole:')
print (filtro)
```

- Una operazione di questo tipo, che seleziona solo alcuni elementi filtrandone altri, è detta **filtro**

```
Lista iniziale:  ['Ab', 'b', 'C', 'DADO', 'Ciclone']
Stampo solo le stringhe maiuscole:
['C', 'DADO']
```

Cancellare elementi da una lista: POP (1)

- Ci sono vari metodi per cancellare un elemento da una lista
- Se si conosce l'indice dell'elemento desiderato, si può usare pop
- Sintassi - nome_Lista.pop(indice):

```
#Cancellazione elemento da una lista: POP
Lista = ['a', 'b', 'c', 'd']
print('Lista prima della cancellazione:')
print(Lista)
x = Lista.pop(1)
print('Lista dopo la cancellazione:')
print(Lista)
print('Valore di ritorno di pop:')
print(x)
print('Significa che ho rimosso:', x)
```

```
Lista prima della cancellazione:
['a', 'b', 'c', 'd']
Lista dopo la cancellazione:
['a', 'c', 'd']
Valore di ritorno di pop:
b
Significa che ho rimosso: b
>
```

- Nota: il valore di ritorno di pop è l'elemento che è stato rimosso dalla lista (Lista)

Cancellare elementi da una lista: remove

- Se si conosce l'elemento da rimuovere ma non il suo indice, allora si può usare remove
- Sintassi nome_lista.remove(valore_elemento):

```
#Cancellazione elemento da una lista: Remove
Lista = ['a', 'b', 'c', 'd']
print('Lista prima della cancellazione:')
print(Lista)
x = Lista.remove('b')
print('Lista dopo la cancellazione:')
#Lista con numeri
print(Lista) L_mista = [10, 'Stringa', 22, 45]
print('Lista prima della cancellazione:')
print(L_mista)
x = L_mista.remove(22)
print('Lista dopo la prima cancellazione:')
print(L_mista)
x = L_mista.remove('Stringa')
print('Lista dopo la seconda cancellazione:')
print(L_mista)
print(x)
```

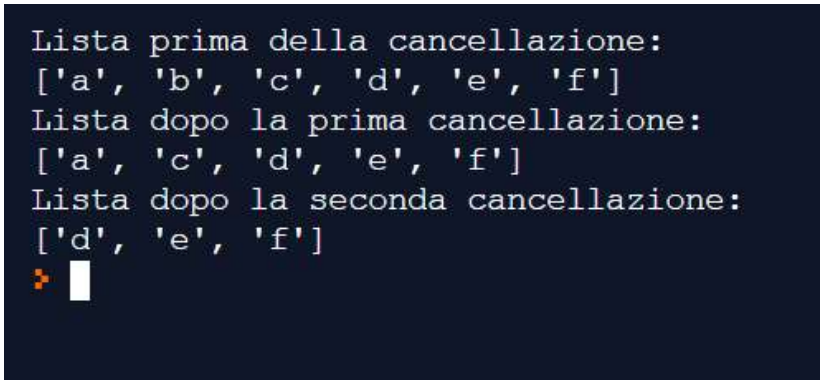
```
Lista prima della cancellazione:
[10, 'Stringa', 22, 45]
Lista dopo la prima cancellazione:
[10, 'Stringa', 45]
Lista dopo la seconda cancellazione:
[10, 45]
None
```

- NOTA: il valore di ritorno di remove è None

Cancellare elementi da una lista: del

- Se non serve l'elemento rimosso, si può usare del:

```
#Cancellazione di un elemento da una lista: del
Lista = ['a', 'b', 'c', 'd', 'e', 'f']
print('Lista prima della cancellazione:')
print(Lista)
del Lista[1]
print('Lista dopo la prima cancellazione:')
print(Lista)
del Lista[0:2] #Cancellazione di più di un elemento da una lista: slicing
print('Lista dopo la seconda cancellazione:')
print(Lista)
```



Cancellare elementi da una lista: slicing (1)

- Per cancellare più di un elemento alla volta, si usa lo slicing:

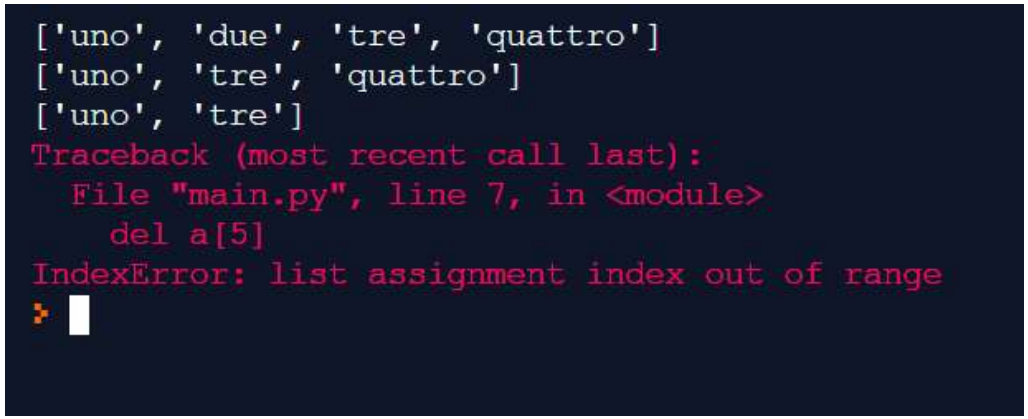
```
#Cancellazione elemento da una lista: slicing
Lista = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
print('Lista prima della cancellazione:')
print(Lista)
del Lista[2:6]
print('Lista dopo la cancellazione:')
print(Lista)
```

```
Lista prima della cancellazione:
['a', 'b', 'c', 'd', 'e', 'f', 'g']
Lista dopo la cancellazione:
['a', 'b', 'g']
```

Cancellare elementi da una lista: slicing (2)

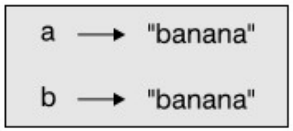
- del gestisce anche indici negativi e rende un messaggio di errore se l'indice è al di fuori dei limiti ammessi

```
a = ['uno', 'due', 'tre', 'quattro']

print(a)
del a[1]
print(a)
del a[-1]
print(a)

del a[5]
#-> IndexError: list assignment index out of range
```

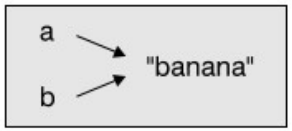


Oggetti e Valori (1)

- Se eseguiamo queste istruzioni:
  - a = "banana"
  - b = "banana"



- Caso a)



- Caso b)

- Sia a che b si riferiscono ad una stringa contenente le lettere "banana"
- A prima vista non possiamo dire se puntano alla stessa stringa in memoria
- I possibili casi sono due:
  - **Caso a):** a e b si riferiscono a due diverse “cose” che hanno lo stesso valore
  - **Caso b):** a e b si riferiscono alla stessa “cosa”
- Queste “cose” hanno un nome: oggetti
- Un oggetto è qualcosa a cui una variabile può **fare riferimento**
- Ogni oggetto ha un **identificatore unico** che possiamo ricavare con la funzione **id**
- Stampando l’identificatore di a e di b possiamo dire subito se le due variabili si **riferiscono allo stesso oggetto**

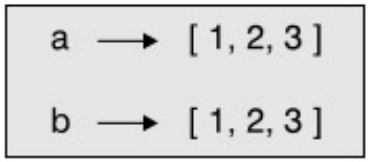
Oggetti e Valori (2) - identificatore unico

- Stampando l'identificatore si ottiene lo stesso id per entrambe le **stringhe**

- Questo significa che Python ha creato in memoria un'unica stringa a cui fanno riferimento entrambe le variabili a e b

Oggetti e Valori (3) - identificatore unico

- Le liste si comportano diversamente dalle stringhe: gli identificativi sono sempre diversi
- **Infatti quando si creano due liste queste sono sempre oggetti diversi**
- Il diagramma di stato risulta infatti:



main.py saved

```
1   print("id stringhe:")
2   a='banana'
3   b='banana'
4   print(id(a))
5   print(id(b))
6   print("\n\n")
7   print("id Liste:")
8   La=['1','2','3']
9   Lb=['1','2','3']
10  print(id(La))
11  print(id(Lb))
```

```
id stringhe:
139868772759152
139868772759152


id Liste:
139868775235744
139868842368048
```

Oggetti e Valori (4) – operatore is

- Una ulteriore modalità per verificare se le variabili o le strutture dati con cui stiamo lavorando si riferiscono o meno allo stesso oggetto, è quello di usare l'operatore is

```
a = 'banana'
b = 'banana'
print("Lavoriamo con due stringhe che
hanno lo stesso valore.
Si riferiscono allo stesso oggetto?")
print(a is b)
print("\nLavoriamo con due Liste che hanno gli stessi
elementi. Si riferiscono allo stesso oggetto?")
a = [1,2,3]
b = [1,2,3]
print(a is b)
```

```
Lavoriamo con due stringhe che hanno lo stesso valore.
 Si riferiscono allo stesso oggetto?
True

Lavoriamo con due Liste che hanno gli stessi elementi.
 Si riferiscono allo stesso oggetto?
False
```

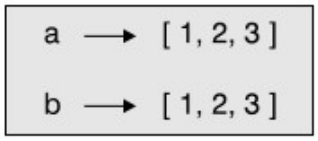
Alias (1)

- Le variabili si riferiscono ad oggetti
- Quando assegniamo una variabile ad un'altra variabile, entrambe le variabili si riferiscono allo stesso oggetto:

```
La=['1','2','3']
Lb=['1','2','3']
Lc = La
print("Id di La:")
print(id(La))
print("Id di Lb:")
print(id(Lb))
print("Id di Lc:")
print(id(Lc))
```

```
Id di La:
139658906111808
Id di Lb:
139658906111616
Id di Lc:
139658906111808
```

- Diagramma di stato:



Alias (2)

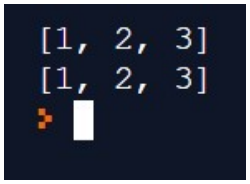
- Una stessa lista in questo caso ha due nomi differenti (a e c).
- Si dice che tali nomi sono due **alias**

- Dato che l'oggetto cui entrambi si riferiscono è lo stesso, è indifferente quale degli alias si usi per effettuare un'elaborazione
- Sebbene questo comportamento possa essere desiderabile, è nella maggior parte dei casi, difficilmente controllabile e può portare ad effetti indesiderati e inattesi
- In generale è buona norma **evitare gli alias in caso di oggetti mutabili**, mentre per quelli immutabili non ci sono problemi
- Ecco perché Python si permette di usare la **stessa stringa con diversi alias** quando si tratta di risparmiare memoria senza che questo fatto causi alcun problema
- **La stringa è un oggetto immutabile e quindi non può essere modificata**: non c'è quindi il rischio di causare spiacevoli effetti collaterali

Clonare le liste (1)

- Se vogliamo modificare una lista e mantenere una copia dell'originale dobbiamo essere in grado di copiare il contenuto della lista e non solo di creare un suo alias
- Questo processo è talvolta chiamato clonazione per evitare l'ambiguità insita nella parola "copia"
- Il modo più semplice per clonare una lista è quello di usare l'operatore porzione (slicing):

```
a = [1, 2, 3]
b = a[:]
print(a)
print(b)
```



Clonare le liste (2)

- Il fatto di prendere una porzione di a crea una nuova lista
- In questo caso la porzione consiste degli elementi dell'intera lista, dato che non sono stati specificati gli indici iniziale e finale
- Ora siamo liberi di modificare b senza doverci preoccupare di a:

```
a = [1, 2, 3]
b = a[:]
print("Lista a:")
print(a)
b[0]=10
print("\nLista a, dopo la modifica
a lista b (invariata):")
print(a)
print("\nLista b modificata:")
print(b)
```

```
Lista a:
[1, 2, 3]

Lista a, dopo la modifica a lista b (invariata):
[1, 2, 3]

Lista b modificata:
[10, 2, 3]
>
```

ATTENZIONE!!! Qual è la differenza?

CASO 1

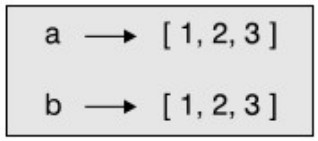
```
a = [1, 2, 3]
b = a[:]
print("\nLista a:")
print(a)
b[0]=10
print("\nLista a, dopo la
modifica a lista b:")
print(a)
print("\nLista b
modificata:")
print(b)
```

CASO 2

```
a = [1, 2, 3]
b = a
print("\nLista a:")
print(a)
b[0]=10
print("\nLista a, dopo la
modifica a lista b:")
print(a)
print("\nLista b
modificata:")
print(b)
```

Caso1: In questo caso è stata clonata la lista

- Siamo nel caso in cui i due puntatori sono diversi (le due liste sono diverse l'una dall'altra, grazie alla operazione di slicing)
- Se faccio modifiche su una lista NON si ripercuoto anche sull'altra perché le due liste sono indipendenti




```
a = [1, 2, 3]
b = a[:]
print("\nLista a:")
print(a)
b[0]=10
print("\nLista a, dopo la modifica a lista b:")
print(a)
print("\nLista b modificata:")
print(b)
```

```
Lista a:
[1, 2, 3]

Lista a, dopo la modifica a lista b:
[1, 2, 3]

Lista b modificata:
[10, 2, 3]
```

Caso 2: In questo caso è stato copiato il riferimento di una lista… (Alias)

- Questo significa che a e b hanno lo stesso id, ovvero puntano alla stessa lista
- Quindi SE faccio una modifica su b, la faccio automaticamente anche su a

```
a = [1, 2, 3]
b = a
print("\nLista a:")
print(a)
b[0]=10
print("\nLista a, dopo la modifica a lista b:")
print(a)
print("\nLista b modificata:")
print(b)
```


```
Lista a:
[1, 2, 3]

Lista a, dopo la modifica a lista b:
[10, 2, 3]

Lista b modificata:
[10, 2, 3]
```

Parametri di tipo lista: passare una lista come INPUT di una funzione (1)

- Se passiamo una lista come parametro di funzione (ad esempio come input di una funzione) in realtà passiamo un suo **riferimento** e non una sua copia
- Per esempio la funzione Testa prende una lista come parametro e ne ritorna il primo elemento:

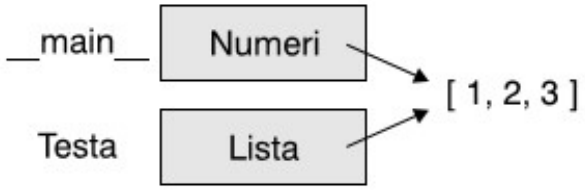
```
def Testa(Lista):
    return Lista[0]
```

```
Lista:  [1, 2, 3]  la cui testa è:  1
>
```

```
Numeri = [1, 2, 3]
print('Lista: ', Numeri, ' la cui testa è: ' ,Testa(Numeri))
```

Parametri di tipo lista: passare una lista come INPUT di una funzione (2)

- L'oggetto Lista è condiviso da due frame, quindi si disegna a cavallo di entrambi



- Se una funzione modifica una lista passata come parametro, viene modificata la lista stessa e **non** una sua copia
- Per esempio CancellaTesta rimuove il primo elemento da una lista

main.py saved

```
1  def Testa(Lista):
2      return Lista[0]
3  def CancellaTesta(Lista):
       del Lista[0]
4
5  Numeri = [1, 2, 3]
6  print(Numeri)
7  print(Testa(Numeri))
8  CancellaTesta(Numeri)
9  print(Numeri)
```

```
[1, 2, 3]
1
[2, 3]
```

Parametri di tipo lista: passare una lista come INPUT di una funzione (3)

- Verifichiamo quanto detto utilizzando gli identificativi:

```
def Testa(Lista):
    return Lista[0]
def CancellaTesta(Lista):
    print('identificativo Lista: ', id(Lista))
    del Lista[0]

Numeri = [1, 2, 3]
print('Elementi iniziali lista Numeri: ',Numeri)
print('identificativo Numeri: ', id(Numeri))
print('Valore della testa: ',Testa(Numeri))
CancellaTesta(Numeri)
print('Elementi lista Numeri, dopo la cancellazione della testa: ',Numeri)
```

```
Elementi iniziali lista Numeri:  [1, 2, 3]
identificativo Numeri:  139947160569280
Valore della testa:  1
identificativo Lista:  139947160569280
Elementi lista Numeri, dopo la cancellazione
 della testa:  [2, 3]
```

Parametri di tipo lista: passare una lista come INPUT di una funzione (4)

- Quando una funzione ritorna una lista in realtà viene ritornato un **riferimento (collegamento con l'identificativo)** alla lista stessa. Per esempio *Coda* ritorna una lista che contiene tutti gli elementi di una lista a parte il primo:

```
def Coda(Lista):
    return Lista[1:]


Numeri = [1, 2, 3]
Resto = Coda(Numeri)
print(Resto)
```

```
In [7]: def Coda(Lista):
   ...:     return Lista[1:]
   ...:
   ...:
   ...: Numeri = [1, 2, 3]
   ...: Resto = Coda(Numeri)
   ...: print(Resto)
[2, 3]
```

- Il valore ritornato è stato creato con l'operatore *porzione*, quindi la funzione *Coda* restituisce una nuova lista
- La creazione di Resto ed ogni suo successivo cambiamento non ha alcun effetto sulla lista originale Numeri

Operatore Slicing: varie azioni sulle liste

```
t = ['a','b','c','d','e','f']
print(t[1:4])
a = t[:] #slicing per effettuare la copia di una
lista
print(a)
t[1:4] = ['ecco','sovrascrivo','adesso']
print(t)
z = ['s','r','y']
t[1:4] = z #slicing per MODIFICARE
print(t)
t[1:4] = ['m', 'n'] #slicing per CANCELLARE
(modifico il numero di elementi)
print(t)
```

```
['b', 'c', 'd']
['a', 'b', 'c', 'd', 'e', 'f']
['a', 'ecco', 'sovrascrivo', 'adesso', 'e', 'f']
['a', 's', 'r', 'y', 'e', 'f']
['a', 'm', 'n', 'e', 'f']
>
```

Liste annidate

- Una lista annidata è una lista che compare come elemento di un'altra lista
- Nell'esempio seguente il quarto elemento (elemento con indice 3) della lista è una lista:

```
#liste annidate
Lista = ["ciao", 2.0, 5, [10, 20]]
i = 0
print("Visita Lista:")
while i < 4:
    print(Lista[i])
    i = i + 1
#estrarre la lista annidata - metodo 1
print("Estrazione:")
Elemento = Lista[3]
print(Elemento)
print(Elemento[0])
#estrarre la lista annidata - metodo 2
print(Lista[3][0])
print(Lista[3][1])
```

```
Visita Lista:
ciao
2.0
5
[10, 20]
Estrazione:
[10, 20]
10
10
20
>
```

```
Elementi di Lista:
ciao
2.0
5
[10, 20]
Estrazione:
Stampo elemento con indice 3 della lista Lista e lo chiamo Elemento (nu
ova lista):  [10, 20]
Stampo elemento con indice 0 della Lista Elemento:  10
Accedo direttamente ai sotto elementi partendo dalla lista Lista:
Lista[3][0] vale:  10
Lista[3][1] vale:  20
```

```python
Lista = ["ciao", 2.0, 5, [10, 20]]
i = 0
print("Elementi di Lista:")
while i < 4:
    print(Lista[i])
    i = i + 1
#estrarre lista annidata-metodo 1
print("Estrazione:")
Elemento = Lista[3]
print('Stampo elemento con indice 3 della lista Lista e lo chiamo Elemento (nuova
lista): ', Elemento)
print('Stampo elemento con indice 0 della Lista Elemento: ',Elemento[0])
#estrarre la lista annidata - metodo 2
print('Accedo direttamente ai sotto elementi partendo dalla lista Lista:')
print('Lista[3][0] vale: ',Lista[3][0])
print('Lista[3][1] vale: ',Lista[3][1])
```

Matrici

- Le liste annidate sono spesso usate per rappresentare matrici. Per esempio la matrice:

```
#Matrice
Matrice = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
#estrazione della singola cella
print(Matrice[2][1])
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Visita di Matrici – ciclo for

```
Matrice = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
#stampa per righe
print('Stampa per righe')
  for i in Matrice:
     print(i)


 #stampa elemento per elemento
print('Stampa elemento per elemento righe')
  for i in Matrice:
    for j in i:
       print(j)
```

```
Stampa per righe
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
Stampa elemento per elemento
1
2
3
4
5
6
7
8
9
```

Visita di Matrici – ciclo for

```
Matrice = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print('Stampa per righe') #stampa per righe
i=0
while i<3:
    print(Matrice[i])
    i=i+1
#stampa elemento per elemento
print('Stampa elemento per elemento')

i=0
while i<3:
    j=0
    while j<3:
        print(Matrice[i][j])
        j= j+1
    i=i+1
```

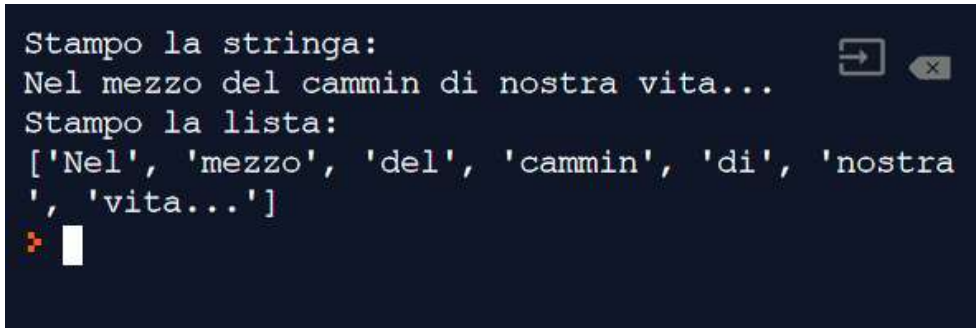
```
Stampa per righe
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
Stampa elemento per elemento
1
2
3
4
5
6
7
8
9
```

Visita di Matrici – uso di len(M)

Stringhe e Liste

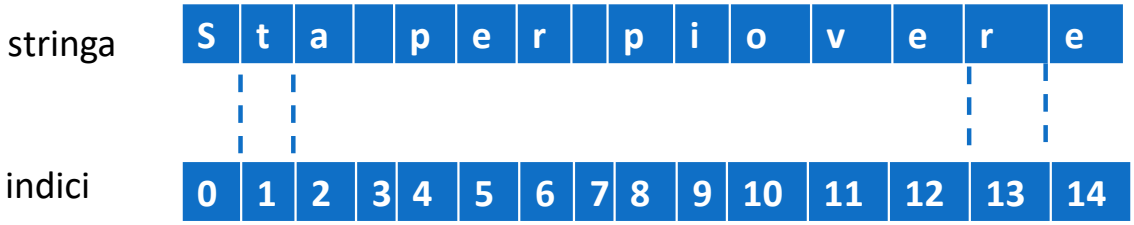
- Due delle funzioni più utili nel modulo string hanno a che fare con le liste di stringhe
- La funzione *split* spezza una stringa in **una lista di parole singole**, considerando un qualsiasi carattere di spazio bianco come punto di interruzione tra parole consecutive:

```
#stringhe e liste
import string
Verso = "Nel mezzo del cammin di nostra vita..."
lista_versi = Verso.split()
print("Stampo la stinga:")
print(Verso)
print("Stampo la lista:")
print(lista_versi)
```



Metodi per la gestione delle stringhe (1)

- Nella programmazione, un **oggetto** è una entità software che rappresenta un valore avente un determinato comportamento
- Tale valore può essere semplice, come una stringa oppure complesso (concetto che vedremo in futuro)
- Il comportamento di un oggetto è definito dai suoi **metodi**
- I metodi sono funzioni, ovvero sequenze di istruzioni che portano a termine un determinato compito
- Però, diversamente da una funzione a se' stante, i metodi possono essere applicati solo agli oggetti per i quali sono stati definiti



Metodi per la gestione delle stringhe (2)

- Ad esempio il metodo upper può essere applicato a qualunque stringa nel modo seguente:

name = ‘Mario Rossi’

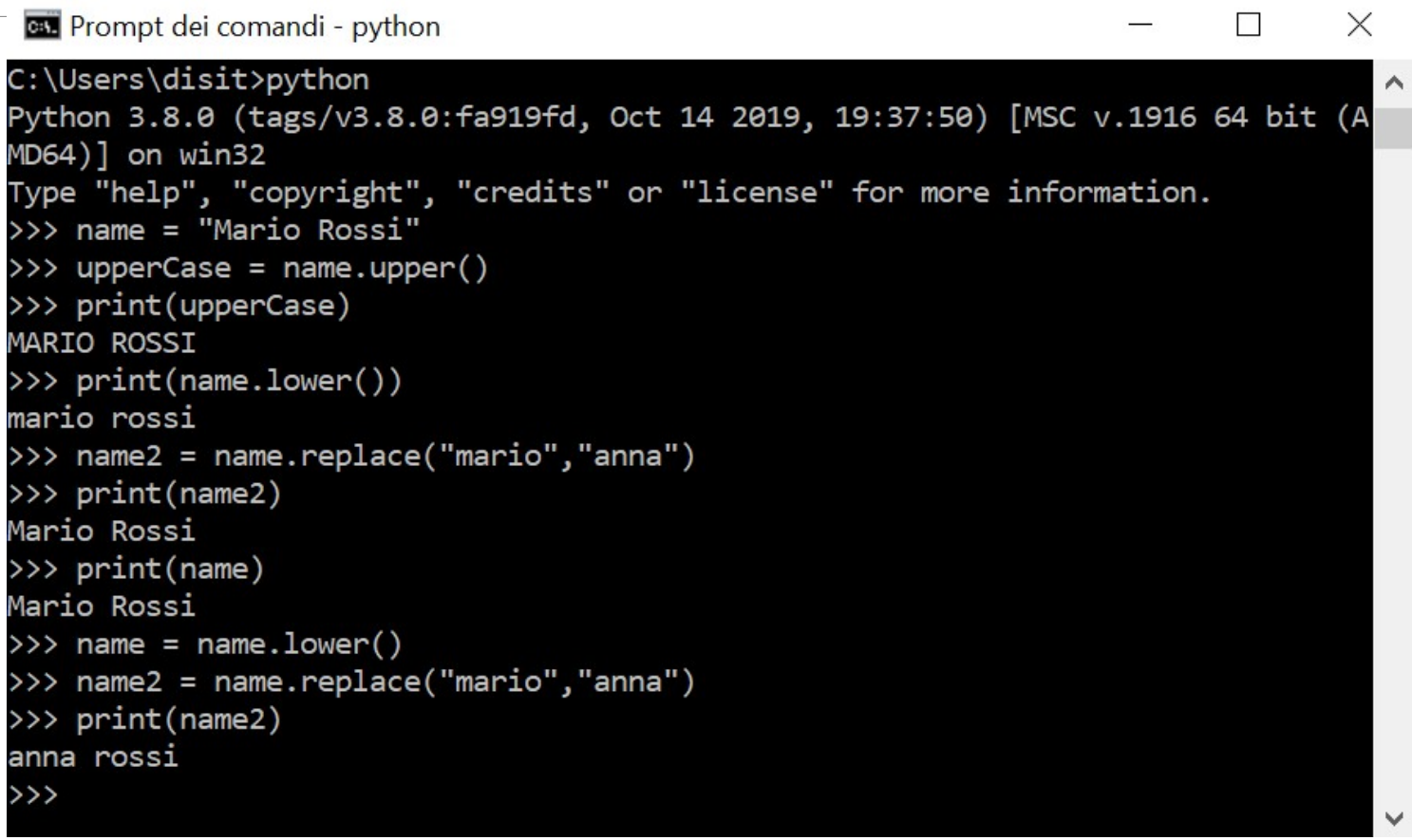
upperCase = name.upper() #assegna il valore ‘MARIO ROSSI’ alla variabile upperCase

Altri metodi:

lower()

replace()

ATTENZIONE alla differenza tra stampa e assegnazione!!!



Metodi per la gestione delle stringhe (3)

Ulteriori metodi per gestire le stringhe:

https://docs.python.it/html/lib/string-methods.html

`capitalize`()
Restituisce una copia della stringa con il solo carattere iniziale maiuscolo.

`center`(*width*[, *fillchar*])
Restituisce la centratura di una stringa, in un campo di ampiezza *width*. Il riempimento viene fatto usando il *fillchar* specificato (il predefinito è uno spazio). Modificato nella versione 2.4: Supporto per l'argomento *fillchar*.

`count`(*sub*[, *start*[, *end*]])
Restituisce il numero di occorrenze della sotto stringa *sub* nella stringa S`[`*start*`:`*end*`]`. Gli argomenti facoltativi *start* e *end* vengono interpretati come nella notazione delle fette.

`decode`([*encoding*[, *errors*]])
Decodifica la stringa usando il codec registrato per *encoding* (NdT: codifica). *encoding* è la modalità predefinita per la codifica delle stringhe. *errors* può essere impostato con un differente schema di gestione. Il predefinito è `'strict'`, ad indicare che errori di codifica solleveranno un'eccezione di tipo `ValueError`. Altri possibili valori sono `'ignore'` e `'replace'`. Nuovo nella versione 2.2.

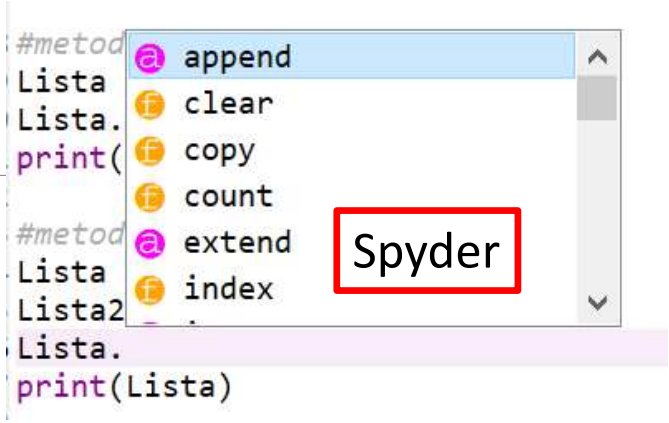
`encode`([*encoding*[,*errors*]])
Restituisce una versione codificata della stringa. La codifica predefinita è quella predefinita per la stringa corrente. *errors* può essere impostato con un differente schema di gestione. Il predefinito è `'strict'`, ad indicare che errori di codifica solleveranno un'eccezione di tipo `ValueError`. Altri possibili valori sono `'ignore'` e `'replace'`. Nuovo nella versione 2.0.

`endswith`(*suffix*[, *start*[, *end*]])
Restituisce `True` se la stringa finisce con lo specificato suffisso *suffix*, altrimenti restituisce `False`. Con il facoltativo *start*, il test inizia da quella posizione. Con il facoltativo *end*, blocca il confronto a quella posizione.

`expandtabs`([*tabsize*])
Restituisce una copia della stringa dove tutti i caratteri tab vengono espansi usando gli spazi. Se *tabsize* non viene fornito, si assume che sia di `8` caratteri.

NOTA – Ambiente di sviluppo

- Quando si scrive codice Python, l'ambiente di sviluppo usato ci da' una serie di suggerimenti
- Ad esempio se usiamo le liste, suggerisce la lista di metodi disponibili per la gestione di tali liste
- Ogni ambiente di sviluppo avrà il suo modo diverso di visualizzazione, ovviamente le informazioni sono equivalenti



INSIEMI E DIZIONARI

Insiemi

Insiemi (1)

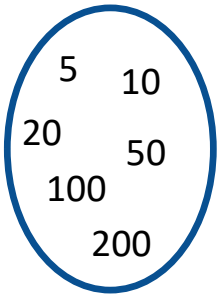
- Quando in un programma, c'è bisogno di organizzare la memorizzazione di più valori, questi possono essere inseriti in un contenitore
- Il contenitore Lista che abbiamo già visto è solo uno dei contenitori possibili messi a disposizione in Python
- Vediamo ora Gli Insiemi e i Dizionari
- Un insieme è un contenitore che memorizza una raccolta di valori univoci
- Diversamente da una lista, gli elementi o membri di un insieme non vengono memorizzati in alcun ordine specifico e NON vi si può accedere in base alla loro posizione (ovvero non si utilizzano gli indici)
- Le operazioni che si possono applicare sugli insiemi, sono le stesse che si possono effettuare sugli insiemi matematici

Insiemi (2)

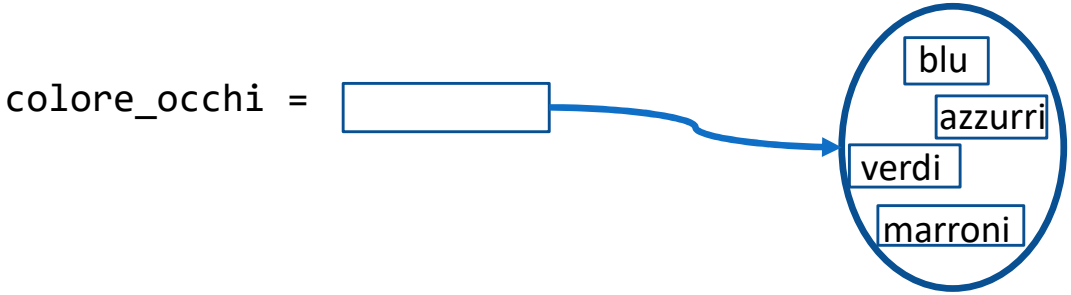
- Negli esempi di insiemi a fianco, si noti che non ci sono duplicati
- Per creare un insieme contenente alcuni elementi iniziali, basta specificare tali elementi dentro una parentesi graffa:

Possibili colori degli occhi

Banconote serie Europa (dal 2019)



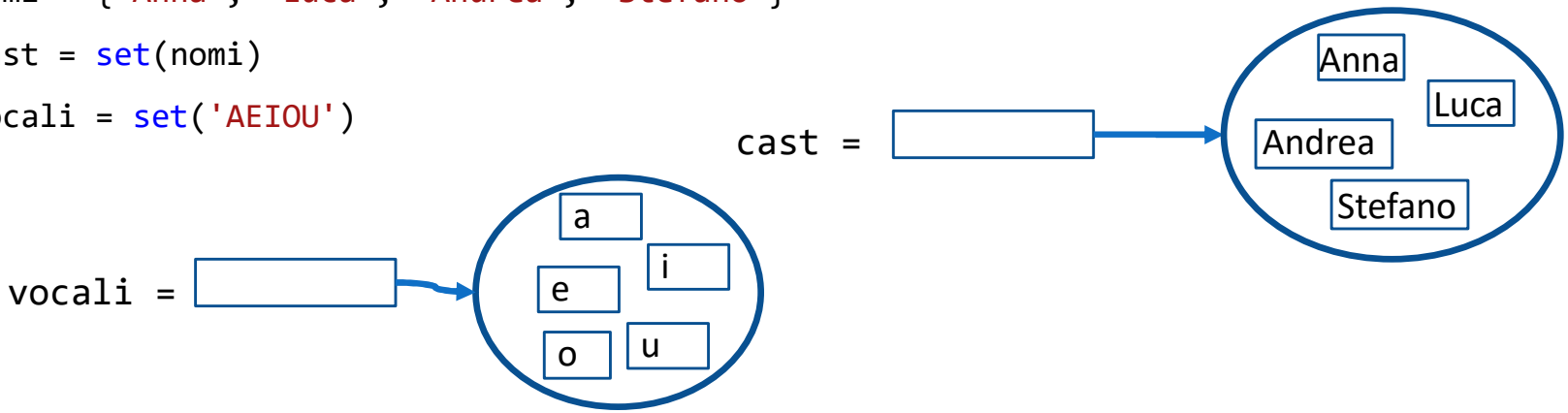

```
colore_occhi ={'verdi', "marroni", 'blu', 'neri'}
```



Insiemi (3)

- Una modalità alternativa per creare un insieme è quella di usare la funzione set, che converte in un insieme qualsiasi sequenza

```
nomi = {'Anna', 'Luca', 'Andrea', 'Stefano'}
cast = set(nomi)
vocali = set('AEIOU')
```



- In Pyhton NON si può creare un insieme vuoto con due parentesi graffe che non racchiudono nulla ({}), si ricorre allora alla funzione set:

```
insieme_vuoto = set()
```

Insiemi (4)

- Come con ogni altro contenitore, è possibile usare la funzione len() per conoscere il numero degli elementi che costituiscono un insieme

```
nomi = {'Anna', 'Luca', 'Andrea', 'Stefano'}
cast = set(nomi)
numero_persone_cast = len(cast)
print(numero_persone_cast)
```

- Per verificare che gli elementi appartengano (o NON) ad un insieme si usa l'operatore in (not in):

```
if 'Anna' in cast:
    print('Anna fa parte del cast di questo spettacolo')
else:
    print('Anna NON fa parte del cast di questo spettacolo')
```

Insiemi (5)

- Gli elementi che fanno parte degli insiemi NON sono ordinati in alcun modo, per questo NON è possibile accedere al singolo elemento tramite indice
- Per scandire gli elementi è necessario farlo con un ciclo for:

```
nomi = {'Anna', 'Luca', 'Andrea', 'Stefano'}

cast = set(nomi)

print('Gli attori che fanno parte del cast di questo spettacolo sono:')

for attore in cast:

    print(attore)
```

```
Gli attori che fanno parte del cast di questo spettacolo sono:
Luca
Stefano
Andrea
Anna
```

- Si noti che l'ordine di visita degli elementi in un insieme, dipende da come questi sono stati memorizzati e NON da come sono stati immessi nel programma per creare l'insieme

Insiemi (6)

- Quando si lavora con gli insiemi, il fatto che siano costituiti da elementi non ordinati NON è un problema, anzi questa caratteristica rende le operazioni tra insiemi, molto più efficienti
- Solitamente vogliamo visualizzare gli elementi in ordine, si usa allora la funzione sorted().
- sorted() restituisce una lista contenente gli stessi elementi dell'insieme MA ordinati

```
nomi = {'Anna', 'Luca', 'Andrea', 'Stefano'}

cast = set(nomi)

print('Gli attori che fanno parte del cast di questo spettacolo sono:')

for attore in cast:

    print(attore)

print('Gli attori, in ordine alfabetico, che fanno parte del cast di questo spettacolo sono:')

for attore in sorted(cast):

    print(attore)
```

```
Gli attori che fanno parte del cast di questo spettacolo sono:
Luca
Stefano
Andrea
Anna
Gli attori, in ordine alfabetico, che fanno parte del cast di questo spettacolo sono:
Andrea
Anna
Luca
Stefano
```

Insiemi: aggiungere e rimuovere elementi (1)

- Gli insiemi sono raccolte di elementi Modificabili, come le Liste
- Si possono allora aggiungere o rimuovere elementi da un insieme
- Per aggiungere elementi si usa add()

```
nomi = {'Anna', 'Luca', 'Andrea', 'Stefano'}
cast = set(nomi)
print('Gli attori che facevano parte a Marzo del cast erano:')
for attore in cast:
    print(attore)

cast.add('Chiara')
print('Gli attori definitivi e in
          ordine alfabetico del cast sono:')
for attore in sorted(cast):
    print(attore)
```

```
Gli attori che facevano parte a Marzo del cast erano:
Luca
Andrea
Anna
Stefano
Gli attori definitivi e in ordine alfabetico del cast sono:
Andrea
Anna
Chiara
Luca
Stefano
```

Insiemi: aggiungere e rimuovere elementi (2)

- Per eliminare un singolo elemento da un insieme, esistono due modalità: discard(), remove() e clear()
- Il metodo discard() elimina un elemento, se questo esiste all'interno di un insieme e NON ha alcun effetto se si tenta di rimuovere un elemento che non fa parte dell'insieme
- Il metodo remove() invece elimina un elemento se questo appartiene all'insieme, altrimenti solleva una eccezione (messaggio di errore)
- Il metodo clear() infine elemina tutti gli elementi da un insieme, lasciandolo vuoto

Insiemi: aggiungere e rimuovere elementi (3)

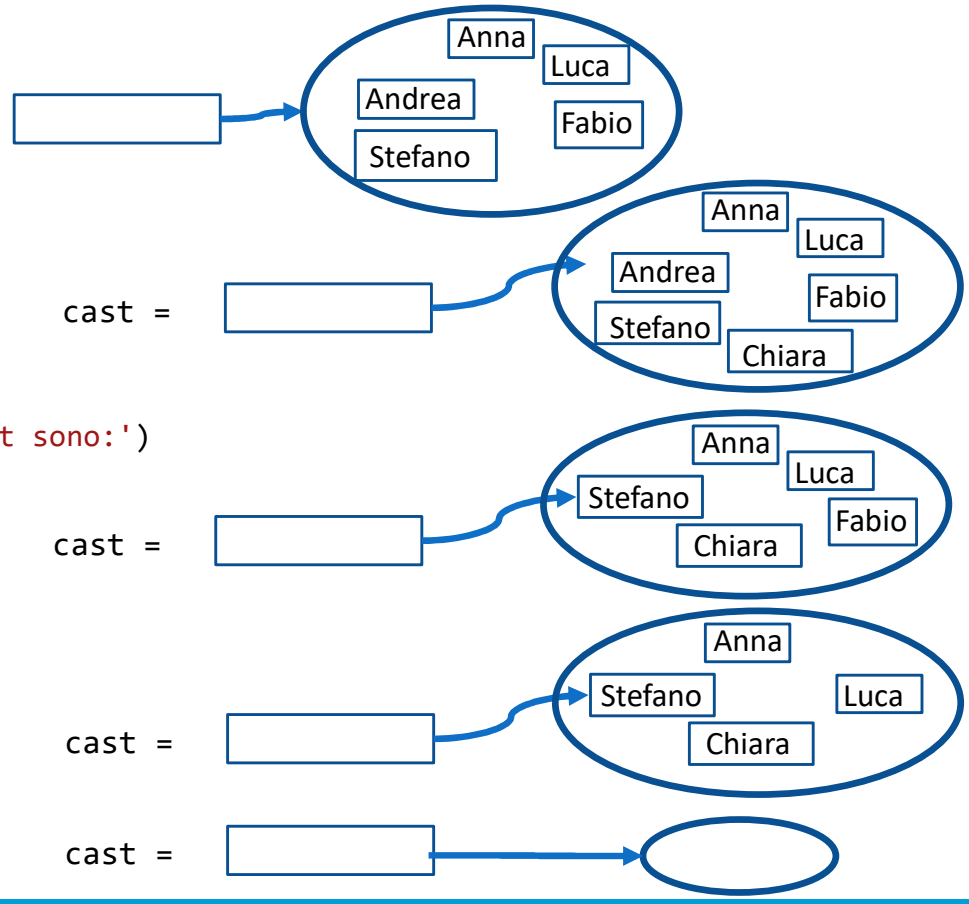
```
nomi = {'Anna', 'Luca', 'Andrea', 'Stefano', 'Fabio'}
cast = set(nomi)
cast.add('Chiara')
cast.discard('luca')
cast.discard('Andrea')
#cast.remove('Massimo')
cast.remove('Fabio')
print('Gli attori definitivi e in ordine alfabetico del cast sono:')
for attore in sorted(cast):
    print(attore)
print('Spettacolo annullato, niente cast!')
cast.clear()
for attore in sorted(cast):
    print(attore)
```

```
Gli attori definitivi e in ordine alfabetico del cast sono:
Anna
Chiara
Luca
Stefano
Spettacolo annullato, niente cast!
```

```
Traceback (most recent call last):
  File "main.py", line 7, in <module>
    cast.remove('Massimo')
KeyError: 'Massimo'
```

Insiemi: aggiungere e rimuovere elementi (3)

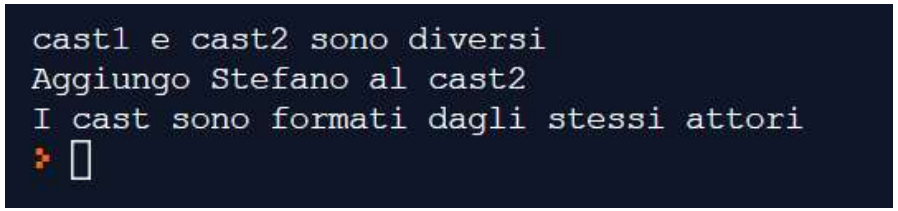
```
nomi = {'Anna', 'Luca', 'Andrea', 'Stefano', 'Fabio'}
cast = set(nomi)
cast.add('Chiara')
cast.discard('luca')
cast.discard('Andrea')
#cast.remove('Massimo')
cast.remove('Fabio')
print('Gli attori definitivi e in ordine alfabetico del cast sono:')
for attore in sorted(cast):
    print(attore)
print('Spettacolo annullato, niente cast!')
cast.clear()
for attore in sorted(cast):
    print(attore)
```



Sottoinsiemi (1)

- Un insieme è un sottoinsieme di un altro insieme solo se tutti gli elementi del primo insieme sono anche elementi del secondo insieme
- Il metodo issubset() restituisce True o False per segnalare se un insieme è un sottoinsieme di un altro
- Si può anche verificare se due insiemi sono uguali (diversi), ovvero se hanno gli stessi elementi. In questo caso si usa l'operatore Uguaglianza '==' (Disuguaglianza '!=')

```
nomi = {'Anna', 'Luca', 'Andrea', 'Stefano'}
cast1 = set(nomi)
cast2 = set(['Anna', 'Luca', 'Andrea'])
if cast1==cast2:
   print('I cast sono formati dagli tessi attori')
else:
   print('cast1 e cast2 sono diversi')
print('Aggiungo Stefano al cast2')
cast2.add('Stefano')
if cast1==cast2:
    print('I cast sono formati dagli stessi attori')
else:
    print('cast1 e cast2 sono diversi')
```



Unione, Intersezione e differenza tra Insiemi (1)

- L'unione tra due insiemi contiene tutti gli elementi che provengono da entrambi gli insiemi, dopo aver eliminato i duplicati
- Il metodo union() genera l'unione di due insiemi

```
nomi = {'Anna', 'Luca', 'Stefano'}

cast1 = set(nomi)

cast2 = set(['Anna', 'Luca', 'Andrea'])

cast = cast1.union(cast2)

print('Ecco tutti gli attori')

for attore in sorted(cast):

    print(attore)
```

```
Ecco tutti gli attori
Andrea
Anna
Luca
Stefano
>
```

Unione, Intersezione e differenza tra Insiemi (2)

- L'intersezione di due insiemi contiene tutti gli elementi che appartengono ad entrambi gli insiemi. Metodo intersection()
- La differenza tra due insiemi contiene tutti gli elementi che appartengono al primo insieme ma non al secondo. Metodo difference()

```
nomi = {'Anna', 'Luca', 'Stefano'}
cast1 = set(nomi)
cast2 = set(['Anna', 'Luca', 'Andrea', 'Simona'])
cast_double = cast1.intersection(cast2)
cast_only1 = cast1.difference(cast2)
cast_only2 = cast2.difference(cast1)
print('Ecco gli attori che fanno parte di due cast:')
for attore in sorted(cast_double):
    print(attore)
print('Ecco gli attori che fanno parte del cast1 ma non del cast2:')
for attore in sorted(cast_only1):
   print(attore)
print('Ecco gli attori che fanno parte del cast2 ma non del cast1:')
for attore in sorted(cast_only2):
    print(attore)
```

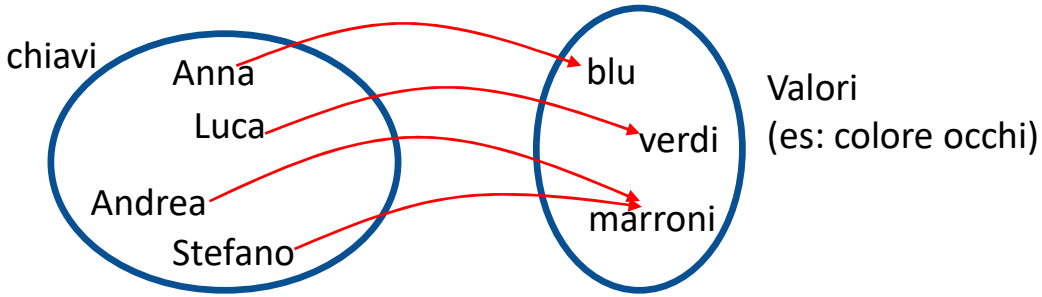
```
Ecco gli attori che fanno parte di due cast:
Anna
Luca
Ecco gli attori che fanno parte del cast1 ma non del cast2:
Stefano
Ecco gli attori che fanno parte del cast2 ma non del cast1:
Andrea
Simona
```

- Si noti che quando si genera l'unione o l'intersezione di due insiemi, l'ordine in cui li si usa è indifferente. Non si puo' dire lo stesso della operazione differenza

DIZIONARI

Dizionari (1)

- Un Dizionario è simile ad una Lista, ma è più generico
- Nel caso della Lista gli indici devono essere numeri interi mentre per i Dizionari possono essere (quasi) di ogni tipo
- Un Dizionario contiene una raccolta di indici detti **chiavi** e una raccolta di **valori**
- Ciascuna chiave è associata ad un unico valore
- L'associazione tra un valore e la chiave è detta **coppia chiave-valore** o anche **elemento**
- «Un dizionario rappresenta una relazione di corrispondenza, o mappatura, da una chiave a un valore, e si può dire pertanto che ogni chiave 'mappa in un valore'»



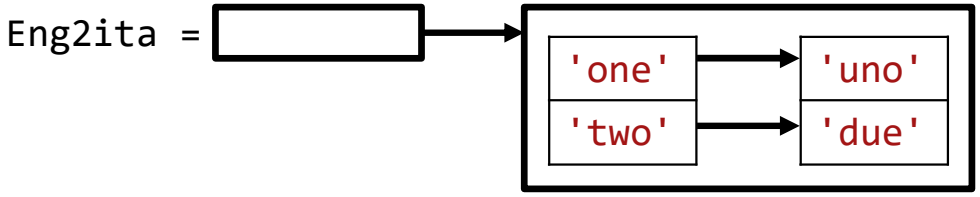
Valori
(es: colore occhi)

- NOTA: la struttura che è stata chiamata Dizionario, è anche nota come **mappa**

Dizionari (2)

- Supponiamo di voler creare un dizionario per la traduzione di parole dall'inglese all'italiano
- In questo caso è utile poter usare la parola inglese come indice di ricerca della corrispondente italiana e quindi gli indici usati sono delle stringhe
- Un modo per creare un dizionario è partire con un dizionario vuoto e aggiungere via via gli elementi. Il dizionario vuoto indicato da {}
- Poi si aggiungono gli elementi

```
#Dizionario inglese-italiano
eng2ita = {}
eng2ita['one'] = 'uno'
eng2ita['two'] = 'due'
print(eng2ita)
```



- Quando si stampa i Dizionario, si vedono le coppie chiave-valore
- In questo caso in numeri in inglese sono le chiavi e quelli in italiano i valori

```
{'one': 'uno', 'two': 'due'}
```

Dizionari (3)

- Il formato di output appena visto può essere anche usato come metodo alternativo di inserimento degli elementi in un Dizionario:

```
eng2ita = {}
eng2ita = {'one': 'uno', 'two': 'due', 'three': 'tre'}
print('Stampo tutto il Dizionario:')
print(eng2ita)
print('Stampo solo la traduzione in italiano di "two":')
print(eng2ita['two'])
```

```
Stampo tutto il Dizionario:
{'one': 'uno', 'two': 'due', 'three': 'tre'}
Stampo solo la traduzione in italiano di "two":
due
```

Dizionari: sintassi

- Gli elementi di una coppia chiave/valore sono separati da un carattere ':' mentre l'elenco di coppie è racchiuso tra graffe
- Una coppia di parentesi graffe senza elementi, per convenzione indica il dizionario vuoto

Esempio:

```
favoriteColors = {"Anna":"Red","Simone":"Blue" }
emptyDict = {} #Dizionario vuoto
oldColors = dict(favoriteColors) #copia

print(favoriteColors["Anna"])
print(oldColors["Anna"])
```

L'operatore di indicizzazione, [], si usa anche per accedere al valore associato ad una chiave in un dizionario

Nome del Dizionario

main.py saved

```
1  favoriteColors = {"Anna":"Red","Simone":"Blue" }
2  emptyDict = {} #Dizionario vuoto
3  oldColors = dict(favoriteColors) #copia
4  print(favoriteColors["Anna"])
5  print(oldColors["Anna"])
```

```
Red
Red
```

Dizionari (4)

- Si noti che il Dizionario non è un contenitore di tipo sequenza, come la lista
- Anche se con i Dizionari si usa l'operatori di indicizzazione, non si può accedere agli elementi di un dizionario tramite indice o in base alla loro posizione
- Si può accedere ad un valore di un dizionario solo tramite la sua chiave
- La chiave per accedere al valore deve essere una chiave valida, altrimenti verrà sollevata una eccezione di tipo 'KeyError'

main.py saved

```
1   favoriteColors = {"Anna":"Red","Simone":"Blue" }
2   emptyDict = {} #Dizionario vuoto
3   oldColors = dict(favoriteColors) #copia
4   print(favoriteColors["Anna"])
5   print(oldColors["Chiara"])
```

```
Red
Traceback (most recent call last):
  File "main.py", line 5, in <module>
    print(oldColors["Chiara"])
KeyError: 'Chiara'
>
```

Operazione sui Dizionari: verifica di esistenza

- Per determinare se una chiave è presente in un dizionario, si usa l'operatore 'in', oppure 'not in'

```
eng2ita = {}
eng2ita = {'one': 'uno', 'two': 'due', 'three': 'quattro'}
print('Stampa del Dizionario:')
print(eng2ita)
if 'one' in eng2ita:
  print('Traduzione richiesta: ')
  print(eng2ita['one'])
else:
  print('Non esiste traduzione!')
#oppure
if 'one' not in eng2ita:
  print('Non esiste traduzione!')
else:
  print('Traduzione richiesta: ‘)
  print(eng2ita['one'])
```

```
Stampa del Dizionario:
{'one': 'uno', 'two': 'due', 'three': 'quattro'}
Traduzione richiesta:
uno
Traduzione richiesta:
uno
>
```

Operazione sui Dizionari: modifica/aggiunta di un elemento

- Un dizionario è un contenitore modificabile, ovvero se ne può cambiare il contenuto dopo che è stato creato
- Usando l'operatore di indicizzazione si possono modificare o aggiungere coppie

```
eng2ita = {}
eng2ita = {'one': 'uno', 'two': 'due', 'three': 'quattro'}
print('Stampa prima della modifica:')
print(eng2ita)
eng2ita['three'] = 'tre'#modifica
eng2ita['four'] = 'quattro'#aggiunta
print('Stampa dopo la modifica:')
print(eng2ita)
dimensione = len(eng2ita)
print('Stampo il numero di elementi: ', dimensione)
```

```
Stampa prima della modifica:
{'one': 'uno', 'two': 'due', 'three': 'quattro'}
Stampa dopo la modifica:
{'one': 'uno', 'two': 'due', 'three': 'tre', 'four': 'quattro'}
Stampo il numero di elementi:  4
```

NOTA: len(eng2ita), rende il numero di elementi del Dizionario

Operazioni sui Dizionari: rimuovere una coppia chiave-valore (del)

```
eng2ita = {}
eng2ita = {'one': 'uno', 'two': 'due', 'three': 'tre'}
print('Stampa prima della rimozione:')
print(eng2ita)
del eng2ita['two']
print('Stampa dopo la rimozione:')
print(eng2ita)
```

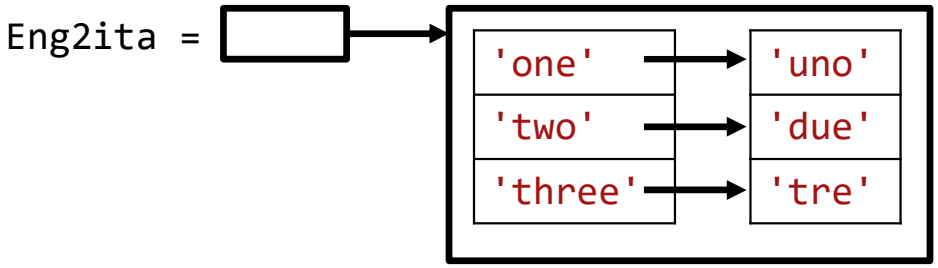
```
Stampa prima della rimozione:
{'one': 'uno', 'two': 'due', 'three': 'tre'}
Stampa dopo la rimozione:
{'one': 'uno', 'three': 'tre'}
```

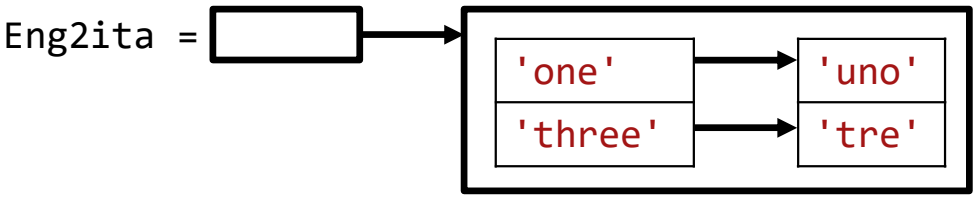
Operazioni sui Dizionari: rimuovere una coppia chiave-valore (pop)

- Per eliminare una coppia chiave/valore da un dizionario è anche possibile invocare il metodo pop, fornendo la chiave come argomento

```
eng2ita = {}
eng2ita = {'one': 'uno', 'two': 'due', 'three': 'tre'}

eng2ita.pop('two')
if 'two' not in eng2ita:
    print('Non esiste traduzione!')
```

```
Non esiste traduzione!
>
```





Metodi dei Dizionari: key, values

Un metodo è simile ad una funzione, visto che prende parametri e ritorna valori, ma la sintassi di chiamata è diversa

Il metodo *keys* prende un dizionario e ritorna la lista delle sue chiavi: invece di invocarlo con la sintassi delle funzioni keys(eng2ita) si usa la sintassi dei metodi eng2ita.keys()

Il metodo *values* prende un dizionario e ritorna la lista dei suoi valori

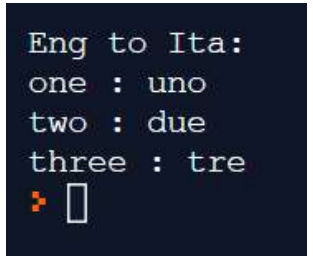
```
#Metodi: keys
eng2ita = {}
eng2ita = {'one': 'uno', 'two': 'due', 'three': 'quattro'}
print(eng2ita.keys())
print(eng2ita.values())
```

```
dict_keys(['one', 'two', 'three'])
dict_values(['uno', 'due', 'quattro'])
>>>
```

Operazioni sui Dizionari: scandire gli elementi

- Usando un ciclo for è possibile scandire le chiavi (e i valori) di un dizionario

```
eng2ita = {}
eng2ita = {'one': 'uno', 'two': 'due', 'three': 'tre'}
print('Eng to Ita:')
for key in eng2ita:
    print(key, ':', eng2ita[key])
```



Cicli e Dizionari

```
Stampo il dizionario:
a 20
r 6
d 9
s 6
f 2
Stampo il dizionario in ordine di chiave
a 20
d 9
f 2
r 6
s 6
```

```
#Metodi: cicli e dizionari
def stampa_dizionario(d):
  for c in d:
     print(c, d[c])

def stampa_dizionario_ordinato_per_chiavi(d):
    for c in sorted(d): #uso della funzione pre
       print(c, d[c])

dizionario = {'a': '20', 'r': '6', 'd': '9','s': '6', 'f': '2'}
print("Stampo il dizionario:")
stampa_dizionario(dizionario)
print("Stampo il dizionario in ordine di chiave")
stampa_dizionario_ordinato_per_chiavi(dizionario)
```

Metodi dei Dizionari: items

- Il metodo items ritorna entrambi nella forma di una lista di tuple (concetto che vedremo), una per ogni coppia chiave-valore:

```
#Metodi: items e has_key
eng2ita = {}
eng2ita = {'one': 'uno', 'two': 'due', 'three': 'quattro'}
print(eng2ita.items())
```

```
dict_items([('one', 'uno'), ('two', 'due'), ('three', 'quattro')])
```

Alias e copia

- I dizionari sono Mutabili, quindi è necessario fare attenzione agli Alias
- Se si vuole poter modificare un dizionario e mantenere una copia dell'originale, è necessario usare il metodo *copy*

```
opposti = {'alto': 'basso', 'giusto': 'sbagliato', 'vero': 'falso'}
alias = opposti
copia = opposti.copy()
print('Stampo il dizionario opposti:')
print(opposti)
copia['alto'] = 'tall'
print('Stampo il dizionario opposti,
       dopo aver modificato la sua copia:')
print(opposti)
alias['alto'] = 'tall'
print('Stampo il dizionario opposti,
       dopo aver modificato il suo alias:')
print(opposti)
```

```
Stampo il dizionario opposti:
{'alto': 'basso', 'giusto': 'sbagliato', 'vero': 'falso'}
Stampo il dizionario opposti,
 dopo aver modificato la sua copia:
{'alto': 'basso', 'giusto': 'sbagliato', 'vero': 'falso'}
Stampo il dizionario opposti,
 dopo aver modificato il suo alias:
{'alto': 'tall', 'giusto': 'sbagliato', 'vero': 'falso'}
>
```

- Se si effettua una modifica su copia, allora tale modifica NON si ripercuote anche su opposti
- Al contrario, se si effettua una modifica su alias, si modifica anche il dizionario iniziale perché si lavora con i puntatori

Matrici sparse (1)

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \end{bmatrix}$$

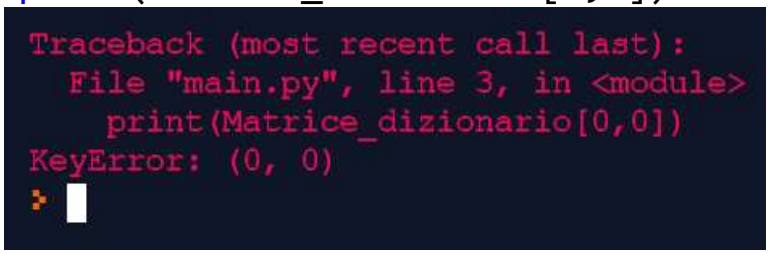
```
#metodo classico per definire una matrice
Matrice = [ [0,0,0,1,0], [0,0,0,0,0], [0,2,0,0,0], [0,0,0,0,0], [0,0,0,3,0] ]
print("Stampa elemento con indice di riga 2 e di colonna 1:")
print(Matrice[2][1])
#definizione di una matrice con un dizionario
Matrice_dizionario = {(0,3): 1, (2, 1): 2, (4, 3): 3}
print("Stampa elemento con indice di riga 2 e di colonna 1, partendo da un
dizionario:")
print(Matrice_dizionario[2,1])
```

```
Stampa elemento con indice di riga 2 e di colonna 1:
2
Stampa elemento con indice di riga 2 e di colonna 1,
 partendo da un dizionario:
2
>
```

Matrici Sparse (2)

- Usando un dizionario per definire una matrice sparsa, se si cerca un elemento nullo, si in corre in un errore:

```
Matrice_dizionario = {(0,3): 1, (2, 1): 2, (4, 3): 3}
print(Matrice_dizionario[0,0])
```



- Per ovviare al problema, si usa il metodo get:

```
Matrice_dizionario = {(0,3): 1, (2, 1): 2, (4, 3): 3}
print(Matrice_dizionario.get((0,3), 0))
print(Matrice_dizionario.get((0,0), 0))
```

- Il primo argomento di get serve per individuare righe e colonne della matrice, il secondo per assegnare il valore di default nel caso non ce ne sia uno già assegnato

Tuple

Tuple (1)

- Una Tupla è una sequenza di valori
- I valori possono essere di qualsiasi tipo
- Sono indicizzati tramite numeri interi (in questo somigliano alle liste)
- **Le tuple**, contrariamente alle Liste, **sono immutabili**
- Sintatticamente, una tupla è un elenco di valori separati da virgole
- A differenza delle liste, se cerchiamo di modificare gli elementi di una tupla, otteniamo un messaggio d'errore

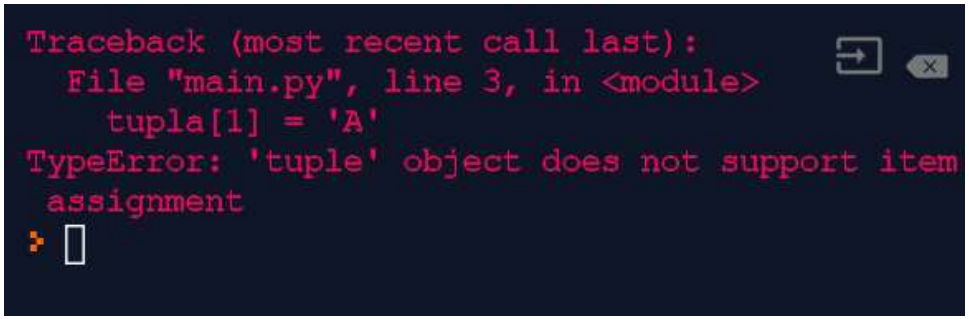
Tuple (2)

```
tupla = 'a','b','c','d','e','f'
tupla_equivalente = 'a','b','c','d','e','f'
tupla_1elem = 'a',
stringa = 'a'
print('Tuple:')
print(tupla)
print(tupla_equivalente)
print('Tuple con 1 solo elemento')
print(tupla_1elem)
print('Stringa:')
print(stringa)
print('Singoli elementi:')
print(tupla[1])
#uso operatore porzione
print('Intervallo:')
print(tupla[2:4])
```

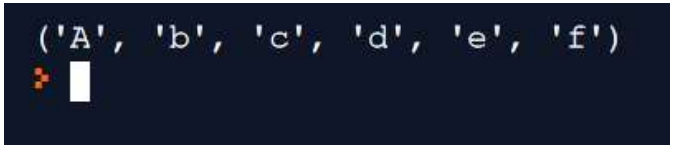
```
Tuple:
('a', 'b', 'c', 'd', 'e', 'f')
('a', 'b', 'c', 'd', 'e', 'f')
Tuple con 1 solo elemento
('a',)
Stringa:
a
Singoli elementi:
b
Intervallo:
('c', 'd')
```

Tuple (3)

```
tupla = 'a','b','c','d','e','f'
#tentativo di modificare un elemento di una tupla
tupla[1] = 'A’
```



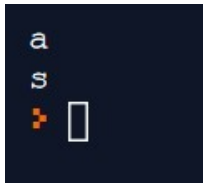

```
#alternativa alla modificа
tupla = ('A',) + tupla[1:]
print(tupla)
```



Assegnazione di Tupla

- Con le istruzioni di assegnazioni convenzionali, per **scambiare** i valori di due variabili, è necessario usare una variabile **temporanea**:

```
a ='s'
b = 'a'
temp = a
a = b
b = temp
print(a)
print(b)
```



- L'uso dell'**assegnazione di tupla** è più elegante:

```
a = 's'
b = 'a'
c = 'c'
a, b, c = b, a,'ciao'
print(a)
print(b)
print(c)
```

Tuple come valori di ritorno di una funzione (1)

- Le funzioni possono ritornare tuple
- Ad esempio una funzione che scambia due valori:

```
#funzione che scambia due valori
   def Scambia(x, y):
       return y, x

   a = 'Giorno '
   b = 'Buon '
   print(a)
   print(b)
   print('Valori iniziali:')
   print('valore di a:' + a)
   print('valore di b:' + b)
   a, b = Scambia(a, b)
   print('Valori scambiati:')
   print('valore di a:' + a)
   print('valore di b:' + b)
```

```
Giorno
Buon
Valori iniziali
valore di a: Giorno
valore di b: Buon
Valori scambiati:
valore di a: Buon
valore di b: Giorno
>
```

Tuple come valori di ritorno di una funzione (2)

- Le funzioni possono ritornare Tuple. Occorre però fare attenzione:

```
#funzione che scambia due valori - ERRATA
def Scambia(x, y):
    x, y = y, x

a = 'Giorno '
b = 'Buon '
print(a)
print(b)
print('Valori iniziali')
print('valore di a: ' + a)
print('valore di b: ' + b)
Scambia(a, b)
print('Valori scambiati:')
print('valore di a: ' + a)
print('valore di b: ' + b)
```

```
Giorno
Buon
Valori iniziali
valore di a: Giorno
valore di b: Buon
Valori scambiati:
valore di a: Giorno
valore di b: Buon
```

COSA succede:

I nuovi valori (lo scambio) vengono assegnati a x e y

Al termine della funzione, x e y vengono rimosse perché **variabili locali**, qualsiasi valore in esse contenuto viene irrimediabilmente perso

Manca il return!!

In questo caso, quando si chiama questa funzione non vengono passate le variabili a e b come argomenti, ma i loro valori

Numeri casuali (1)

- Spesso i programmi fanno la stessa cosa ogni volta che vengono eseguiti e sono detti per questo deterministici
- Di solito un programma deterministico è una cosa voluta in quanto: a parità di dati in ingresso, ci si attende lo stesso risultato
- Per alcune applicazioni, invece, è necessario che l'esecuzione sia imprevedibile
- Creare un programma realmente non deterministico (e quindi imprevedibile) è una cosa piuttosto difficile, ma ci sono dei sistemi per renderlo abbastanza casuale da soddisfare la maggior parte delle esigenze in tal senso

Numeri casuali (2)

- Uno dei sistemi è quello di generare dei numeri casuali ed usarli per determinare i risultati prodotti dal programma
- Python fornisce delle funzioni di base che generano numeri pseudocasuali
- Il modulo random contiene una funzione chiamata random che restituisce un numero in virgola mobile compreso tra 0.0 (compreso) e 1.0 (escluso)
- Ad ogni chiamata di random si ottiene il numero seguente di una lunga serie di numeri pseudocasuali

```
#numeri casuali: random
import random
for i in range(10):
    x = random.random()
    print(x)
```

```
0.7765590350610015
0.8090021598660779
0.8169223721604371
0.8119077162417103
0.6196094126001445
0.053878357921128495
0.5626756720474235
0.34299340811209955
0.32470831775684583
0.4995208053841198
```

Numeri casuali (3)

- Per generare un numero casuale compreso tra 0.0 (compreso) ed un limite superiore Limite (escluso) si moltiplica x per il Limite superiore:

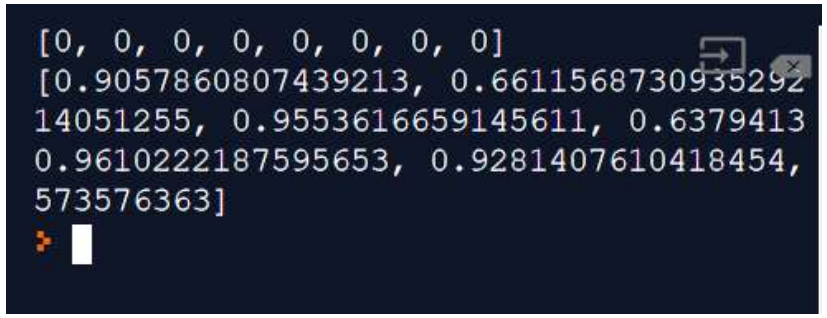
```
#numeri casuali: random
import random
min =0
for i in range(10):
    x = float(random.random()*3)
    print(x)
```

```
2.2353328574554 45
1.7585096950902674
2.9254953262338304
2.653751271499717
1.1398360333044613
2.467508190219747
1.9393791753716 76
0.4693562720699763
0.5302012725622125
0.5634766713337978
```

Lista di numeri casuali

- Proviamo a scrivere un programma che usa i numeri casuali, iniziando con la costruzione di una lista di questi numeri
- ListaCasuale prende un parametro intero Lungh e ritorna una lista di questa lunghezza composta di numeri casuali
- Iniziamo con una lista di Lungh zeri e sostituiamo in un ciclo un elemento alla volta con un numero casuale:

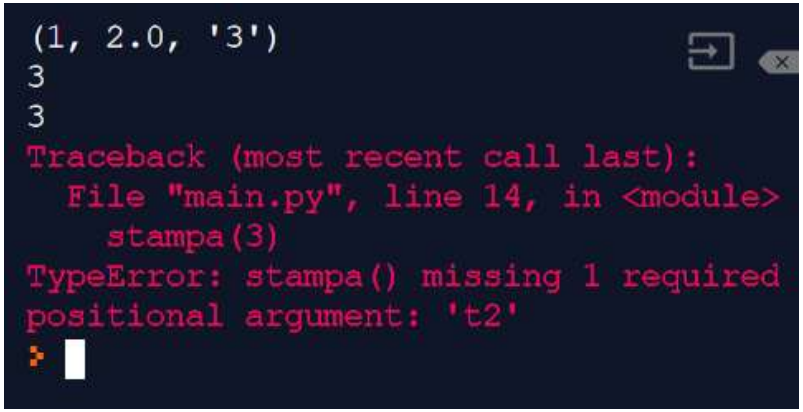
```
import random
def ListaCasuale(Lungh):
    #si inizializza la lista con valori pari a 0
    s = [0] * Lungh
    print(s)
    for i in range(Lungh):
        s[i] = random.random()
        return s

a = ListaCasuale(8)
print(a)
```



Tuple di argomenti a lunghezza variabile

- Le funzioni possono ricevere un numero variabile di argomenti
- Un nome di parametro che comincia con *, raccoglie gli argomenti in un a tupla

```
#raccolta
def stampa_tutti(*arg):
    print(arg)
#spacchettamento
def stampa(t1,t2):
    x=t1+t2
    print(x)

stampa_tutti(1,2.0,'3')
stampa(1,2) #chiamata classica
t=(1,2)
stampa(*t) #chiamata con tupla (spacchettamento al momento della chiamata)
#errore:
stampa(3)
```



Liste e Tuple: zip (1)

- zip è una funzione predefinita che riceve due o più sequenze e rende una lista di tuple, dove ciascuna tupla contiene un elemento di ciascuna sequenza.
- L'esempio connette una stringa con una lista di numeri interi:

```
s = 'abc'
t = [0,1,2]

for coppia in zip(s,t):
    print(coppia)
```

```
('a', 0)
('b', 1)
('c', 2)
>
```

Liste e Tuple: zip (2)

- Un oggetto zip è un tipo di **iteratore**, ovvero un qualsiasi oggetto in grado di iterare attraverso una sequenza
- Gli iteratori sono simili alle liste MA a differenza delle liste, non è possibile usare un indice per scegliere un elemento da un iteratore
- E' possibile creare una lista tramite l'oggetto zip per poi usare i metodi delle liste
- Se si parte da lista e una stringa con il numero di caratteri pari alla dimensione della lista:

```
s = 'abc'
t = [0,1,2]
print(list(zip(s,t)))
```

```
[('a', 0), ('b', 1), ('c', 2)]
```

Liste e Tuple: zip (3)

- E' possibile creare una lista tramite l'oggetto zip per poi usare i metodi delle liste
- Se si parte da lista e una stringa con il numero di caratteri diverso rispetto alla dimensione della lista, allora il risultato ha la lunghezza di quella più corta:

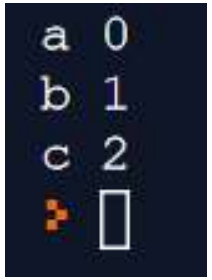
```
s = 'abcd'
t = [0,1,2]
u = 'AA'
print(list(zip(s,t)))
print(list(zip(s,u)))
#alternativa per stampare
for lettera, numero in (zip(s,t)):
    print(lettera, numero)
```

```
[('a', 0), ('b', 1), ('c', 2)]
[('a', 'A'), ('b', 'A')]
a 0
b 1
c 2
```

Liste e Tuple: zip (4)

- Combinando quanto visto, si può fare una funzione:

```
def corrispondenza(t1, t2): #definizione della funzione
    for x,y in zip(t1,t2):
            print(x,y)

s = 'abc'
t = [0,1,2]
corrispondenza(s,t) #chiamata della funzione
```



Liste e Tuple: uso di enumerate

- Se si vuole attraversare gli elementi di una sequenza e i loro indici, è possibile usare la funzione predefinita enumerate:

```
for indice, elemento in enumerate('abc'):
    print(indice, elemento)
```

- Il risultato è un oggetto enumerate, che itera una sequenza di coppie:
  - Ogni coppia contiene un indice (a partire da 0) e un elemento della sequenza di partenza
  - Si noti che l'output è della stessa forma di quella dell'esempio precedente
  - Cambia l'ordine. Perchè?

Dizionari e tuple: metodo *items*

- I dizionari supportano un metodo detto *items*, che restituisce una sequenza di tuple. Dove ogni tupla è una delle coppie chiave-valore:

```
d = {'a':0,'b':1,'c':2}
t = d.items()
print(d)
print('Iterando le coppie chiave-valore:')
for chiave, valore in t:
    print(chiave, valore)
print()

d2 = {0:'a',1:'b',2:'c'}
t2 = d2.items()
print(d2)
print('Iterando le coppie chiave-valore:')
for chiave, valore in t2:
    print(chiave, valore)
```

```
{'a': 0, 'b': 1, 'c': 2}
Iterando le coppie chiave-valore:
a 0
b 1
c 2

{0: 'a', 1: 'b', 2: 'c'}
Iterando le coppie chiave-valore:
0 a
1 b
2 c
>
```

Dizionari e tuple: inizializzare un Dizionario con una tupla

- E' possibile inizializzare un Dizionario con un tupla:

```
tupla = ('c',2),('a',0),('b',1)
print(tupla)
t = [('c',2),('a',0),('b',1)]
print(t)
d1 = dict(t)
print(d1)
print('Combinando dict e zip si può creare un dizionario in modo conciso')
d2 = dict( zip('ABC', range(3)))
print(d2)
```

```
(('c', 2), ('a', 0), ('b', 1))
[('c', 2), ('a', 0), ('b', 1)]
{'c': 2, 'a': 0, 'b': 1}
Combinando dict e zip si può creare uun dizionario in modo conciso
{'A': 0, 'B': 1, 'C': 2}
```

Dizionari e tuple: inizializzare un Dizionario con una tupla (2)

- L'uso delle tuple come chiavi di un dizionario è frequente (meno usato è l'uso delle liste poiché le liste sono mutabili)

```
t2 = [(('Mario','Rossi'),'05555555'),
(('Claudio','Sartu'),'066667777' ),
(('Anita','Bianchi'),'00222222')]
elenco = dict(t2)
print('Partendo da una rubrica inizializzata
tramite una lista di tuple:')
print(elenco)
print('Si stampano i vai nomi, cognomi e numeri
di telefono:')
for nome, cognome in elenco:
    print(nome, cognome, elenco[nome, cognome])
```

```
Partendo da una rubrica inizializzata tramit
e una lista di tuple:
{('Mario', 'Rossi'): '05555555', ('Claudio',
 'Sartu'): '066667777', ('Anita', 'Bianchi')
: '00222222'}
Si stampano i vai nomi, cognomi e numeri di
telefono:
Mario Rossi 05555555
Claudio Sartu 066667777
Anita Bianchi 00222222
```

Sequenze di sequenze (1)

- Fino ad ora si è parlato di liste di tuple
- Quasi tutti gli esempi fatti su Tuple e Liste funzionano anche con:
  - Liste di Liste
  - Tuple di Tuple
  - Tuple di Liste
- In molti casi i diversi tipi di sequenze son infatti intercambiabili
- In generale si parla di sequenze di sequenze
- E' necessario comprendere allora in base a quale criterio è meglio usare una scelta piuttosto che un'altra
- Le stringhe sono ovviamente più limitate perché gli elementi devono essere dei caratteri e sono anche immutabili

Sequenze di sequenze (2)

- Le Liste sono usate più frequentemente delle Tuple perché le Liste sono Mutabili
- In alcuni casi tuttavia è preferibile usare le Tuple:
  - Se si deve restituire un oggetto (output di una funzione), è più semplice fare return di una Tupla, anziché di una lista
  - Se serve una sequenza da usare come chiave in un Dizionario allora è necessario usare oggetti immutabili e quindi Tuple o Stringhe
  - Se si sta passando una sequenza come argomento (input) di una funzione, allora usare le Tuple riduce la possibilità di comportamenti imprevisti dovuti agli Alias
- Le Tuple sono Immutabili quindi NON possiedono metodi come sort o reverse che servono invece per modificare liste esistenti
- Pyhton tuttavia contiene la funzione sorted, che chiede in input una sequenza e rende una nuova lista con gli stessi elementi della sequenza, ordinati
- Python contiene anche la funzione reversed, che prende in ingresso una sequenza e restituisce un iteratore che attraversa la lista in ordine inverso

Sarebbe bene aggiungere esempi

Strutture dati – se avanza tempo da riprendere

- Liste, Dizionari e Tuple, sono esempi di strutture dati
- E' possibile fare uso di strutture dati composte: liste di tuple, dizionari che contengono tuple come chiavi e list
- Vanno pero' gest ato. Ad esempio se un programma si aspetta una lista che contiene un numero intero e invece gli viene passato un numero intero puro, NON funzionerà!

DA RIVEDERE

```
from structshape import structshape

t = [1,2,3]

print(structshape(t))
```

http://thinkpython2.com/code/structshape.py

Capitolo 13: Scelta della struttura Dati

Esercizi da Fare pag. 127

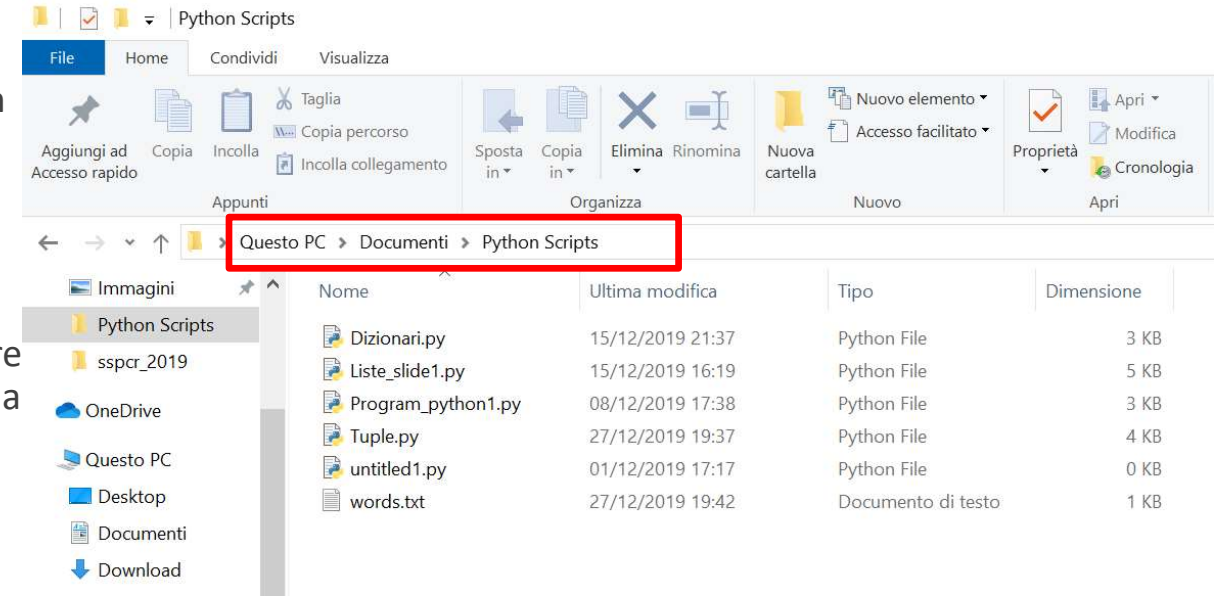
File

FILE: concetto di ‘Persistenza’

- I programmi visti fino ad ora sono transitori, ovvero: vengono eseguiti per breve tempo, producono un risultato. Quando i programmi vengono chiusi però il risultato da essi prodotto svanisce
- Esistono programmi persistenti:
  - Sono eseguiti per un lungo tempo (di continuo a frequenze regolari)
  - Mantengono almeno una parte dei risultati prodotti (dati in output) archiviati in modo permanente
  - Se vengono arrestati o riavviati, riprendono i loro lavoro da dove era stato fermato
- I dati vengono archiviati in FILE o Database
- Esempi:
  - Sistemi Operativi, Web Server, etc.

FILE: Lettura e Scrittura (1)

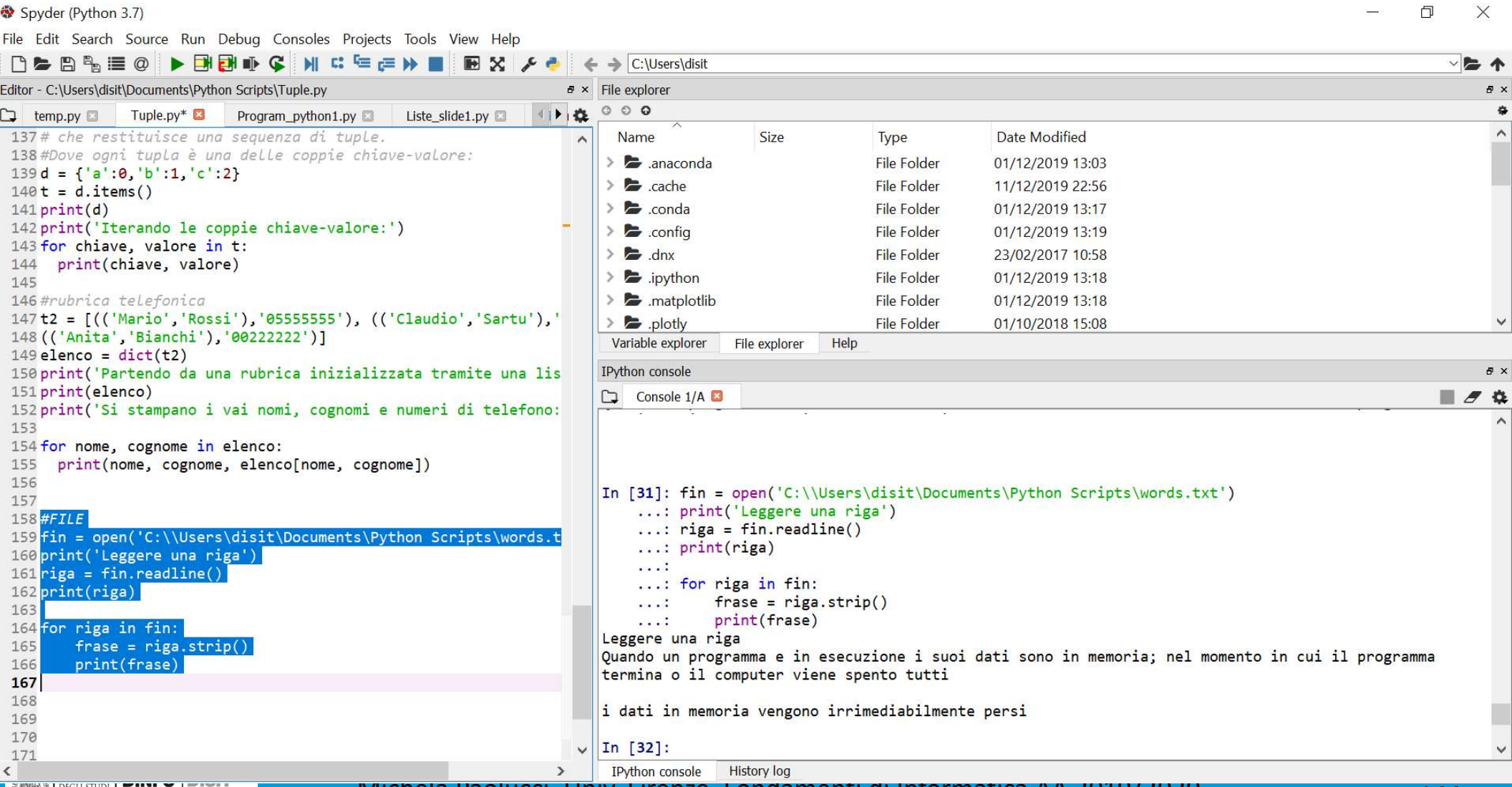
- File di Testo = Sequenza di caratteri salvata su un dispositivo permanente (es: disco fisso, hard disk-esterno, USB, etc.)
- Aprire e leggere un file:
  - Creare un file di testo nel filesystem (ad esempio words.txt)
  - Copiare il percorso o path (es: 'C:\Users\disit\Documents\Python Scripts')
  - Usare path + nome del file per aprire e leggere tale file con un programma Pyhton



FILE: Lettura

- File di Testo = Sequenza di caratteri salvata su un dispositivo permanente (es: disco fisso, hard disk-esterno, USB, etc.)
- Aprire e leggere un file:

```
#FILE
fin = open('C:\\Users\disit\Documents\Python Scripts\words.txt')
print('Leggere una riga')
riga = fin.readline()
#si stampa una sola riga
print(riga)

#Si stampano tutte le righe
for riga in fin:
    frase = riga.strip()
    print(frase)
```

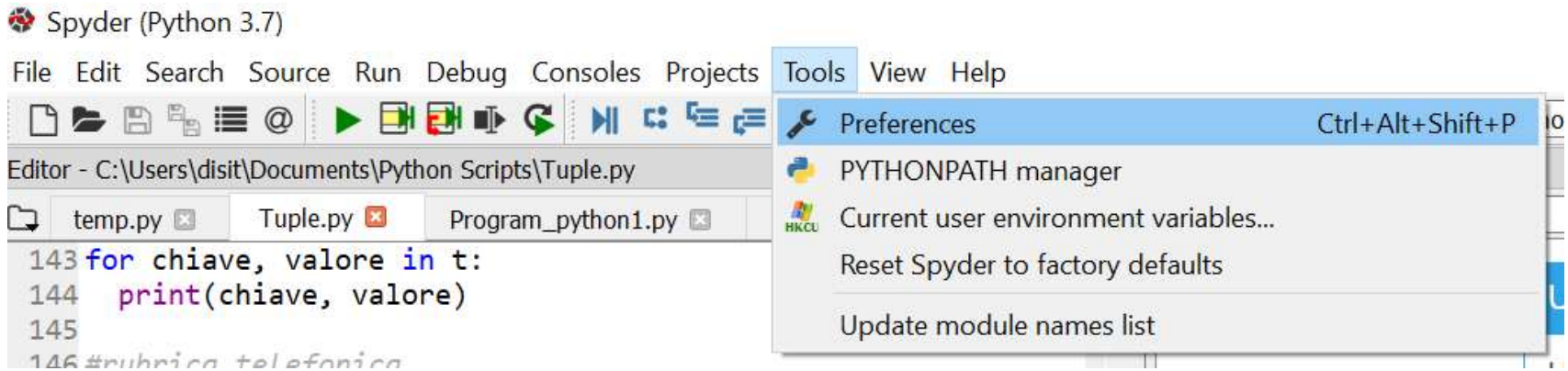

Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\disit
Editor - C:\Users\disit\Documents\Python Scripts\Tuple.py
temp.py
Tuple.py*
Program_python1.py
Liste_slide1.py
che restituisce una sequenza di tuple.
#Dove ogni tupla è una delle coppie chiave-valore:
d = {'a':0,'b':1,'c':2}
t = d.items()
print(d)
print('Iterando le coppie chiave-valore:')
for chiave, valore in t:
print(chiave, valore)
#rubrica telefonica
t2 = [(('Mario','Rossi'),'05555555'), (('Claudio','Sartu'),'
(('Anita','Bianchi'),'00222222')]
elenco = dict(t2)
print('Partendo da una rubrica inizializzata tramite una lis
print(elenco)
print('Si stampano i vai nomi, cognomi e numeri di telefono:
for nome, cognome in elenco:
print(nome, cognome, elenco[nome, cognome])
#FILE
fin = open('C:\\Users\disit\Documents\Python Scripts\words.t
print('Leggere una riga')
riga = fin.readline()
print(riga)
for riga in fin:
frase = riga.strip()
print(frase)
File explorer
Name
Size
Type
Date Modified
.anaconda File Folder 01/12/2019 13:03
.cache File Folder 11/12/2019 22:56
.conda File Folder 01/12/2019 13:17
.config File Folder 01/12/2019 13:19
.dnx File Folder 23/02/2017 10:58
.ipython File Folder 01/12/2019 13:18
.matplotlib File Folder 01/12/2019 13:18
.plotly File Folder 01/10/2018 15:08
Variable explorer
File explorer
Help
IPython console
Console 1/A
In [31]: fin = open('C:\\Users\disit\Documents\Python Scripts\words.txt')
...: print('Leggere una riga')
...: riga = fin.readline()
...: print(riga)
...:
...: for riga in fin:
...: frase = riga.strip()
...: print(frase)
Leggere una riga
Quando un programma e in esecuzione i suoi dati sono in memoria; nel momento in cui il programma
termina o il computer viene spento tutti
i dati in memoria vengono irrimediabilmente persi
In [32]:
IPython console
History log

FILE: path assoluto e relativo

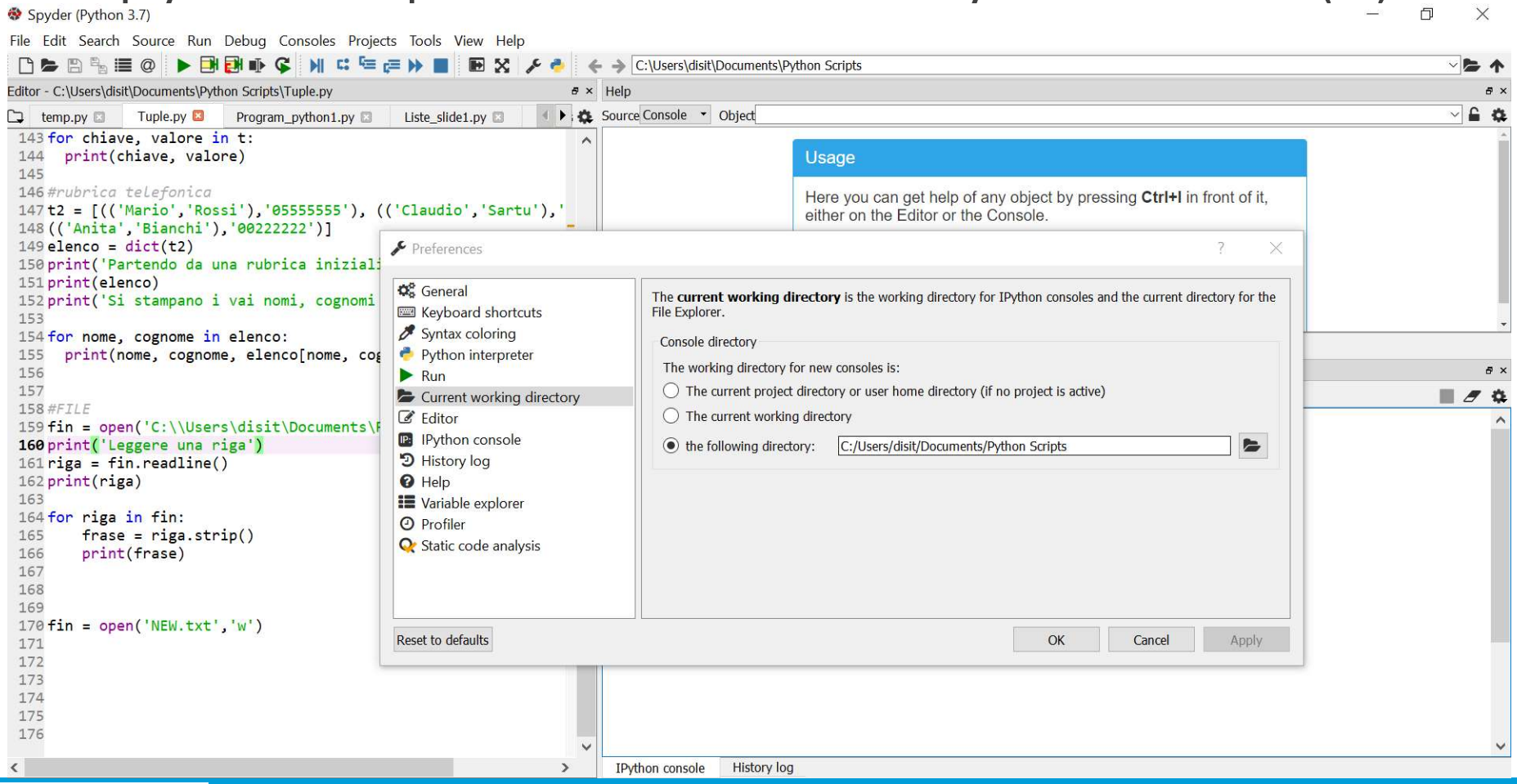
- Cosa succede se si scrive:
  - `fin = open('words.txt’)`
- Che differenza c’è rispetto a:
  - `fin = open('C:\\Users\disit\Documents\Python Scripts\words.txt’)`
- Nel secondo caso si sta usando un path Assoluto, si parte quindi dalla radice (C:) per poi arrivare alla foglia (words.txt), ovvero al nome del file
- Nel secondo caso si sta usando il path Relativo
  - Quindi la IDE che stiamo usando cerca tale file nella directory di lavoro
  - Nel primo caso quindi la IDE potrebbe NON trovare il file!! (a meno che il programmatore non lo abbia inserito nella directory corretta)

Spyder: impostare la Directory di default (1)

- Per impostare la directory andare nel menu Tools > Preferences > Current Working Directory



Spyder: impostare la Directory di default (2)




```
for chiave, valore in t:
  print(chiave, valore)

#rubrica telefonica
t2 = [(('Mario','Rossi'),'05555555'), (('Claudio','Sartu'),'
(('Anita','Bianchi'),'00222222')]
elenco = dict(t2)
print('Partendo da una rubrica iniziali
print(elenco)
print('Si stampano i vai nomi, cognomi

for nome, cognome in elenco:
  print(nome, cognome, elenco[nome, cog

#FILE
fin = open('C:\\Users\disit\Documents\F
print('Leggere una riga')
riga = fin.readline()
print(riga)

for riga in fin:
    frase = riga.strip()
    print(frase)


fin = open('NEW.txt','w')
```

Spyder: impostare la Directory Corrente o Directory di Default (2)

- Posizionare il file nella directory scelta
- A questo punto è possibile anche usare il path relativo (con consapevolezza)
- Il programma precedente può essere scritto quindi anche con il path relativo (e funziona in modo equivalente adesso che è stato impostato il path di default in modo opportuno):

```
#FILE
fin = open('words.txt')
print('Leggere una riga')
riga = fin.readline()
#si stampa una sola riga
print(riga)

#Si stampano tutte le righe
for riga in fin:
    frase = riga.strip()
    print(frase)
```

Nomi di File e percorsi (1)

- I file sono organizzati quindi in directory (cartella o folder)
- Ogni programma in esecuzione ha una directory corrente (o directory di default)
- Il Modulo os fornisce delle funzioni per lavorare con le directory (OS = Operative System)

```
#scrivere su file, modulo os
import os
cwd = os.getcwd()
print(cwd) #cwd = current working directory
```

```
In [24]: import os
    ...: cwd = os.getcwd()
    ...: print(cwd)
C:\Users\disit\Documents\Python Scripts
```

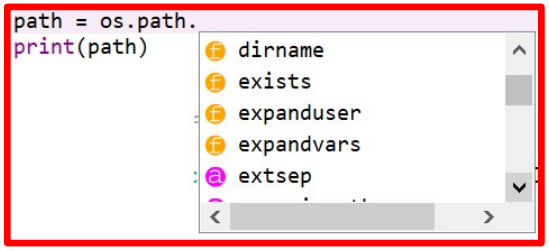
- Se si vuole il path assoluto dato il nome di un file si usa:

```
awd = os.path.abspath('NEW.txt')
print(awd)
```

```
C:\Users\disit\Documents\Python Scripts\NEW.txt
```

Nomi di File e percorsi (2)

- os.path fornisce varie funzioni per lavorare con i nomi dei file e delle directory:
  - `path = os.path.exists('NEW.txt')`
    - Controlla l'esistenza del file e rende True o False
  - `path = os.path.isdir('C:/Users/disit/Documents/Python Scripts')`
    - Controlla se l'input è una Directory e rende True o False
  - `file = os.path.isfile('NEW.txt')`
    - Controlla se l'input è un file e rende True o False
  - `file = os.path.join(dir_name, name)`
    - Prende in ingresso due argomenti: nome di una directory e il nome di un file e rende il percorso completo



Esempio 1

```
import os
cwd = os.getcwd()
print(cwd)
#absolute path form file name
awd = os.path.abspath('NEW.txt')
#return a booolean: True if the file exists or False if not exists
path = os.path.exists('NEW.txt')
print(int(path)) #conversion to int
print(bool(path)) #conversion to boolean: no sense here
print(path) #path is already a boolean
if (path == 1):#path is automatically converted in int
        print('Il file esiste e si trova al seguente percorso:’)
        print(awd)
else:
        print('Il file non esiste!')
```

```
C:\Users\disit\Documents\Python Scripts
1
True
True
Il file esiste e si trova al seguente percorso:
C:\Users\disit\Documents\Python Scripts\NEW.txt
```

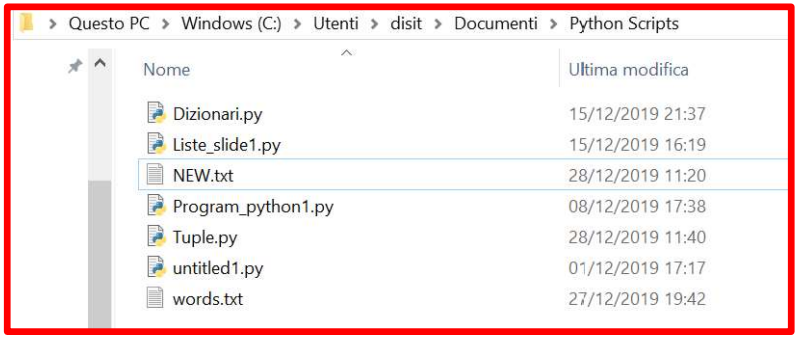
Esempio 2

```
#scrivere su file, modulo os
import os
print('-------Directory: ')
path = os.path.isdir('C:/Users/disit/Documents/Python Scripts')
print(path)
path = os.path.isdir('Hello!')
print(path)

print('------File: ')
file = os.path.isfile('NEW.txt')
print(file)
file = os.path.isfile('NEWy.txt')
print(file)
```

```
-------Directory:
True
False
------File:
True
False
```

Esempio 3

```
#scrivere su file, modulo os
import os


def esplora(dir_name):
        for name in os.listdir(dir_name):
                file = os.path.join(dir_name, name)
                if os.path.isfile(file):
                        print(file)
                else:
                        esplora(dir_name)
esplora('C:/Users/disit/Documents/Python Scripts')
```



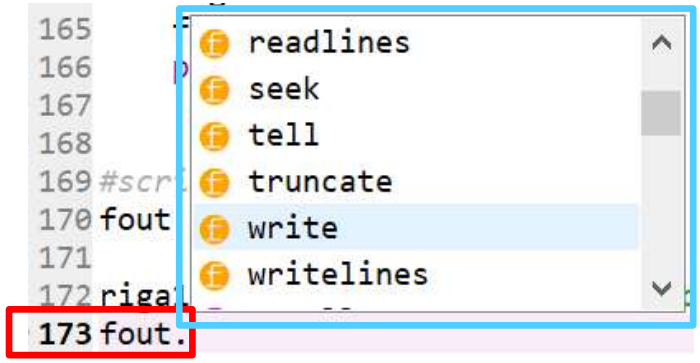

```
C:/Users/disit/Documents/Python Scripts\Dizionari.py
C:/Users/disit/Documents/Python Scripts\Liste_slide1.py
C:/Users/disit/Documents/Python Scripts\NEW.txt
C:/Users/disit/Documents/Python Scripts\Program_python1.py
C:/Users/disit/Documents/Python Scripts\Tuple.py
C:/Users/disit/Documents/Python Scripts\untitled1.py
C:/Users/disit/Documents/Python Scripts\words.txt
```

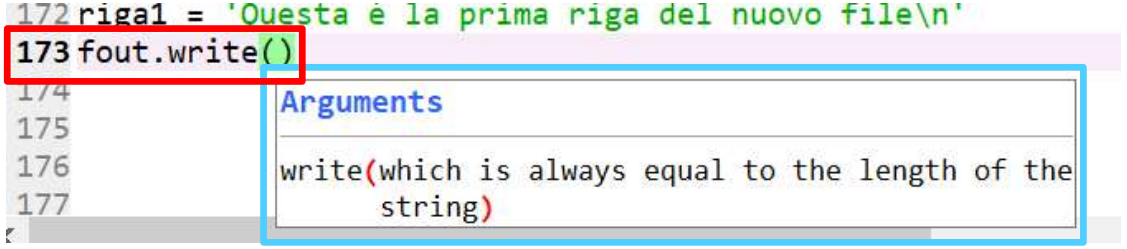
FILE: Scrittura (1)

- Per scrivere su un file è necessario indicare la modalità ‘w’ in fase di apertura del file (come secondo parametro)
  - `fout = open('NEW.txt','w’)`
- In questo modo:
  - Se il file NON esiste, viene creato
  - Se il file esiste già, ne viene cancellato il contenuto (viene ripulito dai vecchi dati) in modo da poterci lavorare
- Esistono alcuni metodi per lavorare sui file iniziando da:
  - Write
  - Close

FILE: Scrittura (2)

- NOTA: Alcune IDE mostrano i vari metodi degli oggetti e i relativi argomenti (input), in questo caso vediamo alcuni dei metodi relativi all'oggetto file:

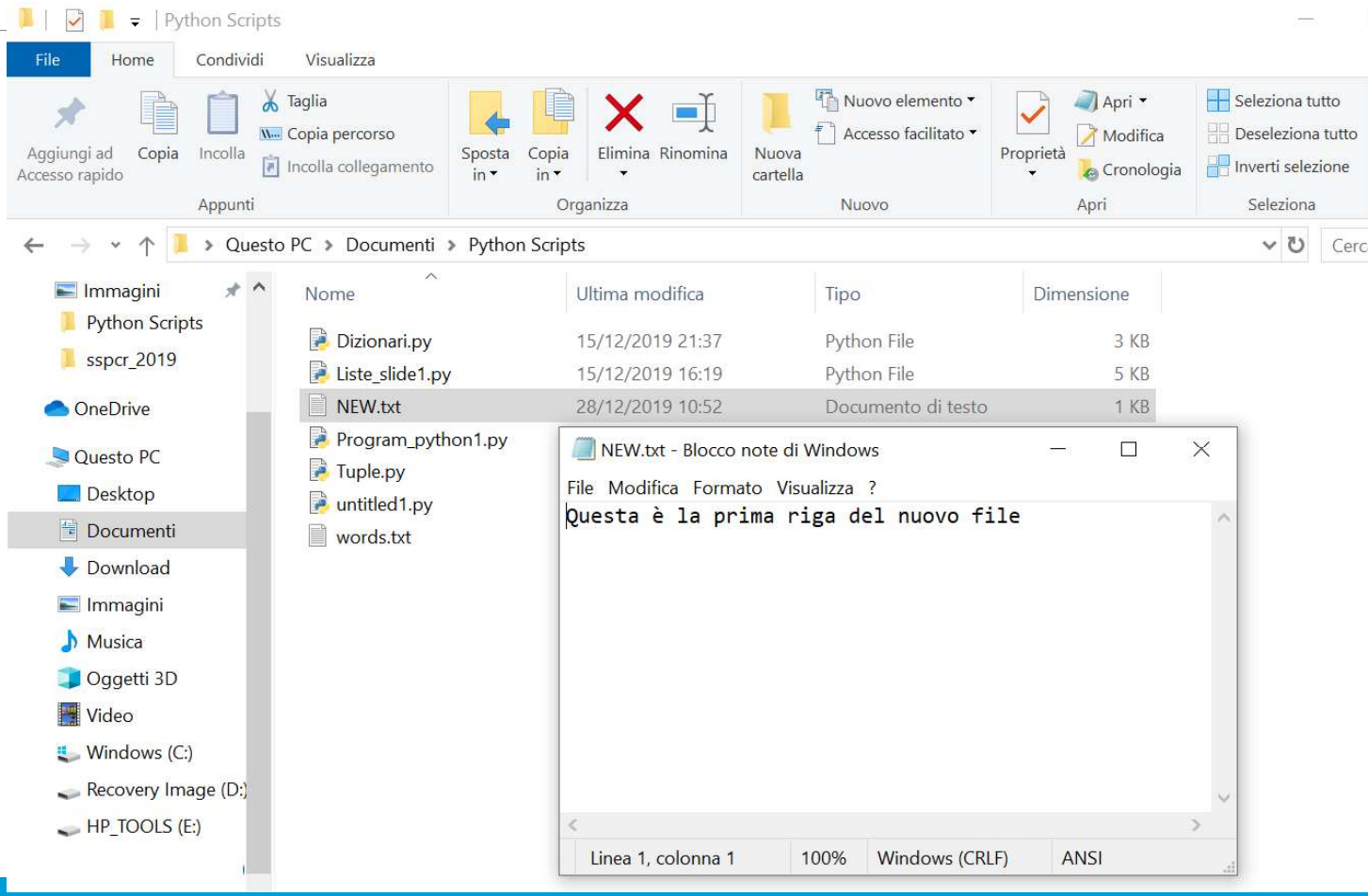





```
#scrivere su file
fout = open('NEW.txt','w')
riga1 = 'Questa è la prima riga del nuovo file\n'
fout.write(riga1)
fout.close()
```

FILE: Scrittura (3)

- Una volta chiuso il file da Python, sarà possibile vedere il risultato sul file
- NOTA: è sempre opportuno chiudere il file, se non viene effettuato dal programmatore, il file viene comunque chiuso al termine del programma



FILE: Scrittura (4)

- L'argomento del metodo write() deve essere una stringa
- Se si vogliono inserire degli argomenti di tipo diverso in file, è necessario prima convertirli in stringhe

```
169 #scrivere su file
170 fout = open('NEW.txt','w')
171
172 riga1 = 'Questa è la prima riga del nuovo file\n'
173 riga2 = 52
174 fout.write(riga1)
175 fout.write(riga2)
176 fout.close()
```

```
Traceback (most recent call last):

  File "<ipython-input-11-20996615a485>", line 6, in <module>
    fout.write(riga2)

TypeError: write() argument must be str, not int
```

```
#scrivere su file
fout = open('NEW.txt','w')
riga1 = 'Questa è la prima riga del nuovo file\n'
riga2 = 52
fout.write(riga1)
fout.write(str(riga2)) #uso di str() per convertire in stringhe
fout.close()
```

FILE: operatore di formato (1)

- Come alternativa a str(), è possibile usare l'operatore di formato %
- Se % è applicato a numeri interi, allora ha significato di modulo:

```
a = 10
b = 3
if a%b == 0:
    print('a è divisibile per b')
else:
    print('a NON è divisibile per b')
```

- NOTA: gli operatori di formato sono gli stessi visti per le stringhe, anche la sintassi è la stessa!

FILE: operatore di formato (2)

- Se il primo operando è una stringa, allora % diventa l'operatore di formato
- Il primo operando è detto **stringa di formato**, che contiene una o più sequenze di formato che specificano il formato del secondo operando
- Il risultato è una stringa:

```
cammelli = 52
stringa = 'Ho contato %d cammelli' % cammelli
print(stringa)
```

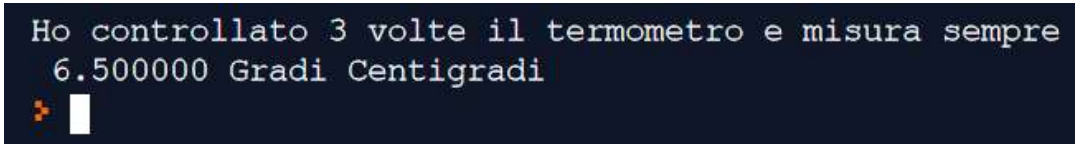
- NOTA: %d si usa per gli interi

```
Ho contato 52 cammelli
>
```

FILE: operatore di formato (3)

- Se la stringa di formato contiene più sequenze di formato, allora il secondo operando deve essere una tupla
- Ciascuna sequenza di formato corrisponde ad un elemento della tupla. L'ordine è ovviamente rilevante:

```
#operatore di formato
n_conteggio = 3
valore = 6.5
unita_misura = 'Gradi Centigradi'
stringa = 'Ho controllato %d volte il termometro e misura
sempre %f %s' % (n_conteggio, valore, unita_misura)
print(stringa)
```



FILE: operatore di formato (3)

- NOTE:
  - %d -> interi
  - %f -> float
  - %s -> stringhe
- Il numero degli elementi nella tupla DEVE essere uguale al numero di volte in cui l'operatore di formato compare nella stringa di formato
- Se si mettono meno elementi nella tupla:
  - "TypeError: not enough arguments for format string"
- Se si mettono in ordine sbagliato o in modo tale da NON rispettare i formati:
  - Esempio: 'TypeError: must be real number, not str'
- Se si mettono più elementi nella tupla rispetto al numero di operatori di formato:
  - 'TypeError: not all arguments converted during string formatting'

FILE: operatore di formato (4)

https://docs.python.it/html/lib/typesseq-strings.html

Conversione	Significato
d	Numero intero decimale con segno.
i	Numero intero decimale con segno.
o	Ottale senza segno.
u	Decimale senza segno.
x	Esadecimale senza segno (minuscolo).
X	Esadecimale senza segno (maiuscolo).
e	Numero in virgola mobile, in formato esponenziale (minuscolo).
E	Numero in virgola mobile, in formato esponenziale (maiuscolo).
f	Decimale in virgola mobile.
F	Decimale in virgola mobile.
g	Lo stesso di "`e`" se l'esponente è più grande di -4 o minore della precisione, "`f`" altrimenti.
G	Lo stesso di "`E`" se l'esponente è più grande di -4 o minore della precisione, "`F`" altrimenti.
c	Carattere singolo (accetta interi o stringhe di singoli caratteri).
r	Stringa (converte ogni oggetto Python usando `repr()`).
s	Stringa (converte ogni oggetto Python usando `str()`).
%	Nessun argomento viene convertito, riporta un carattere "`%`" nel risultato.

Metodo di formato delle stringhe

https://docs.python.org/3/library/stdtypes.html#str.format

`str.`**`format`**`(*args, **kwargs)`

Perform a string formatting operation. The string on which this method is called can contain literal text or replacement fields delimited by braces `{}`. Each replacement field contains either the numeric index of a positional argument, or the name of a keyword argument. Returns a copy of the string where each replacement field is replaced with the string value of the corresponding argument.

```
>>> "The sum of 1 + 2 is {0}".format(1+2)
'The sum of 1 + 2 is 3'
```

See Format String Syntax for a description of the various formatting options that can be specified in format strings.

Note: When formatting a number (`int`, `float`, `complex`, `decimal.Decimal` and subclasses) with the `n` type (ex: `'{:n}'.format(1234)`), the function temporarily sets the `LC_CTYPE` locale to the `LC_NUMERIC` locale to decode `decimal_point` and `thousands_sep` fields of `localeconv()` if they are non-ASCII or longer than 1 byte, and the `LC_NUMERIC` locale is different than the `LC_CTYPE` locale. This temporary change affects other threads.

Changed in version 3.7: When formatting a number with the `n` type, the function sets temporarily the `LC_CTYPE` locale to the `LC_NUMERIC` locale in some cases.

File: Database (1)

- Un database serve per archiviare dati
- Esistono varie modalità di archiviazione dati
- Con Python è possibile usare i file per archiviare i dati
- Molti Database sono organizzati come un dizionario:
  - Fanno una mappatura da chiavi in valori
- I database, al contrario dei dizionari, solitamente risiedono su disco fisso (o dispositivi permanenti) in modo da persistere ance quando il programma viene chiuso
- Il modulo dbm di Python serve per creare e aggiornare file di database

File: Database (2)

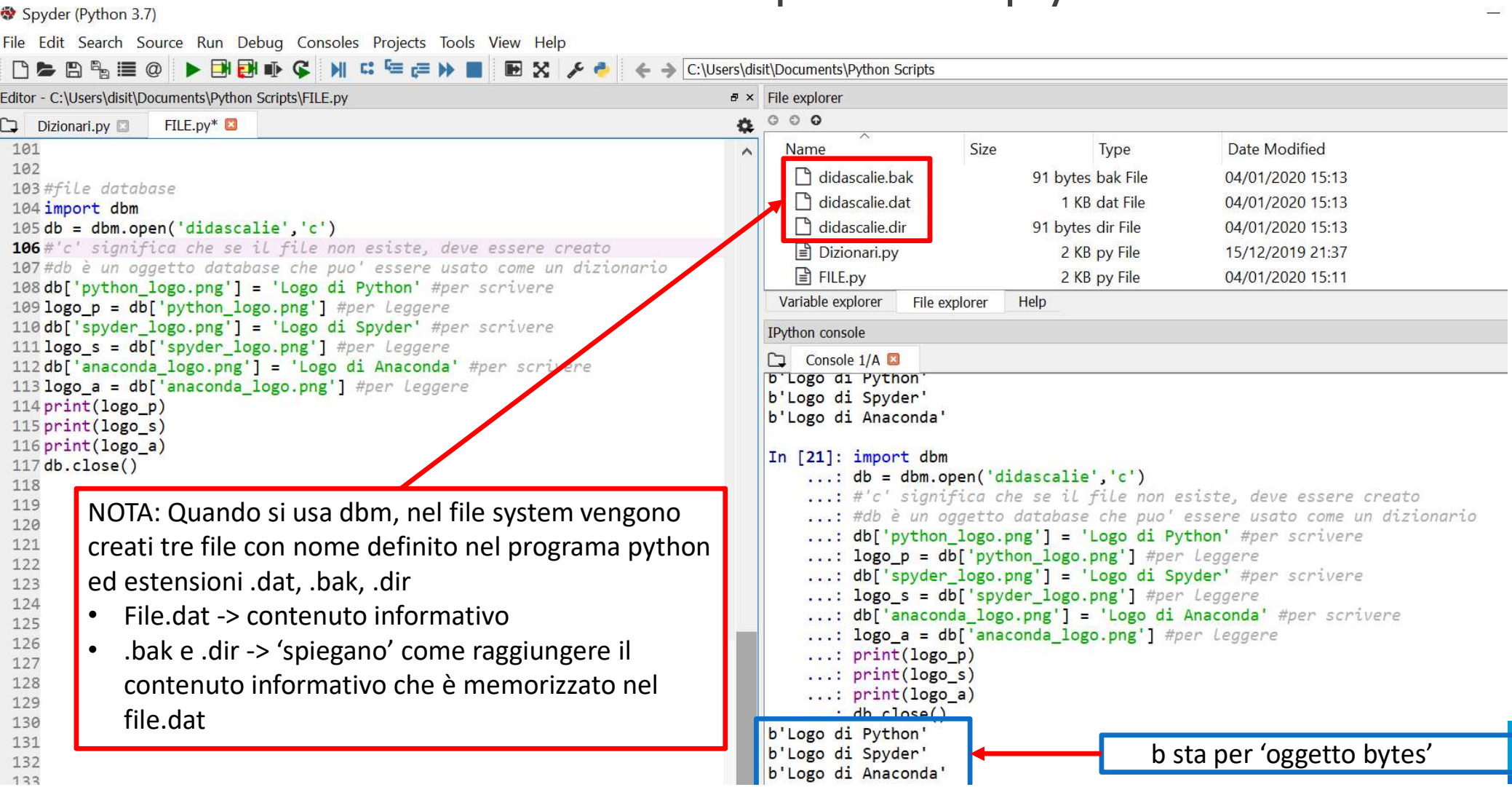
- Esempio: creare un database che contiene le didascalie di alcune immagini

```
#file database
import dbm
db = dbm.open('didascalie','c')
#'c' significa che se il file non esiste, deve essere creato
#db è un oggetto database che puo' essere usato come un dizionario
db['python_logo.png'] = 'Logo di Python' #per scrivere
logo_p = db['python_logo.png'] #per leggere
db['spyder_logo.png'] = 'Logo di Spyder' #per scrivere
logo_s = db['spyder_logo.png'] #per leggere
db['anaconda_logo.png'] = 'Logo di Anaconda' #per scrivere
logo_a = db['anaconda_logo.png'] #per leggere
print(logo_p)
print(logo_s)
print(logo_a)
db.close()
```

File: Database (3)

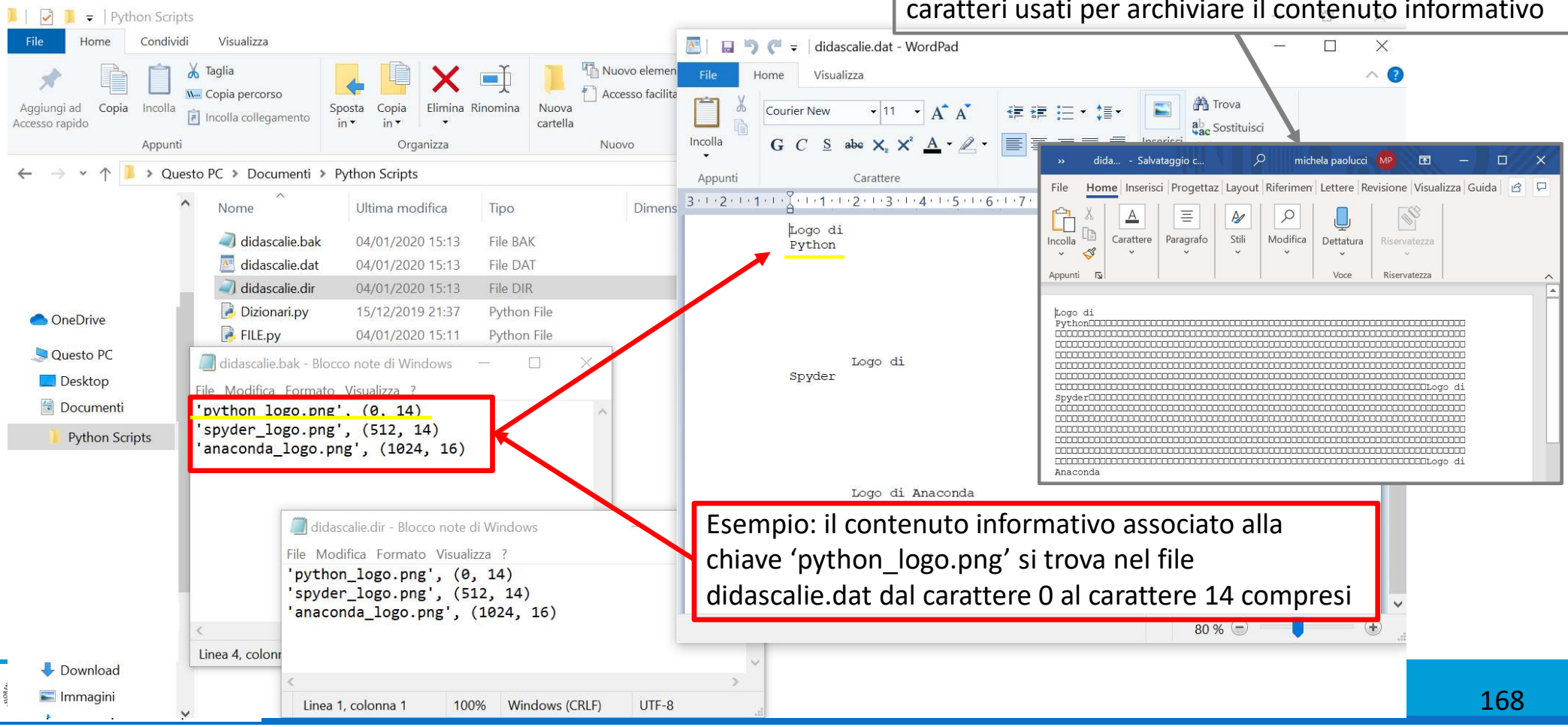
- Il contenuto informativo (valore) memorizzato nel file viene detto ‘oggetto bytes’
- Un oggetto bytes è simile ad una stringa (si approfondirà eventualmente in futuro)
- Se si fa una nuova assegnazione di un valore (o contenuto informativo) ad una chiave esistente, allora dbm sostituisce il vecchio valore
- Alcuni metodi dei dizionari, come keys e items, non funzionano con gli oggetti database
- Il ciclo for per l’iterazione invece funziona

File: Database – esempio su Spyder



File: DataBase -> file per la gestione del DB

NOTA: aprendo con Word, si puo' notare quali sono i caratteri usati per archiviare il contenuto informativo

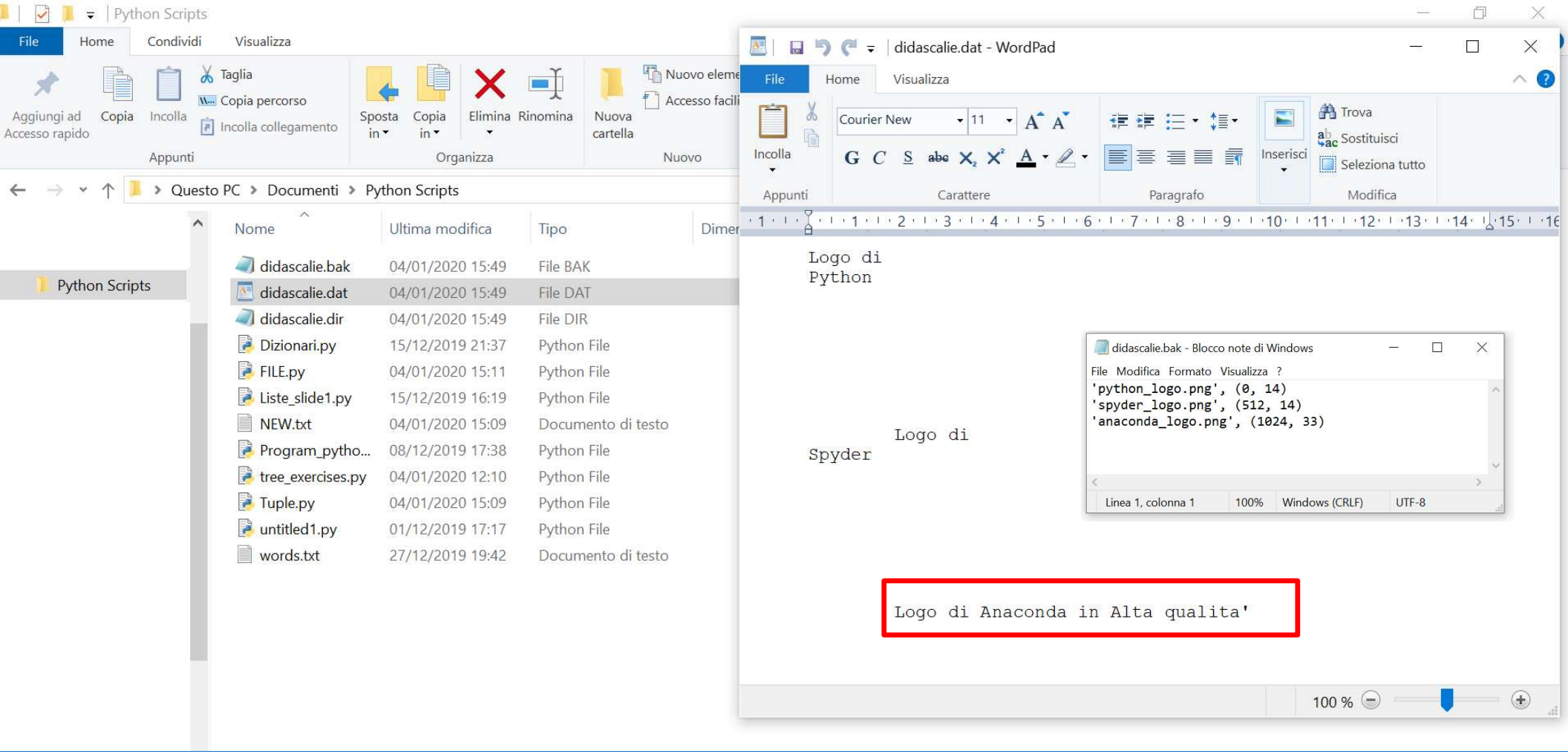


File: Database (4)

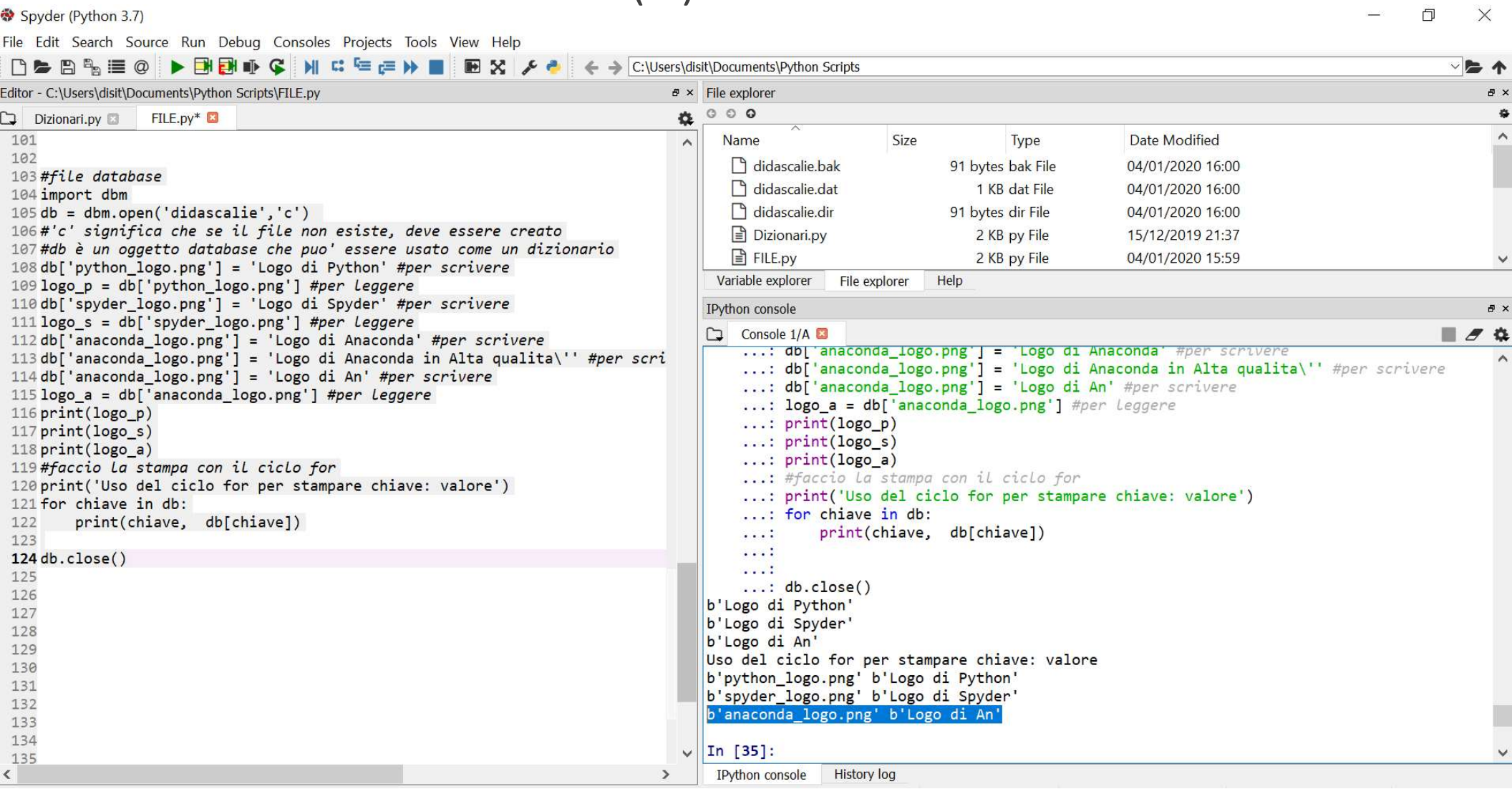
- Esempio: creare un database che contiene le didascalie di alcune immagini – sostituzione e uso del ciclo for per la stampa

```
#file database
import dbm
db = dbm.open('didascalie','c')
#'c' significa che se il file non esiste, deve essere creato
db['python_logo.png'] = 'Logo di Python' #per scrivere
logo_p = db['python_logo.png'] #per leggere
db['spyder_logo.png'] = 'Logo di Spyder' #per scrivere
logo_s = db['spyder_logo.png'] #per leggere
db['anaconda_logo.png'] = 'Logo di Anaconda' #per scrivere
db['anaconda_logo.png'] = 'Logo di Anaconda in Alta qualita\'' #per SOVRAscrivere
logo_a = db['anaconda_logo.png'] #per leggere
#stampa con il ciclo for
print('Uso del ciclo for per stampare chiave: valore')
for chiave in db:
print(chiave, db[chiave])

db.close()
```

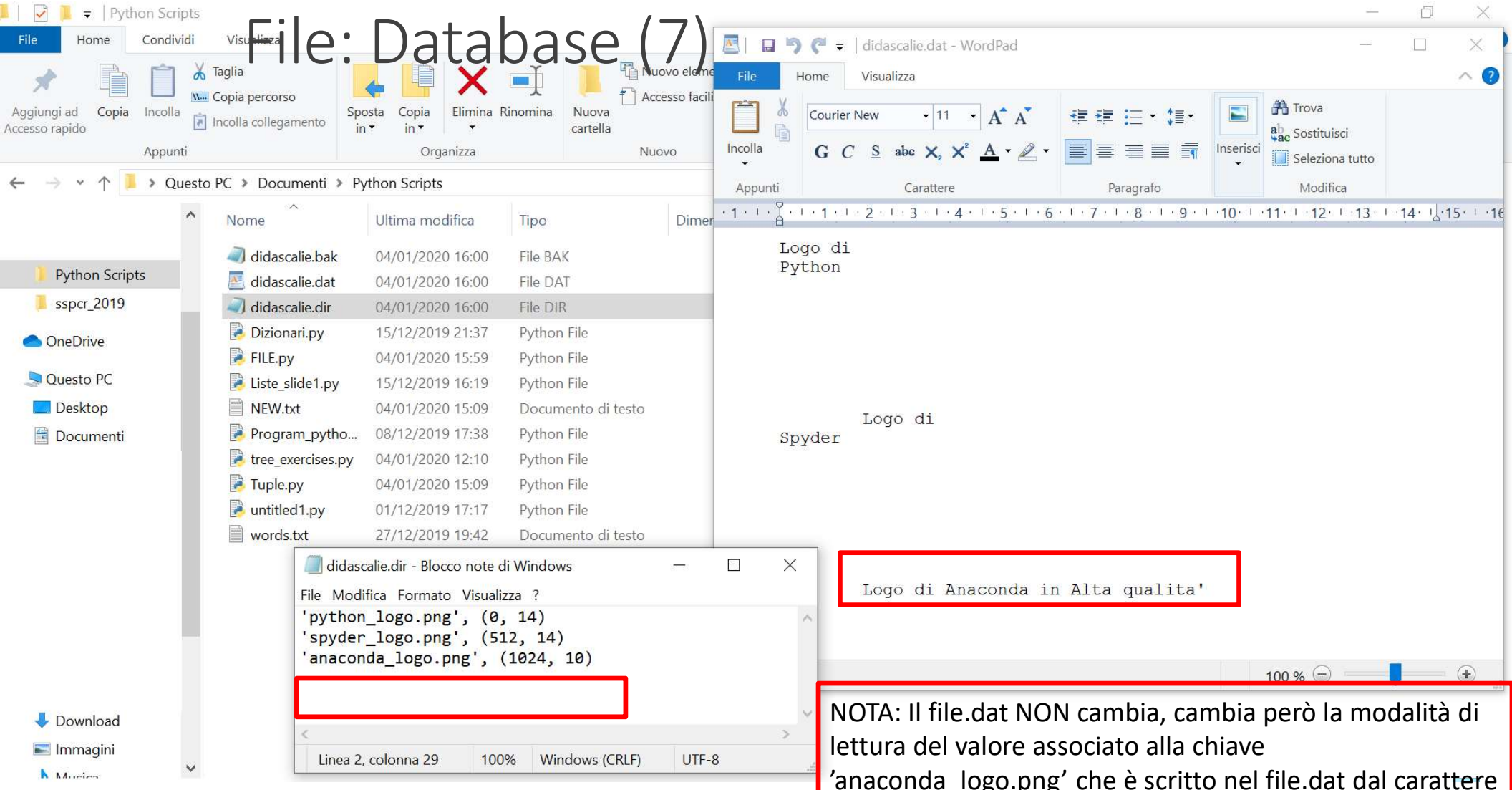
File: Database (5)



File: Database (6)



File: Database (7)



NOTA: Il file.dat NON cambia, cambia però la modalità di lettura del valore associato alla chiave 'anaconda_logo.png' che è scritto nel file.dat dal carattere 1024 al carattere 10 compresi (NON più fino al 33)

File: Database (8)

- Cosa succede se si scrive un programma in cui PRIMA nella chiave `'anaconda_logo.png'` si scrive `'Logo di Anaconda in Alta qualita\‘’`
- E POI si sovrascrive `'Logo di An’?`
- NOTA: In questo casi si taglia rispetto alla scrittura precedente

File: possible 'dbm.open – flags'

https://docs.python.org/3/library/dbm.html

`dbm.` **`open`**(*file*, *flag='r'*, *mode=0o666*)

Open the database file *file* and return a corresponding object.

If the database file already exists, the `whichdb()` function is used to determine its type and the appropriate module is used; if it does not exist, the first module listed above that can be imported is used.

The optional *flag* argument can be:

Value	Meaning
`'r'`	Open existing database for reading only (default)
`'w'`	Open existing database for reading and writing
`'c'`	Open database for reading and writing, creating it if it doesn't exist
`'n'`	Always create a new, empty database, open for reading and writing

The optional *mode* argument is the Unix mode of the file, used only when the database has to be created. It defaults to octal `0o666` (and will be modified by the prevailing umask).

File: Pickling (1)

- Quando si usa dbm si deve tenere presente che le chiavi e i valori devono essere delle stringhe, oppure bytes
- Se si cerca di usare un qualsiasi altro tipo, allora python da' errore
- In questo caso, il modulo pickle può essere di aiuto poiché trasforma 'quasi' ogni tipo di oggetto in una stringa, e quindi ritrasforma la stringa in oggetto
- pickle.dumps:
  - accetta un oggetto come parametro in ingresso
  - Restituisce una serializzazione, ovvero una rappresentazione sotto forma di una stringa
  - 'dumps' sta per 'dump string' (scarica stringa)

File: Pickling (2)

```
import pickle
t = [1,2,3]
ps = pickle.dumps(t)
print(ps)
```

- Il formato NON è leggibile (non è human readable): è stato creato per essere letto dalla libreria pickle
- Se si usa però pickle.loads(…) si risale al valore immesso

```
#FILE: pickling
import pickle
t = [1,2,3]
ps = pickle.dumps(t)
print(ps)
pl = pickle.loads(ps)
print('Usando pickle.loads:')
print(pl)
```

File: Pickling (3)

- Si noti che:
  - Anche se il nuovo oggetto caricato nel programma con pickle.loads, ha lo stesso valore di quello immesso dal programmatore… in realtà NON è lo stesso oggetto!

```
#FILE: pickling
import pickle
t = [1,2,3]
ps = pickle.dumps(t)
print(ps)
pl = pickle.loads(ps)
print('Usando pickle.loads:')
print(pl)
print(pl==t) #pl ha lo stesso valore di t
print(pl is t) #pl NON è lo stesso oggetto rispetto a t
```

File: Pickling (4)

main.py saved

```
1   #FILE: pickling
2   import pickle
3   t = [1,2,3]
4   ps = pickle.dumps(t)
5   print(ps)
6   pl = pickle.loads(ps)
7   print('Usando pickle.loads:')
8   print(pl)
9   print(pl==t) #pl ha lo stesso  valore di t
10  print(pl is t) #pl NON è lo stesso oggetto rispetto a t
```

```
b'\x80\x03]q\x00(K\x01K\x02K\x03e.'
Usando pickle.loads:
[1, 2, 3]
True
False
>
```

- pl NON è lo stesso oggetto di t
- Questo significa che: quando si usa pickle, si STA lavorando con UNA COPIA dell'oggetto inziale
- E' possibile usare pickle per archiviare in un database tutto ciò che NON è una stringa
- Questa combinazione è molto frequente ed è stata incapsulata in un modulo chiamato shelve

File: Pickling (5)

- Senza pickling era necessario convertire tutto in stringhe

```
fout = open('NEW.txt','w')
fout.write (str(12.3))
fout.write (str(4.567))
fout.write (str([1,2,3]))
fout.close()
#leggeere da file
fin = fout = open('NEW.txt','r')
r = fin.readline()
print(r)
```

- In questo caso però se si cerca di recuperare il valore dal file (ad esempio usando fin.readline()), si ottiene una stringa, e non l'informazione originale che era stata memorizzata dal programmatore
- Inoltre non è possibile sapere dove inizia o termina di preciso la stringa che definisce il valore nel file (si ricordi i file .bak o .dir che in questo caso NON esistono!)

File: Scrivere moduli (1)

- Qualunque file che contiene codice python puo' essere importato come modulo
- Esempio: supponiamo di avere un file contenente la definizione della funzione contarighe, che prende in ingresso il nome di un file e ne rende il numero di righe

```
def contarighe(nomefile):
    conta = 0
    for riga in open(nomefile):
            conta =+ 1
    return conta
#main – comando di prova per stampare il numero di righe
print(contarighe('wc.py'))
```

- Eseguendo il programma, legge se stesso e rende 7

File: Scrivere moduli (2)

Nome	Ultima modifica	Tipo	Dimensione
didascalie.bak	04/01/2020 16:00	File BAK	1 KB
didascalie.dat	04/01/2020 16:00	File DAT	2 KB
didascalie.dir	04/01/2020 16:00	File DIR	1 KB
Dizionari.py	15/12/2019 21:37	Python File	3 KB
FILE.py	04/01/2020 17:05	Python File	4 KB
Liste_slide1.py	15/12/2019 16:19	Python File	5 KB
NEW.txt	04/01/2020 16:51	Documento di testo	1 KB
Program_pytho...	08/12/2019 17:38	Python File	3 KB
tree_exercises.py	04/01/2020 12:10	Python File	3 KB
Tuple.py	04/01/2020 15:09	Python File	6 KB
untitled1.py	01/12/2019 17:17	Python File	0 KB
wc.py	04/01/2020 17:08	Python File	1 KB
words.txt	27/12/2019 19:42	Documento di testo	1 KB

wc.py - Blocco note di Windows

File Modifica Formato Visualizza ?

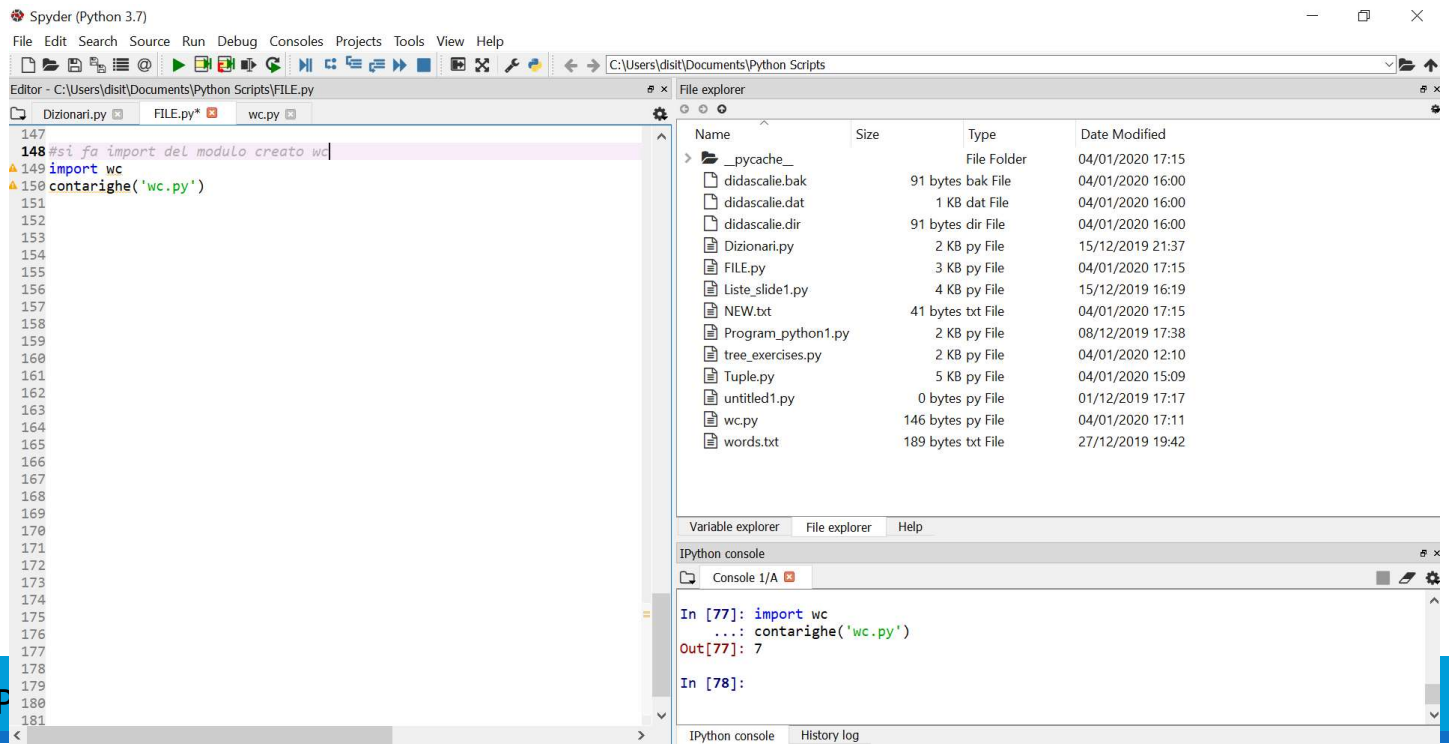
```
def contarighe(nomefile):
    conta = 0
    for riga in open(nomefile):
        conta += 1
    return conta

print(contarighe('wc.py'))
```

Linea 7, colonna 27 | 100% | Windows (CRLF) | UTF-8

File: Scrivere moduli (3)

- E' possibile anche importare la definizione come modulo e/o usarlo da un altro programma
- Esempio 2: importare il modulo creato in un altro programma
  - Si effettua nella stessa modalità in cui sono stati importati i moduli standard Python:

```
import wc

contarighe('wc.py')
```



File: Scrivere moduli (4)

- NOTA: per come è stato scritto il modulo wc.py, il comando di prova (contarighe('wc.py')) viene sempre eseguito

- INFATTI da riga di comando si vede il risultato

- Solitamente un file che poi verrà importato come modulo, definisce solo le funzioni MA NON le esegue, allora nel file wc.py si scrive:

```
def contarighe(nomefile):
    conta = 0
    for riga in open(nomefile):
            conta += 1
    return conta
#main - comando di prova
if __name__ == '__main__':
    print(contarighe('wc.py'))
```

- __name__ è una variabile predefinita impostata all'avvio del programma

- SE il programma viene usato come modulo (ovvero viene importato da un altro programma), il codice di prova NON viene eseguito (viene saltato)

File: Scrivere moduli (5)

- Una volta scritto il file usando __name__ SE si rilancia da riga di comando si NOTA che la stampa NON avviene più!! (proprio perché il comando di prova è preceduto da un if)
- SE si aggiunge la riga:
  - `print(__name__)`
- Allora stampa (solo) il valore di tale variabile

Che succede???? Che valore ha la variabile __name__ nei due casi seguenti??

- Caso 1: lancio wc.py direttamente
- Caso 2: importo wc
  -

File: importare moduli (1)

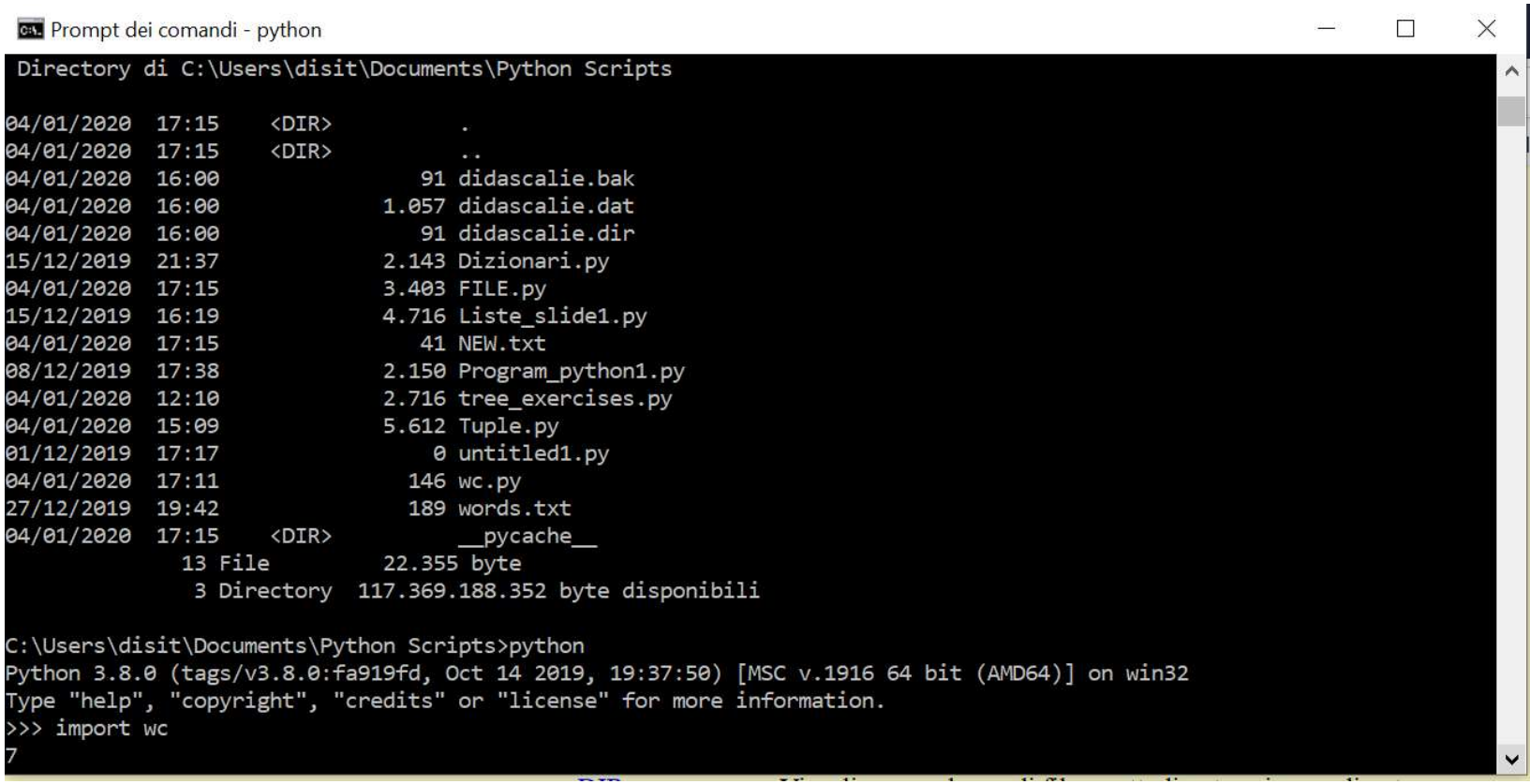
- Una volta scritto il file usando __name__ SE si rilancia da riga di comando si NOTA che la stampa NON avviene più!! (proprio perché il comando di prova è preceduto da un if)
- SE si aggiunge la riga:
  - `print(__name__)`
- Allora stampa (solo) il valore di tale variabile

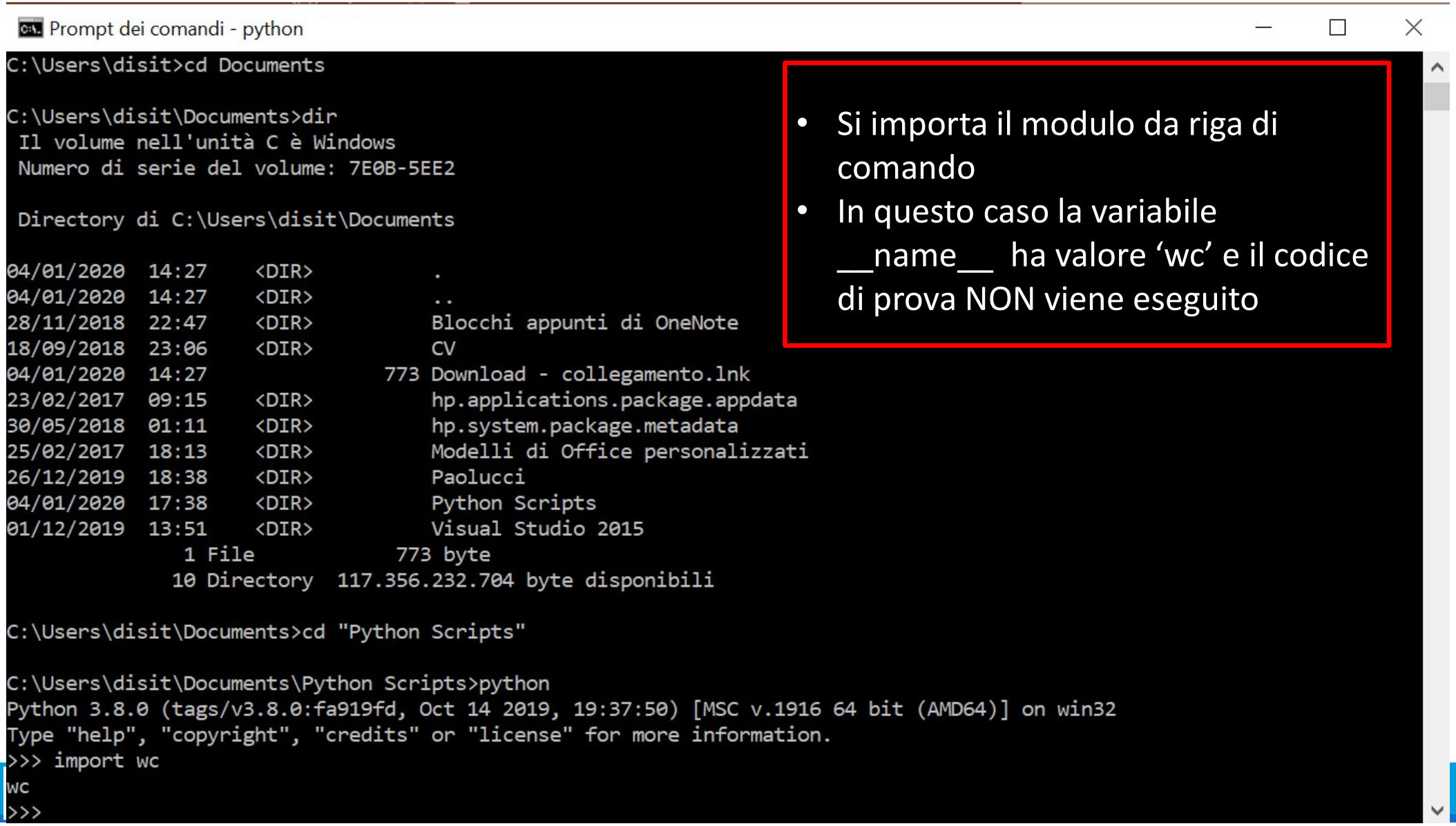
Che succede???? Che valore ha la variabile __name__ nei due casi seguenti??

- Caso 1: lancio wc.py direttamente
  - VALE: __main__
- Caso 2: importo wc
  - VALE: wc (variabile locale)
    NOTA: indica il programma principale!

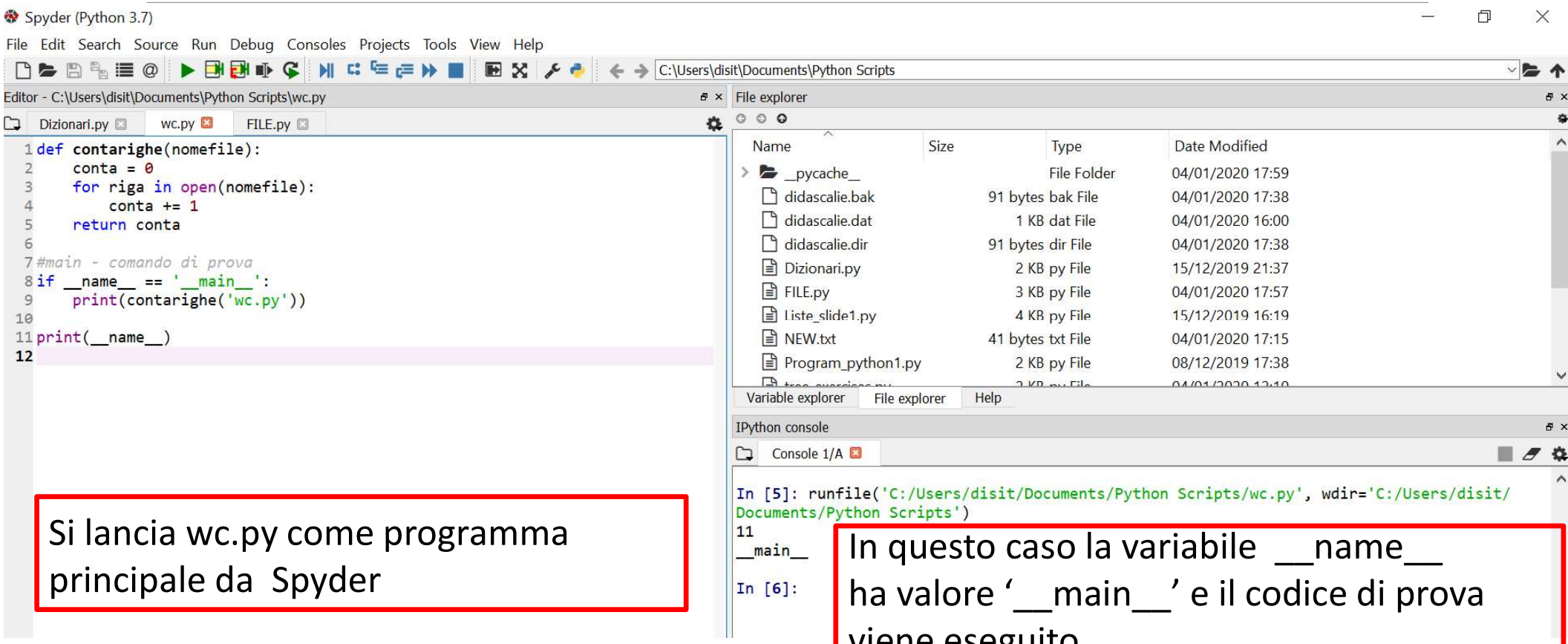
File: importare moduli (2)

- ESEMPIO 2:
  - Import direttamente
    - da riga di comando




Prompt dei comandi - python
C:\Users\disit>cd Documents
C:\Users\disit\Documents>dir
Il volume nell'unità C è Windows
Numero di serie del volume: 7E0B-5EE2
Directory di C:\Users\disit\Documents
04/01/2020 14:27 <DIR> .
04/01/2020 14:27 <DIR> ..
28/11/2018 22:47 <DIR> Blocchi appunti di OneNote
18/09/2018 23:06 <DIR> CV
04/01/2020 14:27 773 Download - collegamento.lnk
23/02/2017 09:15 <DIR> hp.applications.package.appdata
30/05/2018 01:11 <DIR> hp.system.package.metadata
25/02/2017 18:13 <DIR> Modelli di Office personalizzati
26/12/2019 18:38 <DIR> Paolucci
04/01/2020 17:38 <DIR> Python Scripts
01/12/2019 13:51 <DIR> Visual Studio 2015
1 File 773 byte
10 Directory 117.356.232.704 byte disponibili
C:\Users\disit\Documents>cd "Python Scripts"
C:\Users\disit\Documents\Python Scripts>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import wc
wc
>>>
• Si importa il modulo da riga di comando
• In questo caso la variabile __name__ ha valore ‘wc’ e il codice di prova NON viene eseguito

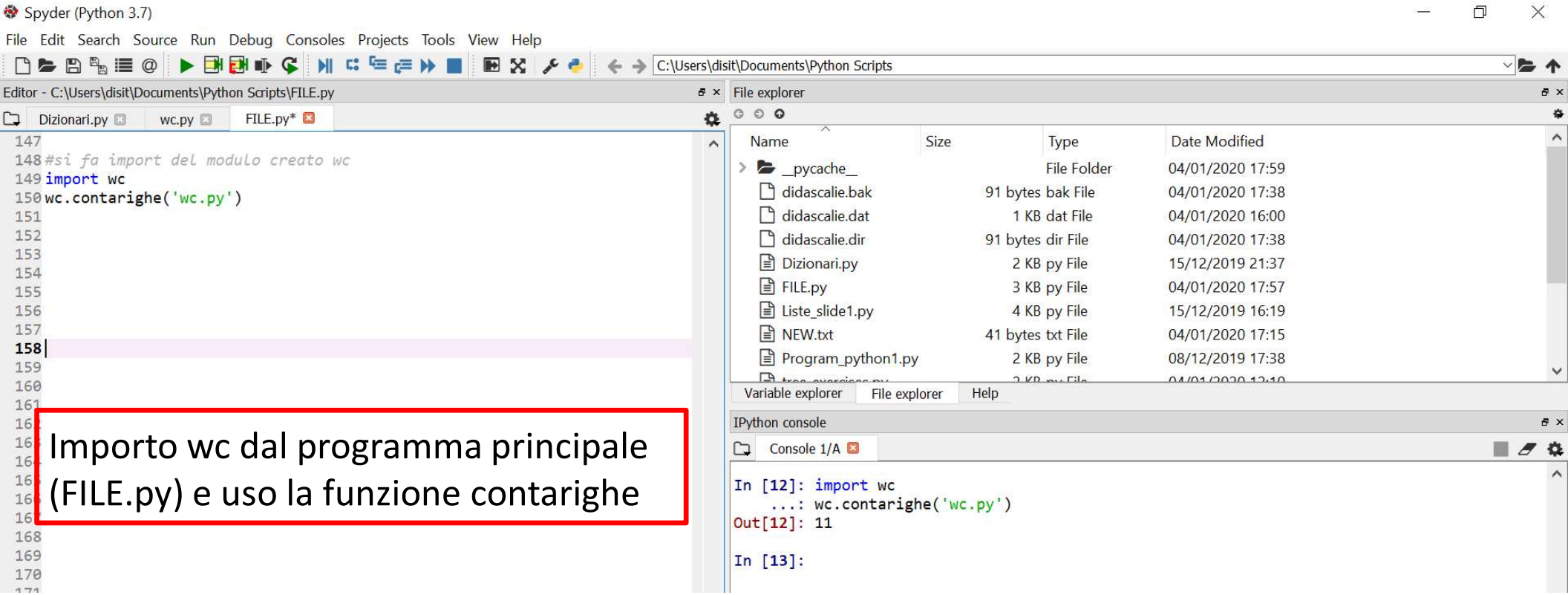
File: Importare moduli (3)



Si lancia wc.py come programma principale da Spyder

In questo caso la variabile __name__ ha valore '__main__' e il codice di prova viene eseguito

File: Importare moduli (4)



File: Importare moduli (5)

- Attenzione, se si vuole importare un modulo già importato, python NON fa nulla
- ALLORA è necessario:
  - Riavviare python
  - Oppure fare il reload del modulo (anche se potrebbe dare noie)

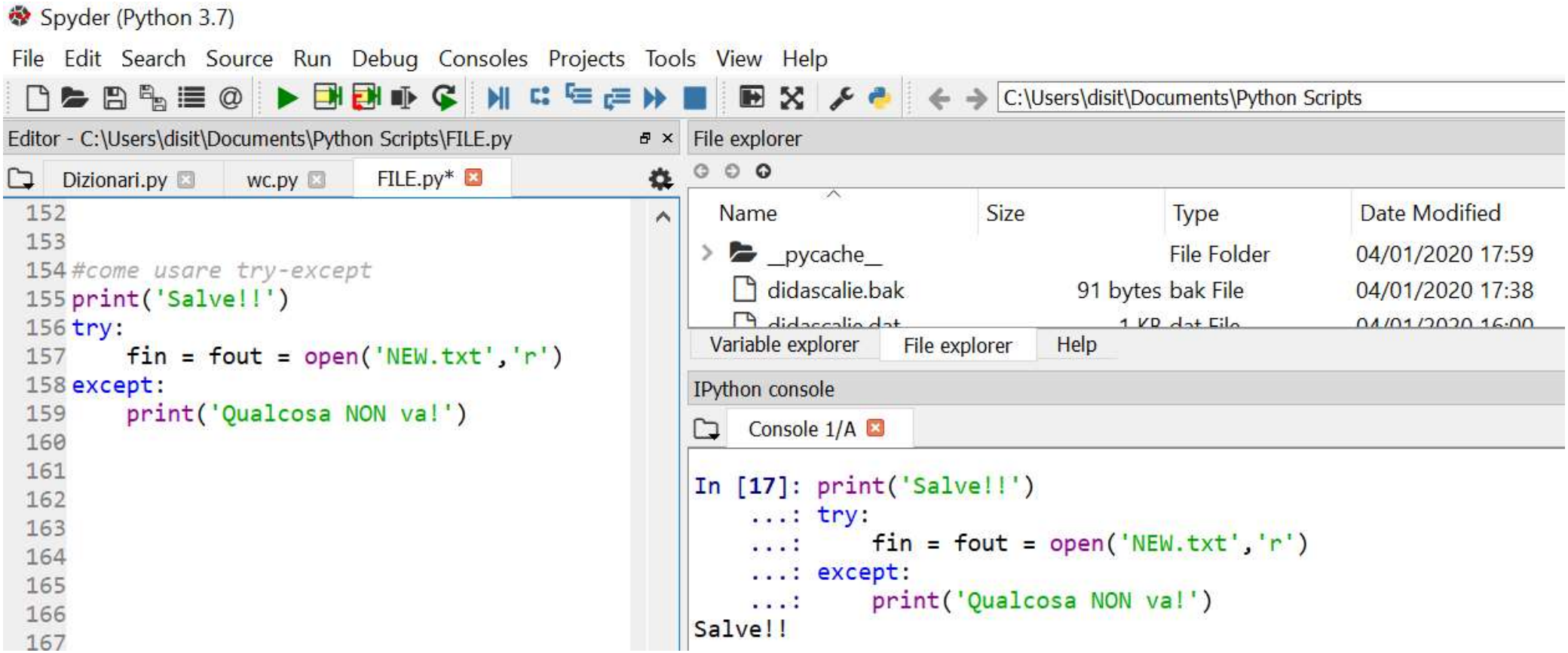
File: gestione delle eccezioni (1)

- Quando si vuole leggere/scrivere/modificare un file, ci sono molte cose che possono dare problemi
- Per questo motivo è sempre bene scrivere codice capace di gestire le eccezioni
- Tipologie di eccezioni:
  - Non si ha il permesso di accedere ad un file
  - Si sta cercando un file in una directory sbagliata
  - etc.
- Per evitare questi errori, si usano le funzioni (già viste) come os.path.exists e os.path.isfile
- Usarle però facendo uso del costrutto if-else potrebbe essere troppo dispendioso perché le possibilità di errore sono molte
- E' meglio affrontare i problemi quando si presentano e fare uso del costrutto: try-except

File: gestione delle eccezioni (2)

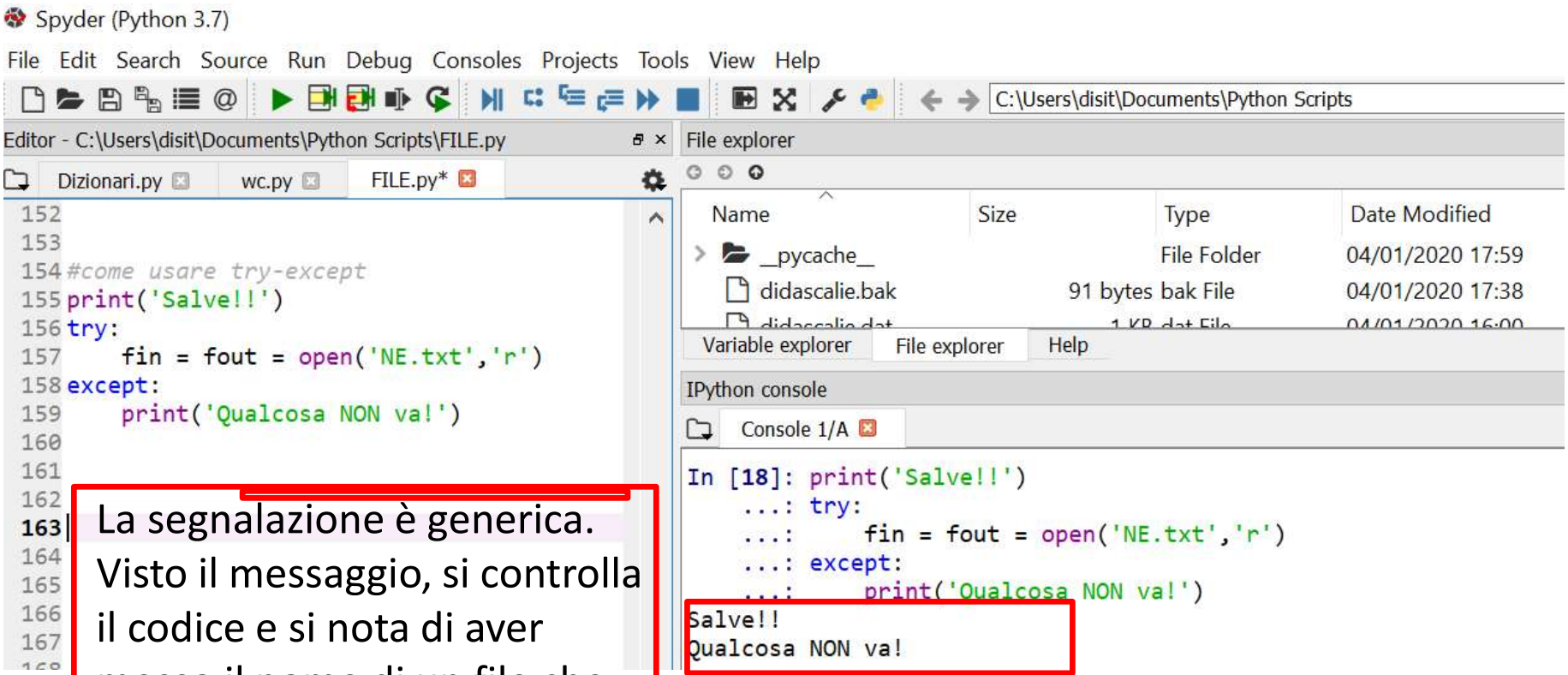
```
#come usare try-except
print('Salve!!')
try:
    fin = fout = open('NEW.txt','r')
except:
    print('Qualcosa NON va!')
```

File: gestione delle eccezioni (3)



File: gestione delle eccezioni (4)

- Segnalazione di un malfunzionamento (generico):



La segnalazione è generica. Visto il messaggio, si controlla il codice e si nota di aver messo il nome di un file che NON esiste!!!

Fare esercizio pag. 509 su Insiemi e Dizionari

Classi e Oggetti

Classi e Oggetti (1)

- Fino ad ora si sono visti concetti di:
  - Funzione: serve per organizzare il codice (tradurre in Python un algoritmo)
  - Tipi predefiniti: servono per organizzare i dati
- Adesso si procede con la programmazione orientata agli oggetti:
  - E' possibile usare tipi personalizzati per organizzare sia il codice che i dati
- Si crea un nuovo tipo: il tipo Punto
  - http://thinkpython2/code/Point1.py
  - http://thinkpython2/code/Point1_soln.py

Classi e Oggetti: tipi personalizzati (1)

- Nella notazione matematica, un punto è definito da una coppia ordinata di numeri: coordinate
- Le coordinate sono spesso scritte tra parentesi, con una virgola che separa i due valori:
  - (0,0) rappresenta l'origine
  - (x,y) rappresenta il punto che si trova a x unità a destra e y unità in alto rispetto all'origine
- Ci sono vari modi per rappresentare un punto in Python:
  - memorizzare le coordinate in due variabili separate
  - Memorizzare le coordinate come elementi di una tupla o di una lista
  - Creare un nuovo tipo che rappresenti i punti come oggetti

Classi e Oggetti: tipi personalizzati e concetto di Classe (1)

- Un tipo personalizzato in Python, ovvero definito dal programmatore, è detto classe
- Una definizione di classe ha la seguente sintassi:

```
class nome_classe:
    """Stringa di documentazione"""
```

  - ESEMPIO:

```
class Punto:
    """Rappresenta un punto in uno spazio 2-D. Attributi: x, y"""
```

- Supponiamo di avere inserito tale definizione nel file Punto.py, cosa succede se lo importiamo da riga di comando Python?

Classi e Oggetti: tipi personalizzati e concetto di Classe (2)

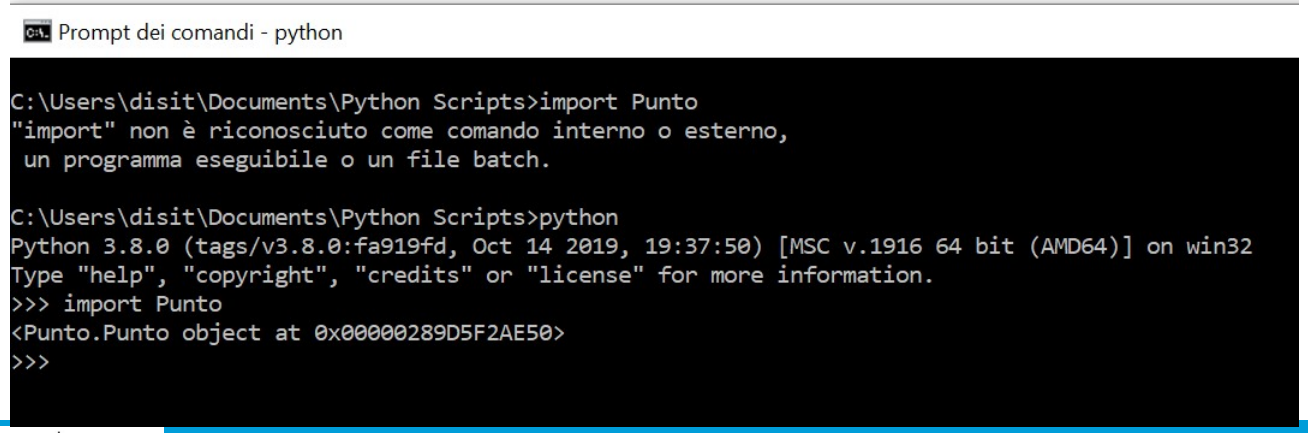
- Una volta effettuato l'import, il valore di ritorno è un riferimento ad un oggetto Punto
- Questo perché l'oggetto classe, è simile ad uno stampo: ci permette di 'fabbricare' una serie di oggetti tutti con le stesse caratteristiche

Prompt dei comandi - python

```
C:\Users\disit\Documents\Python Scripts>import Punto
"import" non è riconosciuto come comando interno o esterno,
 un programma eseguibile o un file batch.

C:\Users\disit\Documents\Python Scripts>python
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import Punto
<Punto.Punto object at 0x00000289D5F2AE50>
>>>
```

Classi e Oggetti: tipi personalizzati e concetto di Classe (3)

- La creazione di un nuovo oggetto, è detta: Istanziazione
- L'oggetto creato è una istanza della classe
- Quando 'si stampa' una istanza, Python informa/comunica:
  - a quale classe appartiene tale istanza
  - In quale posizione di memoria è collocata l'istanza
- Si ricorda che:
  - il prefisso 0x, indica che il numero che segue è nel formato esadecimale

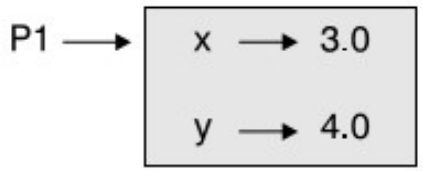


Concetto di Attributo (1)

- E' possibile assegnare dei valori ad una istanza:

```
P1.x = 3.0
P1.y = 4.0
```

- La sintassi è simile a quella usata per la selezione di una variabile appartenente ad un modulo
- In questo caso si stanno assegnando dei valori agli elementi di un oggetto
- Il diagramma di stato mostra il risultato delle assegnazioni e viene detto diagramma di oggetto



Concetto di Attributo (2)

- Supponendo di avere in una stessa directory due file:
  - Il file principale, con nome class.py
  - Il file in cui si definisce la classe, con nome Punto.py
- Allora si ha:

class.py

```
import Punto #Punto è il nome del modulo
P1 = Punto.Punto() #

P1.x = 3.0
P1.y = 4.0
print(P1)
print(P1.x)
print(P1.y)
```

Punto.py

```
#definizione di classe
#intestazione = nome della classe
class Punto:
"""Rappresenta un punto in uno spazio 2-D. Attributes: x, y"""
#il corpo della funzione è una stringa di documentazione
```

OUTPUT

```
<Punto.Punto object at 0x000002D821072108>
3.0
4.0
```

Concetto di Attributo (3)

- L'espressione P1.x significa: "vai all'oggetto puntato da P1 e ottieni il valore del suo attributo x"
- Attenzione, nelle righe seguenti:

```
P1 = Punto.Punto() #
P1.x = 3.0
x = P1.x
```

- x è una **variabile locale**, P1.x è l'**attributo di P1**
- La notazione a punto può essere usata all'interno di qualsiasi espressione:

```
print(P1.x)
```

Concetto di attributo (4)

```
import math
import Punto #Punto è il nome del modulo
P1 = Punto.Punto() #
P1.x = 3.0
P1.y = 4.0
print ('Sia P1 (' + str(P1.x) + ', ' + str(P1.y)+')' )
#distanza^2 dall'origine O(0,0)
DistanzaAlQuadrato = P1.x * P1.x + P1.y * P1.y
#espressione equivalente alla precedeente - elevazione a  potenza:
DistanzaAlQuadrato = P1.x **2 + P1.y **2
print('La distanza al quadrato dall\'origine e\' '+str(DistanzaAlQuadrato))
#usando la funzione matematica sqrt
distanza = math.sqrt(DistanzaAlQuadrato)
print('La distanza dall\'origine e\' '+str(distanza))
```

```
In [27]: runfile('C:/Users/disit/Documents/Python Scripts/class.py', wdir='C:/
Users/disit/Documents/Python Scripts')
Reloaded modules: Punto
<Punto.Punto object at 0x000002D82105C288>
3.0
4.0
3.0
Sia P1 (3.0, 4.0)
La distanza al quadrato dall'origine e' 25.0
La distanza dall'origine e' 5.0
```

Concetto di attributo (5)

Esercizio:

- Scrivere una funzione distanza_tra_punti che:
  - Prende in ingresso due punti
  - Calcola la distanza tra i due punti

Rettangoli (1)

- Supponiamo di voler progettare una classe capace di rappresentare un rettangolo:
  - Sarà necessario definire degli attributi per specificarne le dimensioni e la collocazione nel piano
  - Supponendo di NON considerare l'inclinazione sul piano
- Ci sono due possibili scelte:
  - Definire: centro del rettangolo oppure un angolo, le sue dimensioni
  - Definire: due angoli opposti
- Supponiamo di implementare la prima scelta

Rettangoli (2)

- Attributi:
  - larghezza, altezza, angolo.
  - Dove angolo è un oggetto Punto che indica l'angolo in basso a sinistra
- Per istanziare un nuovo oggetto rettangolo, è necessario istanziare un oggetto Rettangolo e assegnare dei valori ai suoi attributi

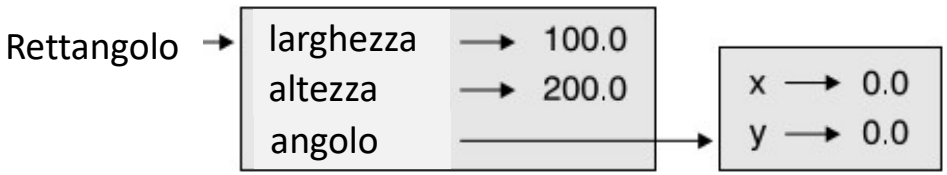
```
1   #definizione di classe
2   #intestazione = nome della classe
3   class Punto:
4       """Rappresenta un punto in uno spazio 2-D. Attributi:
        x, y"""
5   #il corpo della funzione è una stringa di documentazione
6
7   def stampa_punto(p):
8       """Stampa gli attributi: x, y"""
9       print('(%g, %g)' % (p.x, p.y))
10
11  class Rettangolo:
12      """Rappresenta un rettangolo in uno spazio 2-D.
        Attributi:
13      larghezza, altezza, angolo. Dove angolo è un oggetto
        Punto
14      che indica l'angolo in basso a sinistra"""
15  #il corpo della funzione è una stringa di documentazione
16
```

Rettangoli (3)

- L'espressione R1.angolo.x indica: 'vai all'oggetto a cui R1 fa riferimento e seleziona l'attributo angolo; poi vai a quell'oggetto e seleziona l'attributo chiamato x'

class.py

```
#Rettangoli
import Punto #Punto è il nome del modulo
R1 = Punto.Rettangolo() #
R1.larghezza = 100.0
R1.altezza = 200.0
R1.angolo = Punto.Punto()
R1.angolo.x = 0.0
R1.angolo.y = 0.0
```



Punto.py

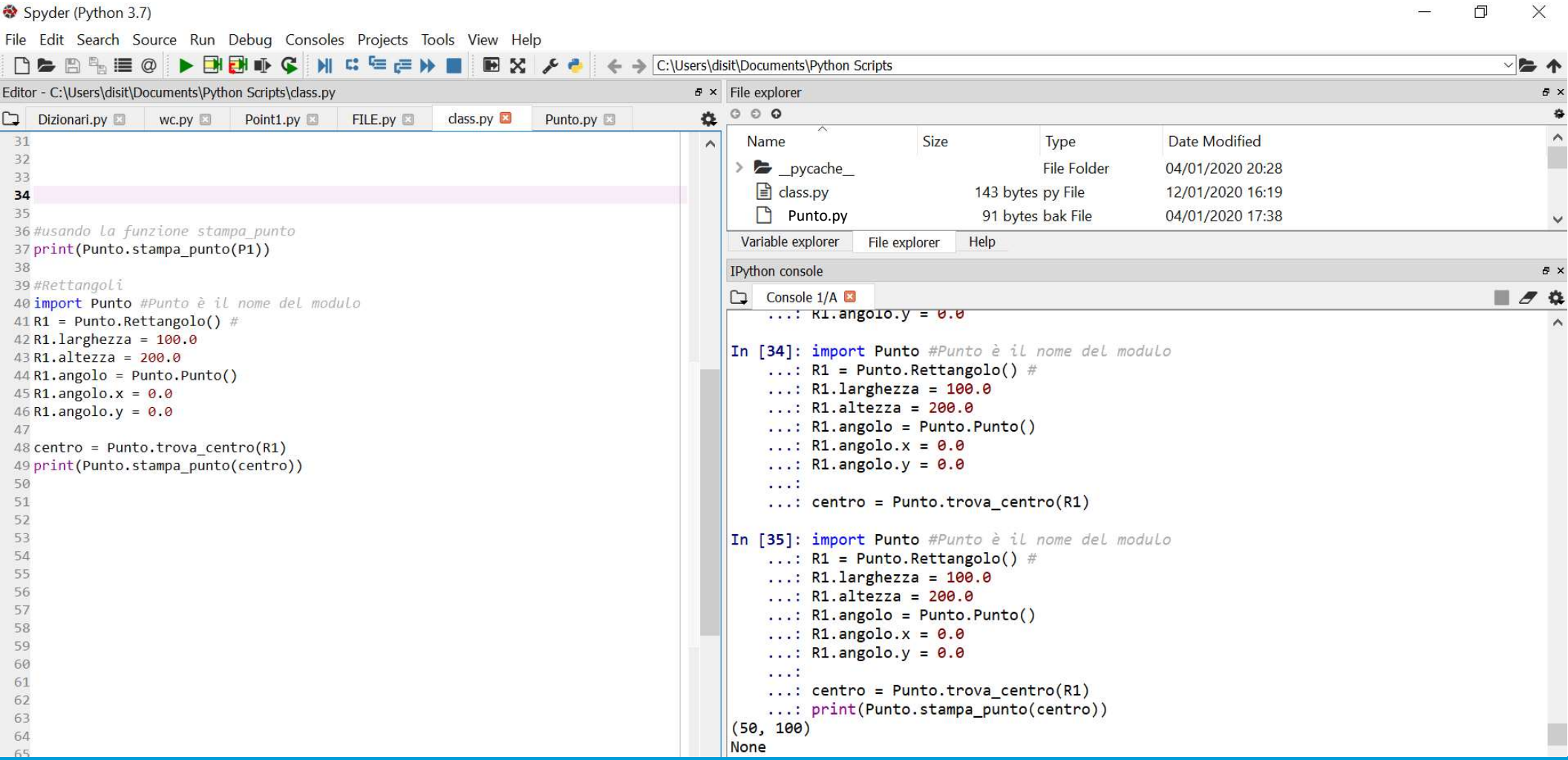
```
#definizione di classe
#intestazione = nome della classe
class Punto:
        """Rappresenta un punto in uno spazio 2-D. Attributi: x, y"""
#il corpo della funzione è una stringa di documentazione

def stampa_punto(p):
        """Stampa gli attributi: x, y"""
        print('(%g, %g)' % (p.x, p.y))


class Rettangolo:
        """Rappresenta un rettangolo in uno spazio 2-D. Attributi:
        larghezza, altezza, angolo. Dove angolo è un oggetto Punto
        che indica l'angolo in basso a sinistra"""
#il corpo della funzione è una stringa di documentazione
```

Istanze come valori di ritorno (1)

- Le funzioni possono restituire istanze
- Si definisce una funzione che prende in ingresso un rettangolo e rende il punto di centro:

```
def trova_centro(rett):
    p = Punto()
    p.x = rett.angolo.x + rett.larghezza/2
    p.y = rett.angolo.y + rett.altezza/2
    return(p)
```

- Dal main si deve richiamare la funzione:
  - `centro = Punto.trova_centro(R1)`


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\disit\Documents\Python Scripts
Editor - C:\Users\disit\Documents\Python Scripts\class.py
Dizionari.py wc.py Point1.py FILE.py class.py Punto.py
#usando la funzione stampa_punto
print(Punto.stampa_punto(P1))
#Rettangoli
import Punto #Punto è il nome del modulo
R1 = Punto.Rettangolo() #
R1.larghezza = 100.0
R1.altezza = 200.0
R1.angolo = Punto.Punto()
R1.angolo.x = 0.0
R1.angolo.y = 0.0
centro = Punto.trova_centro(R1)
print(Punto.stampa_punto(centro))
File explorer
Name Size Type Date Modified
__pycache__ File Folder 04/01/2020 20:28
class.py 143 bytes py File 12/01/2020 16:19
Punto.py 91 bytes bak File 04/01/2020 17:38
Variable explorer File explorer Help
IPython console
Console 1/A
In [34]: import Punto #Punto è il nome del modulo
...: R1 = Punto.Rettangolo() #
...: R1.larghezza = 100.0
...: R1.altezza = 200.0
...: R1.angolo = Punto.Punto()
...: R1.angolo.x = 0.0
...: R1.angolo.y = 0.0
...:
...: centro = Punto.trova_centro(R1)
In [35]: import Punto #Punto è il nome del modulo
...: R1 = Punto.Rettangolo() #
...: R1.larghezza = 100.0
...: R1.altezza = 200.0
...: R1.angolo = Punto.Punto()
...: R1.angolo.x = 0.0
...: R1.angolo.y = 0.0
...:
...: centro = Punto.trova_centro(R1)
...: print(Punto.stampa_punto(centro))
(50, 100)
None

Gli oggetti sono mutabili (1)

- E' possibile cambiare lo stato di un oggetto tramite l'operazione di assegnazione
- Quindi gli oggetti sono mutabili
- Per esempio, per cambiare le dimensioni di un rettangolo senza cambiarne la posizione, basta modificare altezza e larghezza:

```
R1.larghezza = R1.larghezza + 50.0
R1.altezza = R1.altezza + 100.0
```

- E' possibile anche scrivere funzioni che modificano oggetti:

```
def accresci_rettangolo(rett, dlargh, dalt):
    rett.larghezza += dlargh
    rett.altezza += dalt
```

Gli oggetti sono mutabili (2)

class.py

```
#Rettangoli
import Punto #Punto è il nome del modulo
R1 = Punto.Rettangolo() #
R1.larghezza = 100.0
R1.altezza = 200.0
R1.angolo = Punto.Punto()
R1.angolo.x = 0.0
R1.angolo.y = 0.0

centro = Punto.trova_centro(R1)
print(Punto.stampa_punto(centro))

Punto.accresci_rettangolo(R1, 50, 100)
centro2 = Punto.trova_centro(R1)
print(Punto.stampa_punto(centro2))
```

Punto.py

```
#definizione di classe
#intestazione = nome della classe
class Punto:
        """Rappresenta un punto in uno spazio 2-D. Attributi: x, y"""
#il corpo della funzione è una stringa di documentazione

def stampa_punto(p):
    """Stampa gli attributi: x, y"""
    print('(%g, %g)' % (p.x, p.y))

class Rettangolo:
        """Rappresenta un rettangolo in uno spazio 2-D. Attributi:
        larghezza, altezza, angolo. Dove angolo è un oggetto Punto
        che indica l'angolo in basso a sinistra"""
#il corpo della funzione è una stringa di documentazione

#funzione che prende in ingresso un rettangolo e rende il punto di centro
def trova_centro(rett):
    p = Punto()
    p.x = rett.angolo.x + rett.larghezza/2
    p.y = rett.angolo.y + rett.altezza/2
    return(p)

def accresci_rettangolo(rett, dlargh, dalt):
    rett.larghezza += dlargh
    rett.altezza += dalt
```

```
In [36]: import Punto #Punto è il nome del modulo
    ...: R1 = Punto.Rettangolo() #
    ...: R1.larghezza = 100.0
    ...: R1.altezza = 200.0
    ...: R1.angolo = Punto.Punto()
    ...: R1.angolo.x = 0.0
    ...: R1.angolo.y = 0.0
    ...:
    ...: centro = Punto.trova_centro(R1)
    ...: print(Punto.stampa_punto(centro))
    ...:
    ...: Punto.accresci_rettangolo(R1, 50, 100)
    ...: centro2 = Punto.trova_centro(R1)
    ...: print(Punto.stampa_punto(centro2))
(50, 100)
None
(75, 150)
None
```

OUTPUT

Copia (1)

- L'uso degli alias può rendere il programma meno leggibile, perché una modifica in un punto nel programma può generare effetti inattesi in un altro punto del programma
- La copia di un oggetto è spesso una buona alternativa all'alias
- Il modulo '**copy**' contiene una funzione (**copy**) che permette di duplicare qualsiasi oggetto

```
import Punto
import copy
P1 = Punto.Punto()
P1.x = 3.0
P1.y = 4.0
P2 = copy.copy(P1)
print(Punto.stampa_punto(P1))
print(Punto.stampa_punto(P2))
#uso dell'operatore == per dimostrare che P1 e P2 NON sono lo stesso oggetto
print(P1==P2)
```

- In questo caso P1 e P2, contengono gli stessi dati MA non sono lo stesso Punto

Copia (2)

Editor - C:\Users\disit\Documents\Python Scripts\class.py

Dizionari.py | wc.py | Point1.py | FILE.py | class.py* | Punto.py

```
54
55
56 import Punto
57 import copy
58 P1 = Punto.Punto()
59 P1.x = 3.0
60 P1.y = 4.0
61 P2 = copy.copy(P1)
62
63 print(Punto.stampa_punto(P1))
64 print(Punto.stampa_punto(P2))
65
66
67 #uso dell'operatore == per dimostrare che P1 e P2 NON sono lo stesso oggetto
68 print(P1==P2)
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
```

File explorer

Name	Size	Type	Date Modified
__pycache__		File Folder	04/01/2020 20:28
class.py	143 bytes	py File	12/01/2020 16:19
didascalie.bak	91 bytes	bak File	04/01/2020 17:38

Variable explorer | File explorer | Help

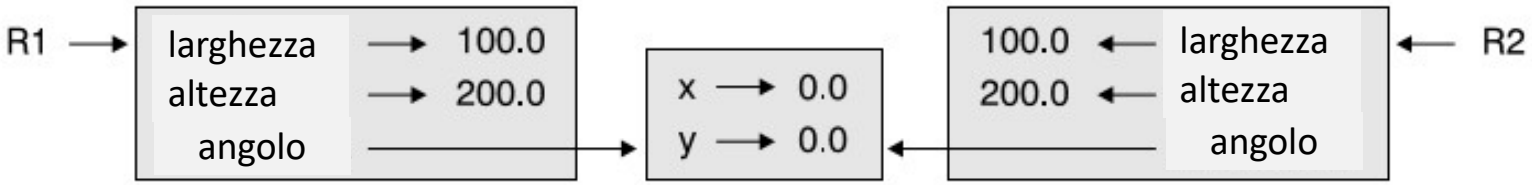
IPython console

Console 1/A

```
    ...: print(Punto.stampa_punto(P1))
    ...: print(Punto.stampa_punto(P2))
(3, 4)
None
(3, 4)
None

In [49]: import Punto
    ...: import copy
    ...: P1 = Punto.Punto()
    ...: P1.x = 3.0
    ...: P1.y = 4.0
    ...: P2 = copy.copy(P1)
    ...:
    ...: print(Punto.stampa_punto(P1))
    ...: print(Punto.stampa_punto(P2))
    ...:
    ...: print(P1==P2)
(3, 4)
None
(3, 4)
None
False
```

Copia (3)

- L'operatore '==' ha reso FALSE come risultato poiché se viene applicato a istanze, allora il comportamento predefinito dell'operatore '==' è lo stesso di quello dell'operatore *is*, ovvero: controlla l'identità dell'oggetto e NON l'equivalenza
- Questo perché per i tipi personalizzati Python non sa cosa debba essere considerato equivalente
- Si noti inoltre che:
  - Se si usa copy su un oggetto Rettangolo, viene fatta la copia solo del Rettangolo e NON del Punto angolo: originale e copia hanno lo stesso angolo



Copia (4)

```
#copia rettangolo
import Punto #Punto è il nome del modulo
R1 = Punto.Rettangolo() #
R1.larghezza = 100.0
R1.altezza = 200.0
R1.angolo = Punto.Punto()
R1.angolo.x = 0.0
R1.angolo.y = 0.0
R2 = copy.copy(R1)

print (R2 == R1) #rende FALSE perché i due oggetti R1 e R2 sono diversi
print (R2.angolo.x == R1.angolo.x) # Rende TRUE perché l'angolo è lo stesso
#equivalente alle seguenti righe
print (R2 is R1) #rende FALSE perché i due oggetti R1 e R2 sono diversi
print (R2.angolo.x is R1.angolo.x) # Rende TRUE perché l'angolo è lo stesso
```

Copia (5)

- L'operazione di copia appena descritta è chiamata copia shallow (o copia superficiale) perché copia l'oggetto e ogni riferimento in esso contenuto ma non gli oggetti contenuti
- Nella maggior parte dei casi questo comportamento NON è quello ottimale
- Ad esempio la chiamata delle seguenti funzioni avrebbe effetti diversi:
  - accresci_rettangolo(R1, 50, 100) -> influenza SOLO R1
  - sposta_rettangolo (R1, 0.7, 1.0) -> influenza sia R1 che R2

Con (nel file Punto.py):

```
def sposta_rettangolo(rett, new_x, new_y):
    rett.angolo.x = new_x
    rett.angolo.y = new_y
```

Copia (6)

Editor - C:\Users\disit\Documents\Python Scripts\class.py

Dizionari.py | wc.py | Point1.py | FILE.py | class.py* | Punto.py

```
93
94
95
96 #funzioni con impliacazioni diverse
97 import Punto #Punto è il nome del modulo
98 R1 = Punto.Rettangolo() #
99 R1.larghezza = 100.0
100 R1.altezza = 200.0
101 R1.angolo = Punto.Punto()
102 R1.angolo.x = 0.0
103 R1.angolo.y = 0.0
104 R2 = copy.copy(R1)
105 print("stampo i centri dei rettangoli prima di chiamare la funzione accresci_rettangolo su R1")
106 centroR1 = Punto.trova_centro(R1)
107 print('centro R1')
108 print(Punto.stampa_punto(centroR1))
109 print('centro R2')
110 centroR2 = Punto.trova_centro(R2)
111 print(Punto.stampa_punto(centroR2))
112 Punto.accresci_rettangolo(R1, 50, 100)
113 print("stampo i centri dei rettangoli dopo la chiamata alla funzione accresci_rettangolo su R1")
114 centroR1 = Punto.trova_centro(R1)
115 print('centro R1')
116 print(Punto.stampa_punto(centroR1))
117 print('centro R2')
118 centroR2 = Punto.trova_centro(R2)
119 print(Punto.stampa_punto(centroR2))
120
121 print("stampo angolo di R1 e di R2 prima di chiamare la funzione sposta_rettangolo su R1")
122 Punto.stampa_punto(R1.angolo)
123 Punto.stampa_punto(R2.angolo)
124
125 Punto.sposta_rettangolo(R1, 0.7, 1)
126 print("stampo angolo di R1 e di R2 dopo la chimata alla funzione sposta_rettangolo su R1")
127
128 Punto.stampa_punto(R1.angolo)
129 Punto.stampa_punto(R2.angolo)
130
131
132
133
```

File explorer

Name	Size	Type	Date Modified
__pycache__		File Folder	04/01/2020 20:28
class.py	143 bytes	py File	12/01/2020 16:19
didascalie.bak	91 bytes	bak File	04/01/2020 17:38

Variable explorer | File explorer | Help

IPython console

Console 1/A

```
stampo i centri dei rettangoli prima di chiamare la funzione accresci_rettangolo
su R1
centro R1
(50, 100)
None
centro R2
(50, 100)
None
stampo i centri dei rettangoli dopo la chiamata alla funzione accresci_rettangolo
su R1
centro R1
(75, 150)
None
centro R2
(50, 100)
None
stampo angolo di R1 e di R2 prima di chiamare la funzione sposta_rettangolo su R1
(0, 0)
(0, 0)
stampo angolo di R1 e di R2 dopo la chimata alla funzione sposta_rettangolo su R1
(0.7, 1)
(0.7, 1)

In [74]:
```

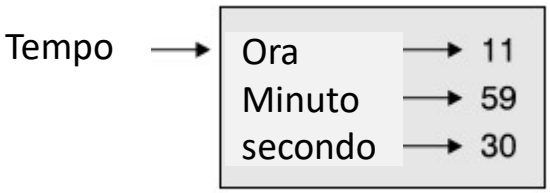
Copia (7)

- Si noti che è possibile anche scrivere main e definizioni di classi in un unico file
- Si vedano esempi:
  - http://www.thinkpython2.com/code/Point1.py
  - http://www.thinkpython2.com/code/Point1_soln.py

Classi e Funzioni - Tempo

- Oltre a poter scrivere nuovi tipi personalizzati in Pyhton, è possibile scrivere delle funzioni che prendano i tipi personalizzati come parametri e restituiscano dei risultati
- Si crea una classe Tempo, che permette di rappresentare un'ora del giorno

```
class Tempo:
"""Rappresenta un'ora del giorno.
attributi: ora, minuto, secondo
"""

def main():
    tempo = Tempo() #dal main si istanzia la funzione Tempo e si assegnano i
valori
    tempo.ora = 11
    tempo.minuto = 59
    tempo.secondo = 30

if __name__ == '__main__':
    main()
```



Esercizi

- Esercizio: scrivere una funzione StampaTempo che prende un oggetto Tempo come argomento e ne stampa il risultato nella classica forma ore:minuti:secondi (hh:mm:ss)

- Esercizio: scrivere una funzione booleana 'Dopo', che prende come argomenti due oggetti Tempo (Tempo1 e Tempo2) e ritorna:
  - vero se Tempo1 segue cronologicamente Tempo2
  - falso in caso contrario

Concetto di 'Funzione Pura'

- Ecco un prototipo della funzione somma_tempo:

```
def somma_tempo(T1, T2):
    somma = Tempo()
    somma.ora = T1.ora + T2.ora
    somma.minuti = T1.minuti + T2.minuti
    somma.secondi = T1.secondi + T2.secondi
    return somma
```

- La funzione:
  - crea un nuovo oggetto Tempo
  - inizializza i suoi attributi
  - ritorna un riferimento al nuovo oggetto
- Questa funzione viene chiamata **funzione pura** perché non modifica in alcun modo gli oggetti passati come suoi parametri e non ha effetti collaterali (del tipo richiedere l'immissione di un valore da parte dell'operatore o stampare un valore a video)

Concetto di ‘Funzione Pura’ (2)

- Per testare la funzione appena definita, creiamo due oggetti nel main: Tempo (ad esempio contenente l’ora di inizio di un film) e durata (la durata del film). Se si chiama la funzione somma_tempo, questa ci dirà a che ora finisce il film

```
def main():
    tempo = Tempo()
    tempo.ora = 11
    tempo.minuto = 59
    tempo.secondo = 30
    inizio = Tempo()
    inizio.ora = 8
    inizio.minuto = 45
    inizio.secondo = 0
    durata = Tempo()
    durata.ora = 1
    durata.minuto = 35
    durata.secondo = 0
    fine = somma_tempo(inizio, durata)
    stampa_tempo(fine)
```

```
CON: (stampa nella forma ‘hh:mm:ss’)

def stampa_tempo(T):
    st_ora = '%g:’
    st_minuto = '%g:’
    st_secondo = '%g’
    if (T.ora < 10):
    st_ora = '0%g:’
    if (T.minuto < 10):
    st_minuto = '0%g:’
    if (T.secondo < 10):
    st_secondo = '0%g’
    stampa = st_ora+ st_minuto + st_secondo
    print(stampa % (T.ora, T.minuto, T.secondo))
```

Come output il programma produce: **09:80:00**

Concetto di ‘Funzione Pura’ (3)

- Il risultato NON è soddisfacente (09:80:00)
- Questo perché la funzione di somma dei tempi non gestisce correttamente i casi in cui la somma dei minuti e dei secondi equivale o supera i sessanta
- Quando questo accade, è necessario riportare i 60 secondi in minuti, oppure i 60 minuti in ore
- Si riscrive allora la funzione stampa_tempo tenendo presente le precedenti osservazioni

Concetto di 'Funzione Pura' (4)

```
def somma_tempo(T1, T2):
    somma = Tempo()
    somma.ora = T1.ora + T2.ora
    somma.minuto = T1.minuto + T2.minuto
    somma.secondo = T1.secondo + T2.secondo
    if (somma.secondo >=60):
            somma.secondo -= 60 #equivale a somma.secondo = somma.secondo - 60
            somma.minuto += 1 #equivale a somma.minuto = somma.minuto +1
    if (somma.minuto >=60):
            somma.minuto -= 60 #equivale a somma.secondo = somma.secondo - 60
            somma.ora += 1 #equivale a somma.minuto = somma.minuto +1
    return somma
```

OUTPUT: **10:20:00**

- NOTA: in questo modo, la funzione è corretta è l'output sarà quello che ci si aspetta, tuttavia tale funzione è lunga -> vedremo allora una alternativa più concisa

Concetto di 'Modificatore' (1)

- Ci sono dei casi in cui è utile una funzione che possa modificare gli oggetti passati come suoi parametri. Quando questo si verifica, la funzione è detta modificatore
- La funzione *incremento* che somma un certo numero di secondi a Tempo può essere scritta in modo molto intuitivo come modificatore. La prima stesura potrebbe essere questa:

```
def incremento(tempo, secondi):
    tempo.secondo += secondi #operazione di somma
    #controlli successivi:
    if (tempo.secondo >= 60): #va bene anche -> if  tempo.secondo >= 60:
            tempo.secondo -= 60
            tempo.minuto += 1
    if tempo.minuto >= 60:
            tempo.minuto -= 60
            tempo.ora += 1
```

  - Questa funzione è corretta?
  - Succede se secondi è molto più grande di 60?

Concetto di 'Modificatore' (2)

- Proviamo a chiamare la funzione dal main:

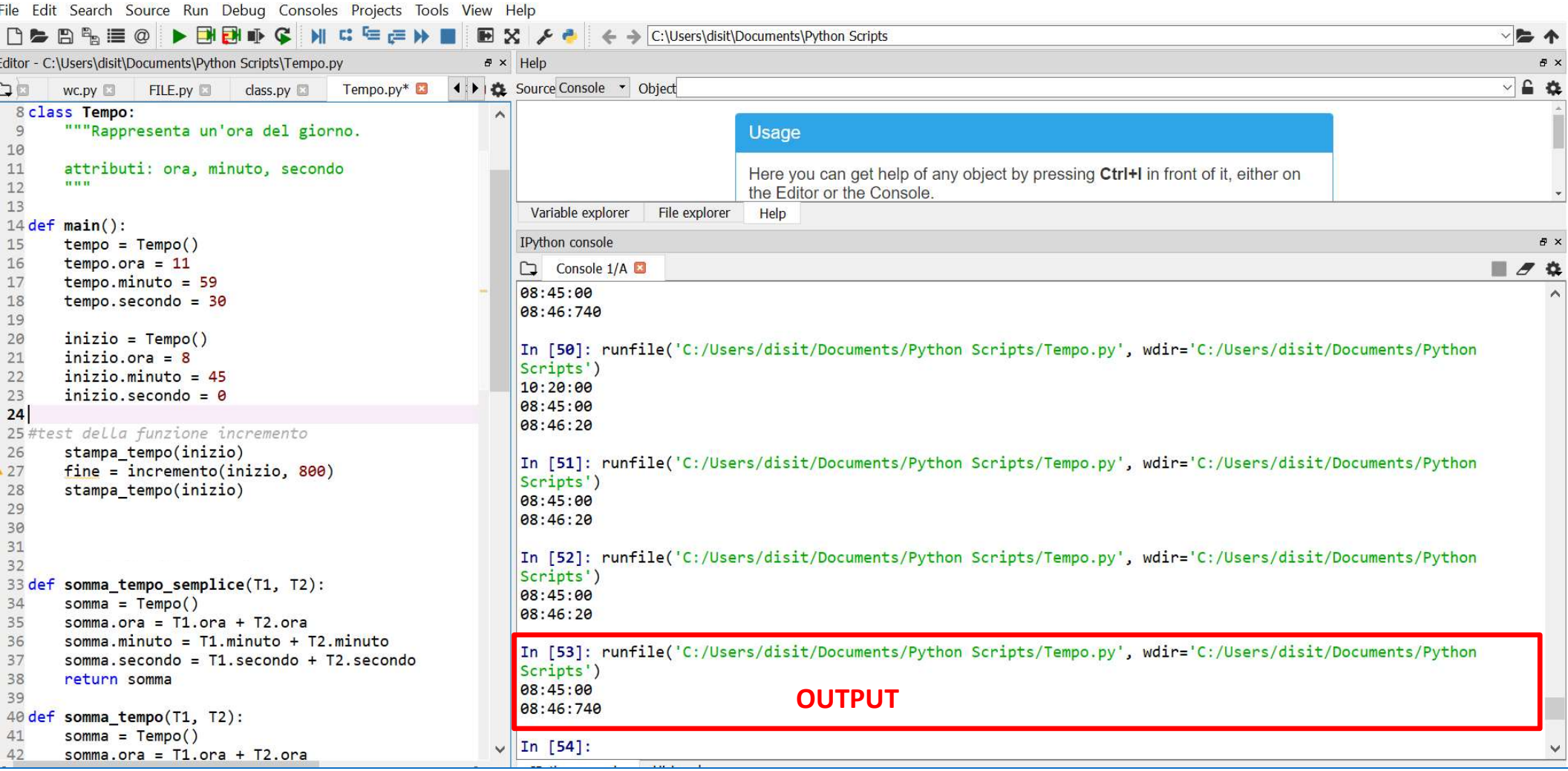
```
def main():
    tempo = Tempo()
    tempo.ora = 11
    tempo.minuto = 59
    tempo.secondo = 30
    inizio = Tempo()
    inizio.ora = 8
    inizio.minuto = 45
    inizio.secondo = 0
    #test della funzione incremento
    stampa_tempo(inizio)
    fine = incremento(inizio, 80)
    stampa_tempo(inizio)
```

```
In [52]: runfile('C:/Users/disit/Documents/Python
Scripts/Tempo.py', wdir='C:/Users/disit/Documents/
Python Scripts')
08:45:00
08:46:20
```

OUTPUT

- In questo caso il risultato è corretto!
- Se si inserisce pero' un numero di secondi molto più grande di 60?? Ad esempio fine = incremento(inizio, 800) -> Cosa Succede??

Concetto di ‘Modificatore’ (3)



Concetto di 'Modificatore' (4)

- Una possibile soluzione è quella di usare i cicli while per definire la funzione incremento:

```
def incremento(tempo, secondi):
    tempo.secondo += secondi #operazione di somma
    #controlli successivi:
    while (tempo.secondo >= 60):
            tempo.secondo -= 60
            tempo.minuto += 1
    while tempo.minuto >= 60:
            tempo.minuto -= 60
            tempo.ora += 1
```

```
In [52]: runfile('C:/Users/disit/Documents/Python
Scripts/Tempo.py', wdir='C:/Users/disit/Documents/
Python Scripts')
08:45:00
08:46:20
```

OUTPUT

- In questo modo l'output sarà corretto…. Lanciando la funzione con un numero di secondi molto alto si ottiene esattamente quello che ci aspettiamo

Concetto di 'Modificatore' (6)

- Proviamo a chiamare la funzione dal main:

```
def main():
    tempo = Tempo()
    tempo.ora = 11
    tempo.minuto = 59
    tempo.secondo = 30
    inizio = Tempo()
    inizio.ora = 8
    inizio.minuto = 45
    inizio.secondo = 0
    #test della funzione incremento
    stampa_tempo(inizio)
    fine = incremento(inizio, 800)
    stampa_tempo(inizio)
```

```
In [56]: runfile('C:/Users/disit/Documents/Python Scripts/
Tempo.py', wdir='C:/Users/disit/Documents/Python Scripts')
08:45:00
08:58:20
```

OUTPUT attuale corretto

```
In [53]: runfile('C:/Users/disit/Documents/Python Scripts/
Tempo.py', wdir='C:/Users/disit/Documents/Python Scripts')
08:45:00
08:46:740
```

OUTPUT NON corretto (funzione SENZA ciclo while)

- NOTA: La funzione è corretta MA NON EFFICIENTE
- Esercizio1: aggiornare il modificatore incremento in modo da rendere la funzione sia corretta che efficiente

Funzioni pure e Modificatori

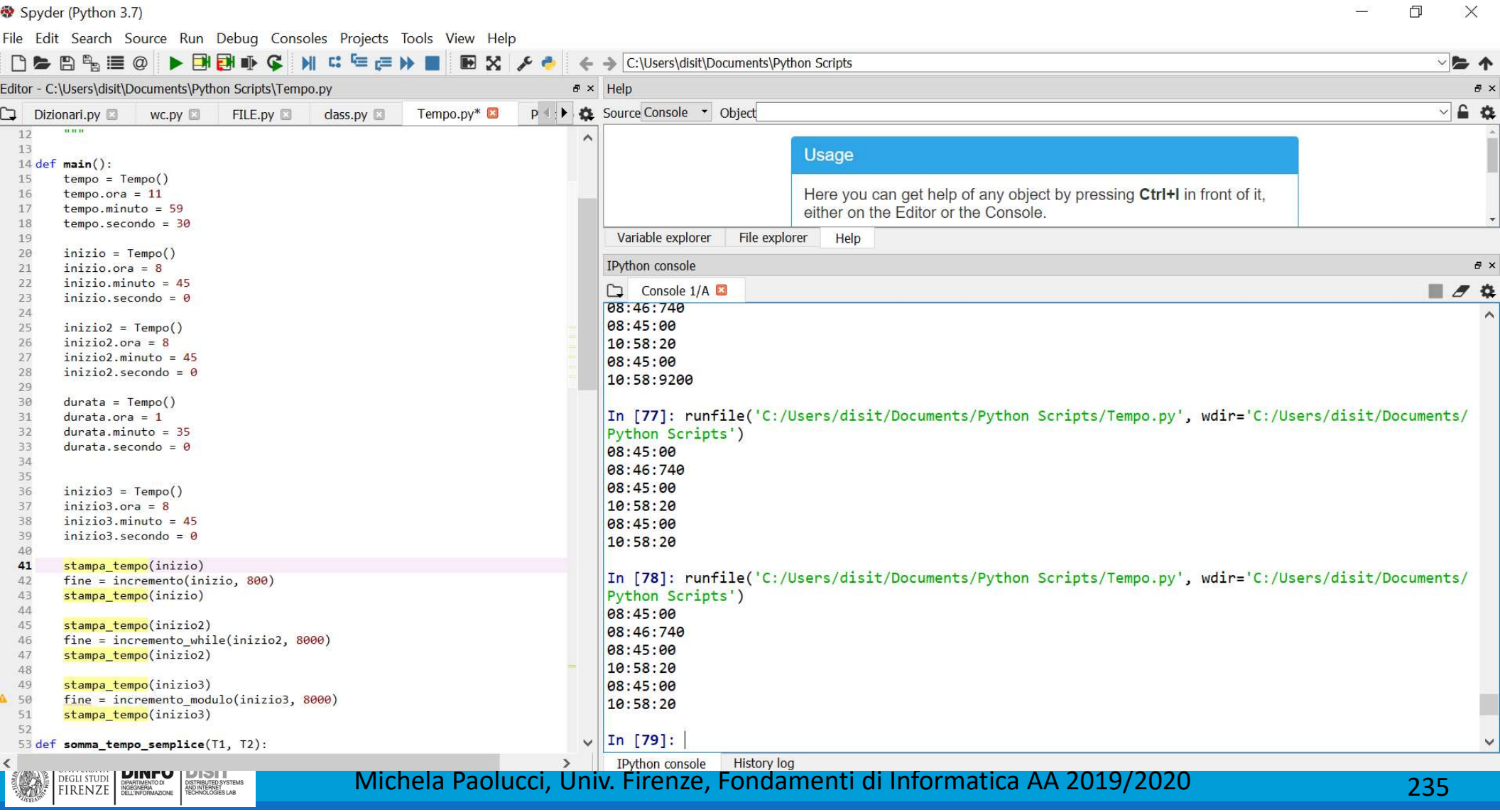
- Tutto quello che può essere fatto con i modificatori può anche essere fatta con le funzioni pure e alcuni linguaggi di programmazione non prevedono addirittura i modificatori.
- Si può affermare che le funzioni pure sono più veloci da sviluppare e portano ad un minor numero di errori, anche se in qualche caso può essere utile fare affidamento ai modificatori
- Suggerimento: usare le funzioni pure dove possibile, anziché ricorrere ai modificatori
- Esercizio2: Scrivere una funzione Pura di incremento che crei e restituisca un nuovo oggetto, anziché modificare il primo

Modificatore: soluzione esercizio1 (1)

```
#aggiornamento del modificatore
#esempio 8000secondi - > 2 ore, 53 minuti, 20 secondi

def incremento_modulo(tempo, secondi):
        if (secondi >= 60):
                incr_ora = int(secondi/(60*60) )
                incr_min = int((secondi/(60*60) - incr_ora)*60)
                incr_sec = int( ( ((secondi/(60*60) - incr_ora)*60)-incr_min)*60)
                tempo.minuto += incr_min
                tempo.ora += incr_ora
                tempo.secondo = incr_sec
```

Modificatore: soluzione esercizio1 (2)

```
#esempio 8000secondi - > 2 ore, 53 minuti, 20 secondi
def incremento_modulo(tempo, secondi):
        tempo.secondo += secondi #operazione di somma
        if (tempo.secondo >= 60):
                incr_ora = int(tempo.secondo/(60*60) )
                incr_min = int((tempo.secondo/(60*60) - incr_ora)*60)
                incr_sec = int( ( ((tempo.secondo/(60*60) - incr_ora)*60)-incr_min)*60)
        tempo.minuto += incr_min
        tempo.ora += incr_ora
        tempo.secondo = incr_sec
```


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\disit\Documents\Python Scripts
Editor - C:\Users\disit\Documents\Python Scripts\Tempo.py
Dizionari.py wc.py FILE.py class.py Tempo.py*
"""
def main():
tempo = Tempo()
tempo.ora = 11
tempo.minuto = 59
tempo.secondo = 30
inizio = Tempo()
inizio.ora = 8
inizio.minuto = 45
inizio.secondo = 0
inizio2 = Tempo()
inizio2.ora = 8
inizio2.minuto = 45
inizio2.secondo = 0
durata = Tempo()
durata.ora = 1
durata.minuto = 35
durata.secondo = 0
inizio3 = Tempo()
inizio3.ora = 8
inizio3.minuto = 45
inizio3.secondo = 0
stampa_tempo(inizio)
fine = incremento(inizio, 800)
stampa_tempo(inizio)
stampa_tempo(inizio2)
fine = incremento_while(inizio2, 8000)
stampa_tempo(inizio2)
stampa_tempo(inizio3)
fine = incremento_modulo(inizio3, 8000)
stampa_tempo(inizio3)
def somma_tempo_semplice(T1, T2):
Help
Source Console Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Variable explorer File explorer Help
IPython console
Console 1/A
08:46:740
08:45:00
10:58:20
08:45:00
10:58:9200
In [77]: runfile('C:/Users/disit/Documents/Python Scripts/Tempo.py', wdir='C:/Users/disit/Documents/Python Scripts')
08:45:00
08:46:740
08:45:00
10:58:20
08:45:00
10:58:20
In [78]: runfile('C:/Users/disit/Documents/Python Scripts/Tempo.py', wdir='C:/Users/disit/Documents/Python Scripts')
08:45:00
08:46:740
08:45:00
10:58:20
08:45:00
10:58:20
In [79]:
IPython console History log

Esercizio2

«Scrivere una funzione Pura di incremento che crei e restituisca un nuovo oggetto, anziché modificare il primo»

```
#esempio 8000secondi - > 2 ore, 53 minuti, 20 secondi
def incremento_modulo_pura(tempo, secondi):
        tempo_new = Tempo()
        tempo_new.ora = tempo.ora
        tempo_new.minuto = tempo.minuto
        tempo_new.secondo = tempo.secondo #operazione di somma
        #controlli successivi:
        if (secondi >= 60):
                incr_ora = int(secondi/(60*60) )
                incr_min = int((secondi/(60*60) - incr_ora)*60)
                incr_sec = int((((secondi/(60*60) - incr_ora)*60)-incr_min)*60)
                tempo_new.minuto += incr_min
                tempo_new.ora += incr_ora
                tempo_new.secondo = incr_sec
        return tempo_new
```

Sviluppo prototipale e sviluppo pianificato (1)

- La tecnica di sviluppo vista in questa definizione della funzione incremento si può definire per 'prototipo ed evoluzioni'
- Significa che per ogni funzione si inizia scrivendo una versione grezza (prototipo) che effettua solo i calcoli fondamentali, per poi testarla e modificarla correggendo gli eventuali errori
- Questo approccio nella fase iniziale di studio di un linguaggio di programmazione può essere utile, ma è necessario fare attenzione a non scrivere funzioni troppo complesse (ovvero troppe righe di codice) e poco affidabili

Sviluppo prototipale e sviluppo pianificato (2)

- Una alternativa allo sviluppo prototipale è lo sviluppo pianificato nel quale una conoscenza approfondita degli aspetti del problema da affrontare, rende la programmazione molto semplice
- Sapendo che l'oggetto Tempo è rappresentabile da un numero a tre cifre in base numerica 60, si può scrivere:

```
#funzione inversa che converte un intero in un Tempo
def tempo_in_int(tempo):
    minuti = tempo.ora *60 +tempo.minuto
    secondi = minuti *60 +tempo.secondo
    return secondi

#funzione che converte un intero in un tempo
#divmod divide il primo argomento per il secondo e rende
#una tupla contenente: quoziente e resto
def int_in_tempo(secondi):
    tempo = Tempo()
    minuti, tempo.secondo = divmod(secondi, 60)
    tempo.ora, tempo.minuto = divmod(minuti, 60)
    return tempo
```

- Queste funzioni possono essere puoi usate per scrivere in modo ottimale a funzione somma_tempo(t1,t2)

Sviluppo prototipale e sviluppo pianificato (3)

- Ecco la versione finale (concisa e più facilmente verificabile):

```
def somma_tempo_ottimizzata(tempo1, tempo2):
    secondi = tempo_in_int(tempo1) + tempo_in_int(tempo2)
    return int_in_tempo(secondi)
```

Nel main:

```
def main():
    inizio = Tempo()
    inizio.ora = 8
    inizio.minuto = 45
    inizio.secondo = 0
    durata = Tempo()
    durata.ora = 1
    durata.minuto = 35
    durata.secondo = 0

    fine = somma_tempo_ottimizzata(inizio, durata)
    stampa_tempo(fine)
```

- Scrivere per esercizio la funzione incremento usando le due funzioni `tempo_in_int(tempo1)` e `int_in_tempo(secondi)`

Soluzione ottimizzata

```
#funzione inversa che converte un intero in un Tempo
def tempo_in_int(tempo):
      minuti = tempo.ora *60 +tempo.minuto
      secondi = minuti *60 +tempo.secondo
      return secondi

#funzione che converte un intero in un tempo
#divmod divide il primo argomento per il secondo e rende
#una tupla contenente: quoziente e resto
def int_in_tempo(secondi):
      tempo = Tempo()
      minuti, tempo.secondo = divmod(secondi, 60)
      tempo.ora, tempo.minuto = divmod(minuti, 60)
      return tempo

def somma_tempo_ottimizzata(tempo1, tempo2):
      secondi = tempo_in_int(tempo1) + tempo_in_int(tempo2)
      return int_in_tempo(secondi)

def incremento_ottimizzato(tempo, secondi):
      return (int_in_tempo(tempo_in_int(tempo) + secondi))
```

DEFINIZIONI

```
def main():
    tempo = Tempo()
    tempo.ora = 8
    tempo.minuto = 45
    tempo.secondo = 0
    durata = Tempo()
    durata.ora = 1
    durata.minuto = 35
    durata.secondo = 0
    fine = somma_tempo_ottimizzata(inizio2, durata)
    stampa_tempo(fine)
    stampa_tempo (incremento_ottimizzato(tempo, 8000))
```

MAIN

```
In [99]: runfile('C:/Users/disit/Documents/Python Scripts/
Tempo.py', wdir='C:/Users/disit/Documents/Python Scripts')
10:20:00
10:58:20
```

OUTPUT

Generalizzazione (1)

- Sicuramente la conversione numerica da base 10 a base 60 e viceversa è meno intuitiva da capire, data la sua astrazione
- In un primo momento il lavoro è stato fatto in un modo molto più comprensibile anche se meno efficace
- Una volta progettato il programma facendo in modo di trattare i tempi come numeri in base 60, il tempo investito nello scrivere le funzioni di conversione viene abbondantemente recuperato quando riusciamo a scrivere un programma molto più corto, facile da leggere e correggere, e soprattutto più affidabile

Generalizzazione (2)

- Se il programma è progettato in modo oculato è più semplice aggiungere nuove caratteristiche
- Per esempio se si immagina di sottrarre due tempi per determinare l'intervallo trascorso
- L'approccio iniziale avrebbe portato alla necessità di dover implementare una sottrazione con il prestito
- Con le funzioni di conversione, scritte una sola volta ma riutilizzate in varie funzioni, è molto più facile e rapido avere un programma funzionante anche in questo caso
- Talvolta il fatto di rendere un problema più generale e quindi leggermente più difficile da implementare, permette di gestirlo in modo più semplice dato che ci sono meno casi speciali da gestire e di conseguenza minori possibilità di errore
- Quando si trova una soluzione ad una classe di problemi, invece che ad un singolo problema, si ha a che fare con un algoritmo

Classi e Metodi (1)

- Python è un linguaggio di programmazione orientato agli oggetti il che significa che fornisce il supporto alla programmazione orientata agli oggetti
- In realtà i programmi visti fino ad ora non sono del tutto orientati agli oggetti perché non mettono in evidenza le relazioni che esistono tra i tipi personalizzati e le funzioni che operano su di essi
- Il passo successivo è quello di trasformare tali funzioni in metodi, in modo da rendere esplicite queste relazioni

Classi e Metodi – funzionalità orientate agli oggetti

- Quando si dice che Python è un linguaggio di programmazione orientato agli oggetti il che significa che:
  - I programmi includono definizioni di classi e metodi
  - Buona parte della programmazione è espressa in termini di operazioni sugli oggetti
  - Gli oggetti corrispondono spesso ad un oggetto o concetto del mondo reale, mentre i metodi che operano sugli oggetti corrispondono spesso al modo in cui gli oggetti interagiscono tra di loro nella realtà quotidiana

Classi e Metodi – funzionalità orientate agli oggetti (2)

- Tempo:
  - la classe Tempo, corrisponde al modo in cui le persone pensano alle ore del giorno e Le funzioni definite invece corrispondo al tipo di operazioni che le persone possono effettuare sul tempo
- Punto e Rettangolo:
  - Le classi Punto e Rettangolo corrispondono ai relativi concetti matematici, le funzioni definite alle operazioni che si possono effettuare su di essi
- Fino ad ora non si è sfruttata la relazione tra classe e funzione
- In molti casi è utile per scrivere i programmi in modo più conciso e chiaro

Classi e Metodi – funzionalità orientate agli oggetti (3)

- Ad esempio nel caso del Tempo, quando si sono definite le funzioni non è stata specificata la relazione con la classe.
- Facendo maggiore attenzione, appare evidente che tutte le funzioni hanno in comune il fatto di ricevere in ingresso (come argomento) una classe di tipo Tempo
- Questa osservazione giustifica l'esistenza dei Metodi
- Un metodo è una funzione associata ad una particolare classe
- Abbiamo già visto i metodi predefiniti di Python, ad esempio quelli associati a: stringhe, liste, dizionari e tuple
- In questo caso si tratta di definire classi e metodi personalizzati (relativi alle classi definite)

Classi e Metodi – funzionalità orientate agli oggetti (4)

- Da un punto di vista logico, i metodi sono la stessa cosa delle funzioni MA con due differenze sintattiche:
  - I metodi sono definiti all'interno di una definizione di classe, per rendere più esplicita la relazione tra classe e metodo
  - La sintassi per invocare un metodo è diversa da quella usata per chiamare una funzione
- La trasformazione delle funzioni scritte in precedenza in metodi è puramente meccanica e si ottiene:
  - Spostando la definizione della funzione all'interno della definizione della classe

Classi e Metodi – funzionalità orientate agli oggetti (5)

- Ad esempio stampa_tempo :

```
class Tempo:
    """Rappresenta un'ora del giorno.
    attributi: ora, minuto, secondo
    """
    def stampa_tempo(T):
        st_ora = '%g:'
        st_minuto = '%g:'
        st_secondo = '%g'
        if (T.ora < 10):
            st_ora = '0%g:'
        if (T.minuto < 10):
            st_minuto = '0%g:'
        if (T.secondo < 10):
            st_secondo = '0%g'
        stampa = st_ora+ st_minuto + st_secondo
        print(stampa % (T.ora, T.minuto, T.secondo))
```

Chiamare metodi (1)

- Una volta definiti i metodi di una classe, sarà necessario chiamarli. Esistono due modalità:
  - (meno usato)
  - Nome_classe.nome_metodo(argomenti del metodo)
  - (più usato)
  - Istanza_classe.nome_metodo(argomenti del meteodo, tenendo presente che l'istanza è già stata 'passata')

Chiamare metodi (2)

```
def main():
    tempo = Tempo()
    tempo.ora = 8
    tempo.minuto = 45
    tempo.secondo = 0

    inizio = Tempo()
    inizio.ora = 8
    inizio.minuto = 45
    inizio.secondo = 0
    durata = Tempo()
    durata.ora = 1
    durata.minuto = 35
    durata.secondo = 0

    #primo modo meno usato per chiamare i metodi
    Tempo.stampa_tempo(inizio)
    fine = Tempo.incremento(inizio, 800)
    Tempo.stampa_tempo(inizio)

    #secondo modo, più usato pee chiamare i metodi
    fine = inizio.somma_tempo_ottimizzata(durata)
    tempo.stampa_tempo()
    tempo.incremento_ottimizzato(8000).stampa_tempo()
```

Il metodo speciale init

- Il metodo speciale init viene invocato quanto un oggetto viene istanziato
- Il suo nome completo è __init__:

```
def __init__(self, ora=0, minuto=0, secondo =0):
    self.ora = ora
    self.minuto = minuto
    self.secondo = secondo
```

- Solitamente i parametri di init hanno gli stessi nomi degli attributi
  - `self.ora = ora`
- Quando si chiama una funzione, ad esempio da main, i paremetri sono opzionali:

```
tempo_default = Tempo()
tempo_default.stampa_tempo()
```

  - ottiene in output

```
In [118]: runfile('C:/Users/disit/Documents/Python Scripts/
Tempo_Metodi.py', wdir='C:/Users/disit/Documents/Python
Scripts')
00:00:00
```

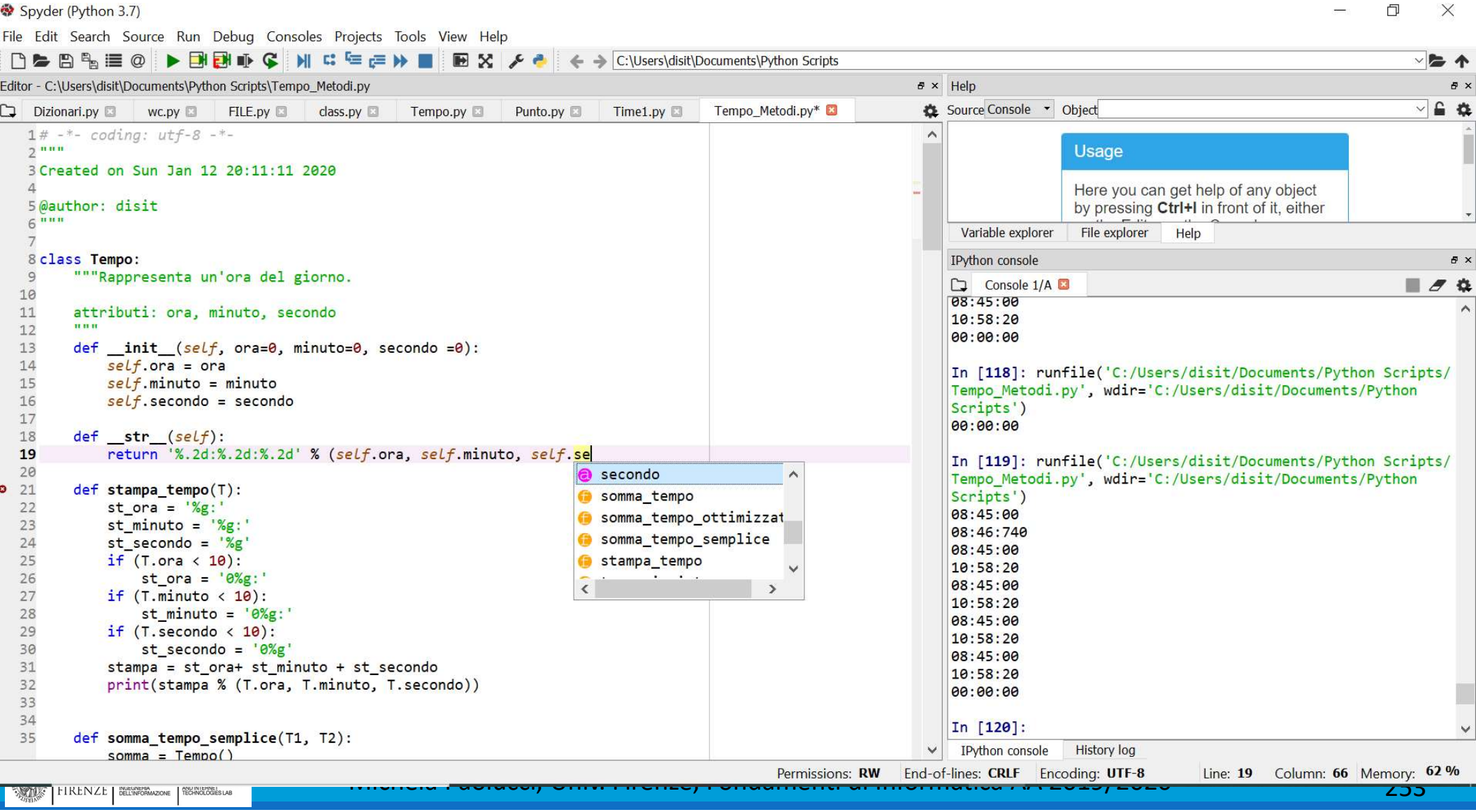
Metodo speciale __str__

- Il metodo speciale __str__ ha come scopo quello di restituire una rappresentazione di un oggetto in termini di stringa:

```
def __str__(self):
    return '%.2d:%.2d:%.2d' % (self.ora, self.minuto, self.secondo)
```

- Da main le chiamate risultano:

```
def main():
    tempo_default = Tempo(9,45,0)
    print(tempo_default)
```


Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\disit\Documents\Python Scripts
Editor - C:\Users\disit\Documents\Python Scripts\Tempo_Metodi.py
Dizionari.py wc.py FILE.py class.py Tempo.py Punto.py Time1.py Tempo_Metodi.py*
-*- coding: utf-8 -*-
"""
Created on Sun Jan 12 20:11:11 2020
@author: disit
"""
class Tempo:
"""Rappresenta un'ora del giorno.
attributi: ora, minuto, secondo
"""
def __init__(self, ora=0, minuto=0, secondo =0):
self.ora = ora
self.minuto = minuto
self.secondo = secondo
def __str__(self):
return '%.2d:%.2d:%.2d' % (self.ora, self.minuto, self.se
def stampa_tempo(T):
st_ora = '%g:'
st_minuto = '%g:'
st_secondo = '%g'
if (T.ora < 10):
st_ora = '0%g:'
if (T.minuto < 10):
st_minuto = '0%g:'
if (T.secondo < 10):
st_secondo = '0%g'
stampa = st_ora+ st_minuto + st_secondo
print(stampa % (T.ora, T.minuto, T.secondo))
def somma_tempo_semplice(T1, T2):
somma = Tempo()
secondo
somma_tempo
somma_tempo_ottimizzat
somma_tempo_semplice
stampa_tempo
Help
Source Console Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either
Variable explorer File explorer Help
IPython console
Console 1/A
08:45:00
10:58:20
00:00:00
In [118]: runfile('C:/Users/disit/Documents/Python Scripts/Tempo_Metodi.py', wdir='C:/Users/disit/Documents/Python Scripts')
00:00:00
In [119]: runfile('C:/Users/disit/Documents/Python Scripts/Tempo_Metodi.py', wdir='C:/Users/disit/Documents/Python Scripts')
08:45:00
08:46:740
08:45:00
10:58:20
08:45:00
10:58:20
08:45:00
10:58:20
08:45:00
10:58:20
00:00:00
In [120]:
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 19 Column: 66 Memory: 62 %

Overloading pag 170

Non fatto da pag. 170 a pagina 173

Liste linkate - How_To_Think_ITA cap.17 , pag.177

Le liste linkate fanno uso dei riferimenti interni

Le liste linkate sono costituite da nodi:

- ognuno di questi nodi contiene:
  - il riferimento al nodo successivo della lista
  - un'unità di dati utili chiamata contenuto

Una lista linkata è considerata una struttura di dati ricorsiva perché la sua definizione è di per se' ricorsiva

Una lista linkata è:

- una lista vuota, rappresentata da None,
- Oppure un nodo che contiene un oggetto "contenuto" ed un riferimento ad una lista linkata.

Le strutture di dati di tipo ricorsivo sono gestite da metodi ricorsivi

Liste Linkate

Lista linkata: Classe Nodo (1)

- Inizializzazione:
  - Utile per testare la creazione del tipo Lista
- metodo str:
  - Utile per testare la visualizzazione del nuovo tipo Lista

```
class Nodo:
      def __init__(self, value=None, next=None):
                  self.value = value
                  self.next = next

      def __str__(self):
                  return str(self.value)
```

- I parametri di ingresso per il metodo di inizializzazione sono definiti come opzionali. I due valori di input di default hanno valore 'None'
- La rappresentazione a stringa del Nodo è solo la stampa del suo contenuto:
  - alla funzione str può essere passato qualsiasi tipo di valore (value)
  - Ovvero è possibile memorizzare nella lista ogni tipo di dato

File Nodo.py per testare creazione e stampa

```
class Nodo:
        def __init__(self, value=None, next=None):
                self.value = value
                self.next = next
        def __str__(self):
                return str(self.value)


def main():
        Nodo1 = Nodo("test")
        print(Nodo1)

if __name__ == '__main__':
main()
```

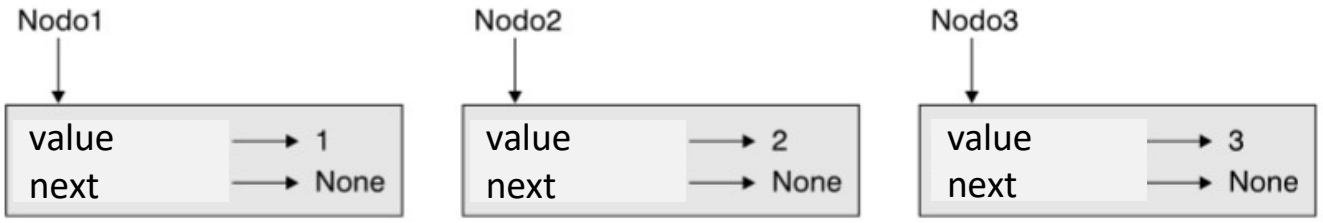
Lista linkata: Classe Nodo (2)

- Definiamo e stampiamo tre nodi:

```
def main():
    Nodo1 = Nodo(1)
    Nodo2 = Nodo(2)
    Nodo3 = Nodo(3)
    print(Nodo1)
    print(Nodo2)
    print(Nodo3)
```

```
In [9]: runfile('C:/Users/disit/Documents/Python
Scripts/Nodo_lista.py', wdir='C:/Users/disit/
Documents/Python Scripts')
1
2
3
```

- I tre nodi sono indipendenti:

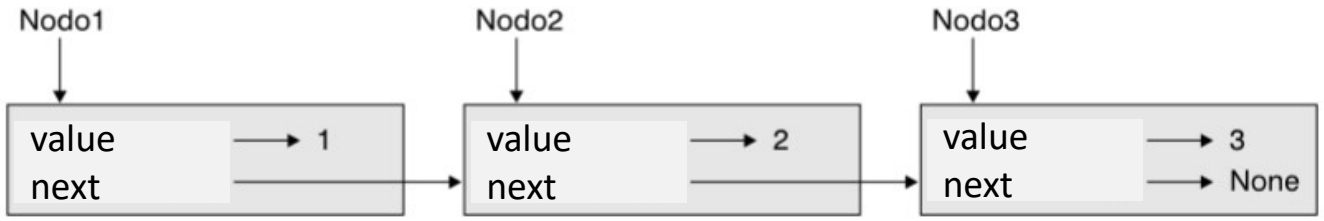


Lista linkata: Classe Nodo (3)

- Per linkare i nodi, è necessario fare in modo che il primo si riferisca al secondo, ed il secondo al terzo:

```
Nodo1.next = Nodo2
Nodo2.next = Nodo3
```

- Il riferimento del terzo nodo (il 'next') ha valore 'None' e questo indica che ci troviamo alla fine della lista.
- Il diagramma di stato risulta:



Liste come collezioni (1)

- Le liste sono utili perché forniscono un modo per assemblare più oggetti in una entità singola, chiamata ***collezione***
- Nell'esempio visto, il primo nodo serve come riferimento all'intera lista, ovvero ne rappresenta il punto di partenza
- Per passare una lista di questo tipo come parametro ad una funzione, basta passare il riferimento al suo primo nodo
- Ad esempio, si definisca la funzione StampaLista in modo che:
  - prenda come input un singolo nodo (argomento), considerandolo l'inizio della lista
  - e stampi il contenuto di ogni nodo finché non viene raggiunta la fine della lista

Liste come collezioni (2)

- Definizione della funzione StampaLista:

```
def StampaLista(Nodo):
    while(Nodo):
            print (Nodo)
            Nodo = Nodo.next
```

DEFINIZIONE

```
class Nodo:
      def __init__(self, value=None, next=None):
            self.value = value
            self.next = next
      def __str__(self):
            return str(self.value)
```

- La funzione StampaLista ha come input il riferimento al primo nodo della lista
- Come si raggiungono gli altri nodi?
- Per passare da un nodo al successivo si usa il valore Nodo.next usando la variabile Nodo per riferirsi ad ognuno dei nodi in successione
- Nel ciclo while quindi:
  - **Prima si stampa il valore del nodo (value) usando il metodo str (la cui chiamata è print(Nodo))**
  - **Poi si aggiorna il next, ovvero si visita il nodo successivo**

Liste come collezioni (3)

- Definizione della funzione StampaLista:

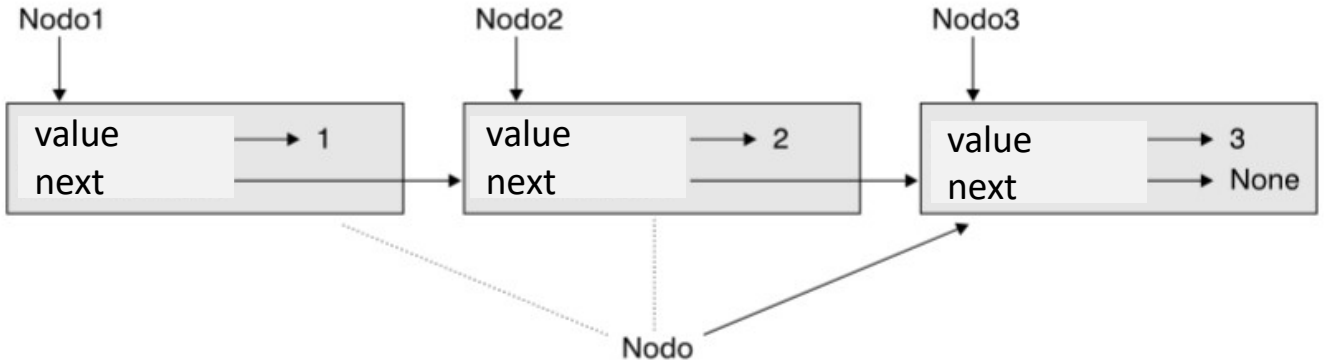
```
def StampaLista(Nodo):
    while(Nodo):
            print (Nodo)
            Nodo = Nodo.next
```

- Chiamata della funzione da main():
  - `Nodo1.StampaLista()`

DEFINIZIONE

```
class Nodo:
    def __init__(self, value=None, next=None):
        self.value = value
        self.next = next
    def __str__(self):
        return str(self.value)
```

```
In [18]: runfile('C:/Users/disit/Documents/Python
Scripts/Nodo_lista.py', wdir='C:/Users/disit/
Documents/Python Scripts')
```



- Il diagramma mostra il valore di Lista ed il valore assunto da Nodo alle varie iterazioni

Liste come collezioni (4)

Esercizio:

- Esercizio: per convenzione le liste sono stampate tra parentesi quadrate con virgole che ne separano gli elementi, come in [1, 2, 3]. Modificare la funzione StampaLista in modo che sia in grado di generare una stampa in tale formato

Liste e ricorsione

- Data la sua natura ricorsiva è intuitivo esprimere molte operazioni sulle liste con metodi ricorsivi
- Quello che segue è un algoritmo per stampare una lista a partire dall'ultimo elemento:
  - Separa la lista in due parti: il primo nodo (chiamato testa) ed il resto (la coda)
  - Stampa la coda in ordine inverso
  - Stampa la testa
- Logicamente il passo 2, la chiamata ricorsiva, parte dal presupposto che ci sia un metodo per stampare la lista al contrario
- Se partiamo dal presupposto che la chiamata ricorsiva funzioni correttamente, allora l'algoritmo lavora in modo corretto

Liste e ricorsione (1)

- E' necessario partire da un caso base per poi verificare che per ogni tipo di lista sia possibile ricondursi al caso base per interrompere la serie di chiamate ricorsive
- Partiamo da un esempio di ricorsione:

```
def scala(n):
    if(n == 0): return
    print("numero: ", n)
    scala(n-1)

#equivalente di:
def scala1(n):
   if n == 0:
     return
   else:
      print("numero: ", n)
      scala(n-1)
```

OUTPUT

```
numero:  10
numero:  9
numero:  8
numero:  7
numero:  6
numero:  5
numero:  4
numero:  3
numero:  2
numero:  1
```

- Chiamata da main():
  - scala(10)

Liste e ricorsione (2)

- E' necessario partire da un caso base per poi verificare che per ogni tipo di lista sia possibile ricondursi al caso base per interrompere la serie di chiamate ricorsive:

```
def StampaInversa(Lista):
    if(Lista.next == None):#condizione sul primo nodo
            print(Lista)
            return
    Testa = Nodo() #creo l'istanza Testa di tipo Nodo
    Coda = Nodo() #creo l'istanza Coda di tipo Nodo
    Testa = Lista #Testa fa riferimento a Lista
    Coda = Lista.next #Coda fa riferimento al next di Lista
    Coda.StampaInversa() #richiamo la funzione stessa nella Coda
    print(Testa) #alla fine stampo la testa
```

- Chiamata da main():

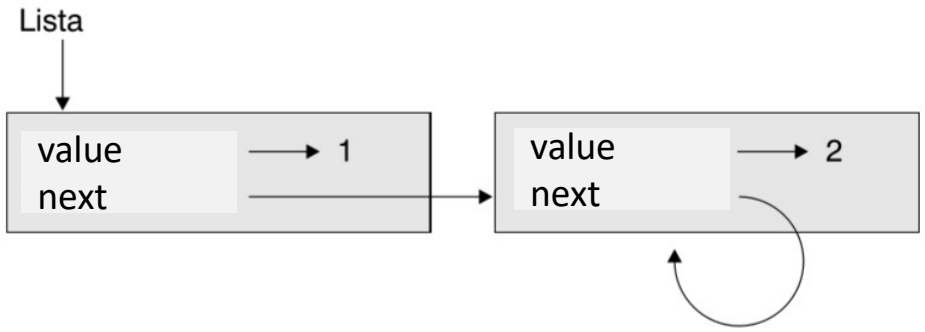
```
Nodo1.StampaInversa()
```

OUTPUT

```
In [107]: runfile('C:/Users/disit/Documents/Python
Scripts/Nodo_lista.py', wdir='C:/Users/disit/
Documents/Python Scripts')
3
2
1
```

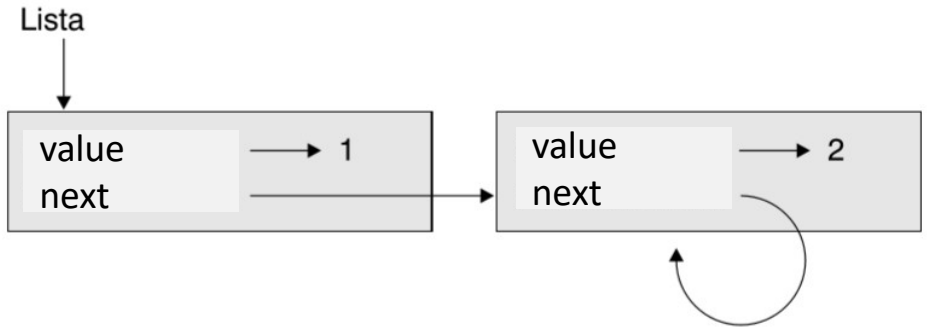
Liste infinite (1)

- Possiamo provare che StampaInversa giungerà sempre alla fine, raggiungendo il caso base?
- La risposta è no e infatti la sua chiamata causerà un errore in esecuzione nel caso in cui la lista passata come parametro sia di tipo particolare
- Non c'è nulla che vieti ad un nodo di fare riferimento ad un nodo precedente della lista o addirittura a se stesso
- Il diagramma seguente mostra una lista di due nodi in cui uno di essi si riferisce a se stesso:



Liste infinite (2)

- Se invocassimo StampaLista o StampaInversa su questa lista si creerebbe una ricorsione infinita: questo tipo di comportamento rende particolarmente difficile lavorare con le liste



- In alcuni casi, le liste infinite possono rivelarsi molto utili:
  - Ad esempio se si vuole rappresentare un numero come lista di cifre usando una lista infinita per la descrizione della parte decimale periodica
- Resta comunque il problema che non è possibile dimostrare che StampaLista e StampaInversa raggiungano sempre il caso base
- La cosa migliore è stabilire una precondizione:
  - Ad esempio si assuma che: "se non sono presenti anelli all'interno della lista, questi metodi termineranno". La precondizione impone una limitazione ai parametri e descrive il comportamento di un metodo nel caso essa venga soddisfatta

Il teorema dell'ambiguità fondamentale (1)

- Una parte di StampaInversa aveva qualcosa di sospetto:

```
Testa = Lista
Coda  = Lista.next
```

- Dopo la prima assegnazione Testa e Lista hanno lo stesso tipo e lo stesso valore. Perché è stata creata una nuova variabile?
- La ragione è che le due variabili giocano ruoli differenti:
  - Testa è il riferimento ad un singolo nodo
  - Lista è il riferimento al primo nodo della lista
  - Questi "ruoli" non sono espressamente necessari al programma, servono solo per chiarire inizialmente il concetto al programmatore

Il teorema dell'ambiguità fondamentale (2)

- Spesso si usano nomi come Nodo e Lista per documentare l'uso della variabile e si introducono variabili addizionali solo per rendere meno ambiguo il codice al momento della lettura
- Avremmo anche potuto scrivere StampaInversa senza Testa e Coda. Il risultato sarebbe stato più conciso (anche se inizialmente meno chiaro):

```
def StampaInversa2(Node):
    if Node.next == None:
    print(Node)
    return
Coda = Node.next
Coda.StampaInversa2()
print(Node.value)
```

- Chiamata da main():
  - Nodo1.StampaInversa2()

OUTPUT

```
In [107]: runfile('C:/Users/disit/Documents/Python
Scripts/Nodo_lista.py', wdir='C:/Users/disit/
Documents/Python Scripts')
3
2
1
```

Il teorema dell'ambiguità fondamentale (3)

- Alternativa:

```
def StampaInversa3(self):
    if self.next == None:
            print(self)
            return
    Coda = self.next
    Coda.StampaInversa3()
    print(self.value)
```

- Il teorema dell'ambiguità fondamentale descrive l'ambiguità inerente al riferimento ad un nodo:
  - *Una variabile che si riferisce ad un nodo può trattare il nodo come oggetto singolo o come primo elemento di una lista di nodi linkati*

Liste (3)

- Definizione alternativa alla StampaLista:

```
def StampaLista(Nodo):
    while(Nodo):
            if((Nodo.next==None) and (Nodo.value==None)):
                    print("Lista vuota")
            else:
                    print (Nodo)
            Nodo = Nodo.next
```

Modifica delle liste (1)

- Ci sono due modi per modificare una lista linkata:
  - cambiare il contenuto di uno dei nodi
  - Aggiungere, rimuovere o riordinare i nodi
- Ad esempio, definiamo il metodo 'RimuoviSecondo' capace di:
  - rimuovere il secondo nodo di una lista
  - Restituire (output) un riferimento al nodo rimosso

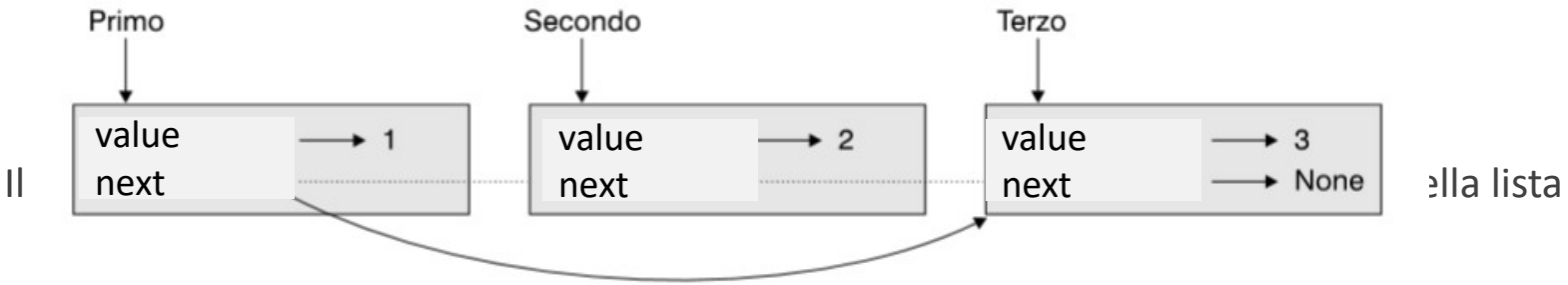
```
def RimuoviSecondo(Lista):
    if Lista == None: return
    Primo = Lista
    Secondo = Lista.next
    # il primo nodo deve riferirsi al terzo
    Primo.next = Secondo.next
    # separa il secondo nodo dal resto della lista
    Secondo.next = None
    return Secondo
```

Modifica delle liste (2)

- Chiamata dal main():
  - Nodo1.RimuoviSecondo()

DEFINIZIONE

```
def RimuoviSecondo(Lista):
    if Lista == None: return
    Primo = Lista
    Secondo = Lista.next
    # il primo nodo deve riferirsi al terzo
    Primo.next = Secondo.next
    # separa il secondo nodo dal resto della lista
    Secondo.next = None
    return Secondo
```



Il ella lista

Modifica delle liste (3)

Domande:

- Cosa succede se si invoca questo metodo passando una lista composta da un solo elemento (elemento singolo)?
- Cosa succede se si passa come argomento una lista vuota?
- C'è una precondizione per questo metodo? Se esiste riscrivere il metodo per gestire gli eventuali problemi

DEFINIZIONE

```
def RimuoviSecondo(Lista):
    if Lista == None: return
    Primo = Lista
    Secondo = Lista.next
    # il primo nodo deve riferirsi al terzo
    Primo.next = Secondo.next
    # separa il secondo nodo dal resto della lista
    Secondo.next = None
    return Secondo
```

Se si passa una lista vuota…

Errore:

```
  File "C:/Users/disit/Documents/Python Scripts/
Nodo_lista.py", line 87, in <module>
    main()

  File "C:/Users/disit/Documents/Python Scripts/
Nodo_lista.py", line 81, in main
    Vuota.RimuoviSecondo()

  File "C:/Users/disit/Documents/Python Scripts/
Nodo_lista.py", line 56, in RimuoviSecondo
    Primo.next = Secondo.next

AttributeError: 'NoneType' object has no attribute
'next'
```

Una soluzione…

```
def RimuoviSecondo(Lista):
        if ((Lista == None) or (Lista.next == None)): return
        Primo = Lista
        Secondo = Lista.next
        # il primo nodo deve riferirsi al terzo
        Primo.next = Secondo.next
        # separa il secondo nodo dal resto della lista
        Secondo.next = None
        return Secondo
```

Metodi contenitore e aiutante

- Può essere utile dividere un'operazione su una lista in due metodi
- Per esempio per stampare una lista al contrario secondo il formato convenzionale [3, 2, 1] possiamo usare il metodo StampaInversa per stampare 3, 2, ma abbiamo bisogno di un metodo diverso per stampare le parentesi ed il primo nodo
- Definiamo allora il metodo StampaInversaFormato:

```
def StampaInversaFormato(Lista):
    print("[")
    if Lista != None :
            Testa = Lista
            Coda = Lista.next
            Coda.StampaInversa ()
            print(Testa)
            print("]")
```

OUTPUT

```
[
3
2
1
]
```

Metodi contenitore e aiutante (2)

Per stampare i valori consecutivi:

```
def stampa_consecutivi(self):
    return print(self.value, end =' ')

def StampaInversaConsecutivi(self):
    if self.next == None:
    self.stampa_consecutivi()
    return
    Coda = self.next
    Coda.StampaInversaConsecutivi()
   self.stampa_consecutivi()
```

OUTPUT

```
[3 2 1 ]
```

```
def StampaInversaFormato(Lista):
    print("[")
    if Lista != None :
      Testa = Lista
      Coda = Lista.next
      Coda.StampaInversaConsecutivi()
      Testa.stampa_consecutivi()
      print("]")
```

Metodi contenitore e aiutante (3)

- Ancora una volta è una buona idea testare questo metodo per vedere se funziona correttamente anche in casi particolari, quando cioè una lista è vuota o composta da un solo elemento.
- Quando si chiama questo metodo dal main, si chiama direttamente StampaInversaFormato e questa invoca a sua volta StampaInversa. In questo senso StampaInversaFormato agisce come un contenitore che usa StampaInversa come aiutante.

```
def StampaInversaFormato(Lista):
    print("[")
    if Lista != None :
            Testa = Lista
            Coda = Lista.next
            Coda.StampaInversa ()
            print(Testa)
            print("]")
```

```
[3 2 1 ]
```

PILE

Pile – pag 186

https://www.python.it/doc/Howtothink/Howtothink-html-it/

Salta per ora

Pile (1)

- I tipi di dati visti fino ad ora sono concreti, ovvero sono stati [illegible]ficati al momento della implementazione
- Un **tipo di dato astratto** (TDA) specifica un insieme di operaz[illegible]ro semantica (cosa fa ciascuna operazione) ma senza specificare la loro im[illegible]ta caratteristica è ciò che lo rende astratto
- Per che cosa è utile questa "astrazione"?
- Semplifica il compito di specificare un algoritmo, dato che si può decidere cosa dovranno fare le operazioni senza dover pensare allo stesso tempo a come implementarle

Salta per ora

Pile (2)

- Ci sono molti modi per implementare un TDA e può essere u lgoritmo in grado di funzionare per ciascuna delle possibili implementaz
- TDA molto ben conosciuti, tipo la Pila (o Stack) che vedremo ono spesso implementati nelle librerie standard dei vari linguaggi di programmazione così da poter essere usati da molti programmatori senza dover essere reinventati ogni volta.
- Le operazioni sui TDA forniscono un linguaggio di alto livello che consente di specificare e descrivere gli algoritmi.
- Quando parliamo di TDA spesso distinguiamo il codice che usa il TDA (**cliente**) dal codice che lo implementa (**fornitore**).

Salta per ora

Il TDA Pila (1)

- La Pila è un tipo di dato astratto molto comune
- Una pila è una collezione e cioè una struttura di dati che con li
- Altre collezioni che abbiamo già visto sono i dizionari e le list
- Un TDA è definito dalle operazioni che possono essere effettuate su di esso e che sono chiamate interfaccia

Salta per ora

Il TDA Pila (2)

- L'interfaccia per una pila consiste di queste operaz
  - __init__: Inizializza un pila vuota.
  - Push: Aggiunge un elemento alla pila.
  - Pop: Rimuove e ritorna un elemento dalla pila. L'elemento t ltimo inserito.
  - EVuota: Controlla se la pila è vuota

Salta per ora

- Una pila è spesso chiamata struttura di dati LIFO ("last in/first out", ultimo inserito, primo fuori) perché l'ultimo elemento inserito in ordine di tempo è il primo ad essere rimosso: un esempio è una serie di piatti da cucina sovrapposti, ai quali aggiungiamo ogni ulteriore piatto appoggiandolo sopra agli altri, ed è proprio dall'alto che ne preleviamo uno quando ci serve

Implementazione delle pile con le liste

- Il codice che segue è chiamato implementazione del TDA Pila
- Più in generale un'implementazione è un insieme di metod ntassi e la semantica dell'interfaccia richiesta

Salta per ora

```
class Pila:
    def __init__(self):
            self.Elementi = []

    def Push(self, Elemento):
            self.Elementi.append(Elemento)

    def Pop(self):
            return self.Elementi.pop()

    def EVuota(self):
            return (self.Elementi == [])
```

Implementazione delle pile con le liste (2)

- L'oggetto Pila contiene un attributo chiamato Elementi che è [illegible] tenuta nella pila
- Il metodo init inizializza Elementi come lista vuota
- Push inserisce un nuovo elemento nella pila aggiungendolo a Elementi
- Pop esegue l'operazione inversa, rimuovendo e ritornando l'ultimo elemento inserito nella pila
- Per controllare se la pila è vuota EVuota confronta Elementi con una lista vuota e ritorna vero/falso

Salta per ora

Implementazione delle pile con le liste (3)

- Un'implementazione di questo tipo in cui i metodi sono solo [illegible] zione di metodi già esistenti viene detta maschera
- In informatica, una maschera è un pezzo di codice che nasco[illegible] un'implementazione per fornire un'interfaccia più semplice

Salta per ora

- Sia inoltre la funzione StampaPila tale da stampare i valori degli elementi che sono nella pila:

```
def StampaPila(self):
    while not self.EVuota():
        print(self.Pop())
```

Implementazione delle pile con le liste (4)

- Se si effettua la seguente sequenza di chiamate da main():

```
def main():
    P = Pila()
    P.Push(54)
    P.Push(45)
    P.Push("+")
    P.StampaPila()
```

Salta per ora

- Si ottiene la stampa degli elementi in ordine inverso!
- Anche se questo non è il formato standard per la stampa di una lista usando una pila

```
In [188]: runfile('C:/Users/disit/Documents/Python
Scripts/Pila.py', wdir='C:/Users/disit/Documents/
Python Scripts')
+
45
54
```

OUTPUT

Implementazione delle pile con le liste (5)

- Confrontando questo codice con l'implementazione di Stam[illegible]nkate, le due versioni sono molto più simili di ciò che sembra a prima vista[illegible] dello stesso meccanismo:
  - mentre nell'implementazione della classe Pila appena scritta l'u[illegible]nella versione ricorsiva vista in precedenza il carico della gestione della pila era delegato all'interprete stesso
  - Ad ogni chiamata di funzione infatti viene usata una pila interna all'interprete che tiene conto della successione delle chiamate alle funzioni

Salta per ora

Uso della pila per valutare espressioni postfisse (1)

- A partire dall'inizio dell'espressione ricava un termine (oper a volta
  - Se il termine è un **operando**, si aggiunge all'inizio della pila
  - Se il termine è un **operatore**, si estrae dalla pila il numero di ope ratore, si elabora il risultato dell'operazione su di essi e si aggiunge il risultato all'i
- Quando tutta l'espressione è stata elaborata, nella pila ci dovrebbe essere un solo elemento che rappresenta il risultato
- Esercizio: applica questo algoritmo all'espressione 1 2 + 3 *.

Salta per ora

Uso della pila per valutare espressioni postfisse (2)

- Nella maggior parte dei linguaggi di programmazione le espr… sono scritte con l'operatore tra i due operandi, come nella consueta 1+2
  - Questo formato è chiamato notazione infissa
- Un modo alternativo, è chiamato notazione postfissa: nella … operatore segue gli operandi, tanto che l'espressione appena vista sarebbe scritta in questo modo: 1 2 +
- Il motivo per cui la notazione postfissa può rivelarsi utile è che c'è un modo del tutto naturale per valutare espressioni postfisse con l'uso della pila

Salta per ora

Parsing (1)

- Per implementare l'algoritmo di valutazione dell'espressione [illegible] grado di attraversare una stringa e di dividerla in una serie di operand[illegible]
- Questo processo è un esempio di parsing e il risultato consis[illegible]nenti chiamati token
- Python fornisce un metodo split in due moduli, sia in string (per la gestione delle stringhe) che in re (per le espressioni regolari)

Salta per ora

Parsing (2)

- La funzione string.split divide una stringa scomponendola in [illegible] sando un singolo carattere come delimitatore:

Salta per ora

```
import string
def main():
    string="Nel mezzo del cammin"
    print(string.split(" ") )
```

OUTPUT

```
['Nel', 'mezzo', 'del', 'cammin']
```

Parsing (3)

- In questo caso il delimitatore è il carattere spazio, quindi la s[illegible] ad ogni spazio
- La funzione re.split è molto più potente, permettendo l'uso [illegible]golare invece di un delimitatore singolo
- Un'espressione regolare è un modo per specificare un insieme di stringhe e non soltanto un'unica stringa. Ad esempio:
  - [A-Z] è l'insieme di tutte le lettere maiuscole dell'alfabeto
  - [0-9] è l'insieme di tutti i numeri
  - L'operatore ^ effettua la negazione dell'insieme così che [^0-9] rappresenta l'insieme di tutto ciò che non è un numero

Salta per ora

Parsing (4)

- Quelli visti sono solo alcuni esempi, l'espressione regolare ch[illegible]e un'espressione postfissa è la seguente:
  - `re.split("([^0-9])", "123+456*/")`

```
['123', '+', '456', '*', '', '/', '']
```

Salta per ora

OUTPUT

- Si noti come l'ordine degli operandi sia diverso da quello di string.split in quanto i delimitatori sono indicati prima della stringa da dividere
- La lista risultante include gli operandi 123 e 456, e gli operatori * e /. Include inoltre due stringhe vuote inserite dopo gli operandi.

Valutazione postfissa (1) - PAG.190

Per valutare un'espressione postfissa si usa il parser e l'algoritmo visto in precedenza

Per cominciare dalle cose più semplici, si implementano solo gli operatori + e *:

Valutazione postfissa (1)

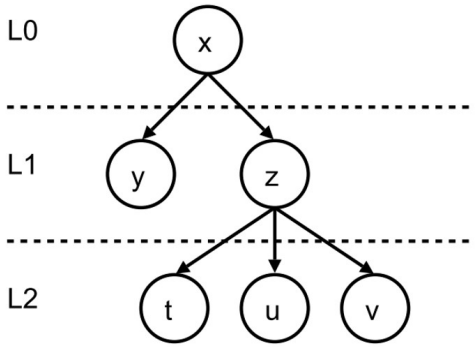
Alberi – sistema implementazione pag.220

How to think ITA

Alberi

Albero (1)

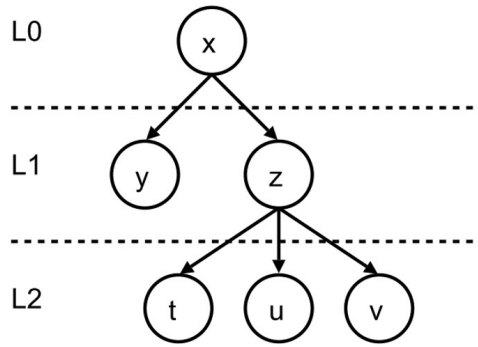
- L'albero è un insieme di elementi, detti **nodi**, sui quali è definita una relazione di discendenza
- L'albero gode delle seguenti caratteristiche:
  - Esiste un unico nodo, detto **radice**, che non ha predecessori
  - Qualsiasi altro nodo ha un unico predecessore
  - I nodi che non hanno successori sono detti **foglie**
  - Un nodo che non è né la radice né una foglia si dice **intermedio**
- Nella caratterizzazione di un albero hanno rilevanza:
  - il **grado di uscita** e la **profondità** dei nodi



- Il grado di uscita di un nodo è il numero dei suoi successori diretti
- La profondità definisce la distanza di un nodo dalla radice ed è definita in modo ricorsivo:
  - la radice ha profondità 0
  - un generico nodo diverso dalla radice ha profondità pari a quella del suo predecessore aumentata di 1
- Per estensione, la **profondità di un albero** è il massimo delle profondità dei diversi nodi

Albero (2)

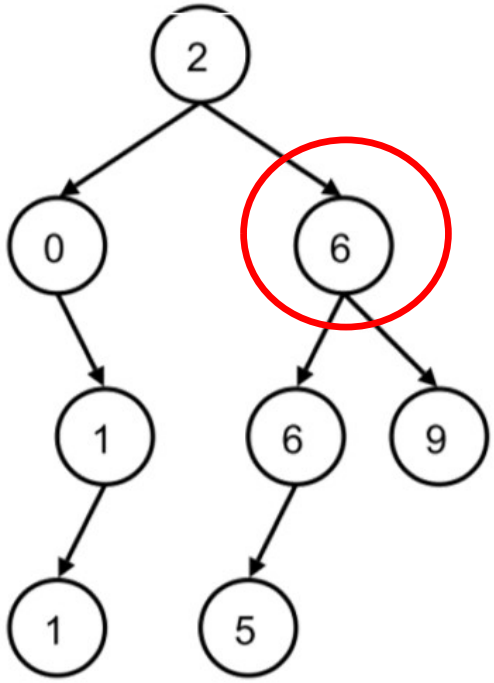
- L'insieme dei nodi che si trovano ad una stessa profondità forma un livello
- Su un albero si eseguono sostanzialmente le stesse operazioni che si applicano anche ad una lista, con un paio di differenze:
  - su un albero non esiste un unico nodo terminale ma esiste invece una molteplicità di foglie, quindi l'operazione di inserimento in "coda" deve essere qualificata con un criterio specifico capace di selezionare quale tra le diverse foglie è il target dell'operazione di inserimento
  - l'inserimento sulla radice o su nodi intermedi tende a degenerare la struttura dell'albero verso una forma sequenziale, per cui gli inserimenti sono tipicamente eseguiti sulle foglie
- Esempio:
  - l'elemento x è la radice e ha due successori y e z
  - z ha tre successori t, u e v
  - i nodi y, t, u e v sono foglie
  - z è un nodo intermedio e realizza il massimo grado di uscita (pari a 3)
  - l'albero è ripartito su tre livelli ed ha profondità 2



- NOTA: un albero in cui ciascun nodo ha al massimo un successore degenera nella forma di una lista

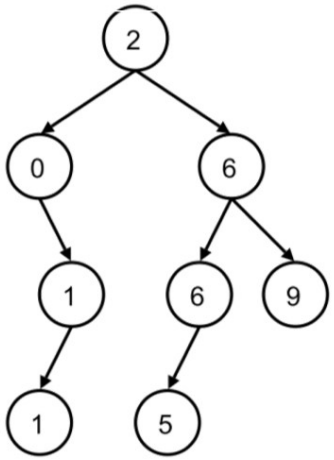
Alberi Binari di Ricerca (1)

- Un albero binario è un albero sul quale:
  - ciascun nodo ha grado di uscita minore o uguale a 2
- Un albero binario si dice '**Albero binario di ricerca**' quando:
  - il valore codificato su ciascun nodo è maggiore o uguale del valore codificato sul figlio sinistro del nodo stesso e minore secco del valore codificato sul figlio destro



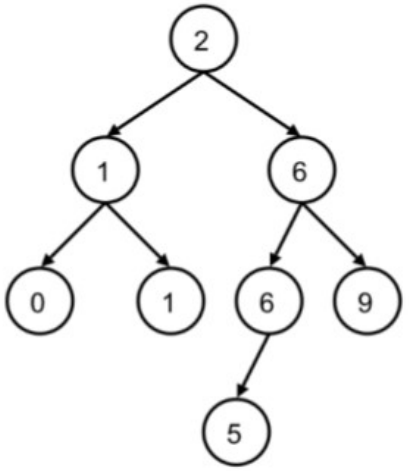
Albero Binario di Ricerca Bilanciato (1)

- Un albero binario si dice anche bilanciato quando ogni nodo che si trova su un livello diverso dall'ultimo o dal penultimo ha due figli



Albero binario di ricerca

NON Bilanciato

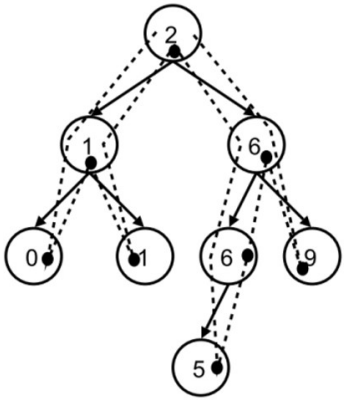


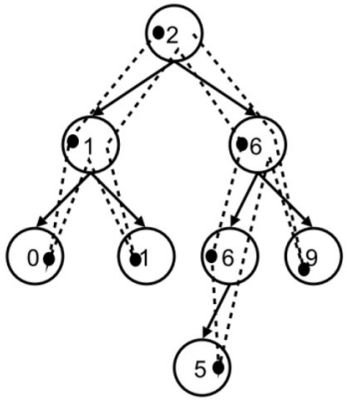
Albero binario di ricerca

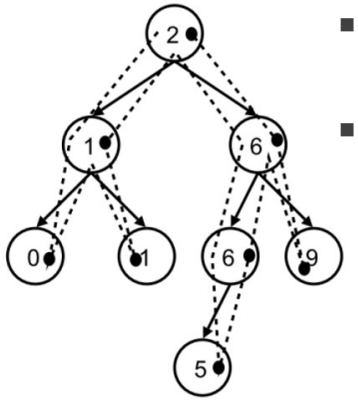
Bilanciato

Ordine di visita di un Albero Binario

- L'ordine di visita dei nodi di un Albero Binario può essere in forma:
  - Simmetrica
  - Anticipata
  - Posticipata







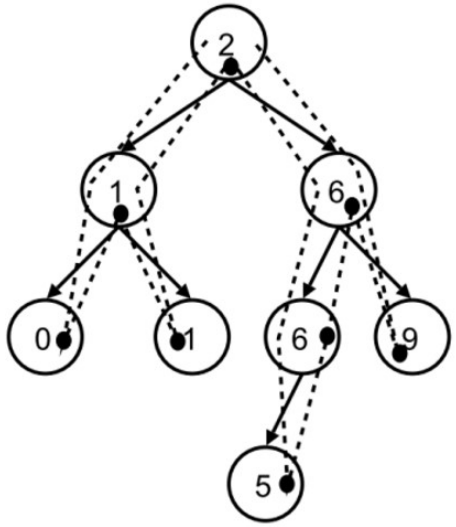
- La linea tratteggiata rappresenta il flusso del controllo
- I punti marcati lungo la linea rappresentano il momento in cui si esegue l'operazione della visita del nodo (es: stampa del valore del nodo visitato) e il numero indica l'ordine di visita

Visita Simmetrica di un Albero binario

- Ordine:
  - Sottoalbero **sinistro**
  - **Radice** (es: operazione stampa)
  - Sottoalbero **destro**

- La radice viene trattata:
  - **Solo DOPO** aver trattato tutti i nodi del sottoalbero **sinistro**, e **PRIMA** di trattare i nodi del sottoalbero **destro**

- Così facendo, SE l'albero è di ricerca, i valori sono stampati in ordine.

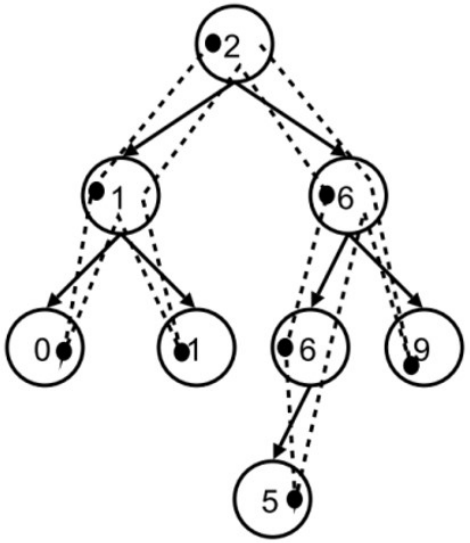


- Un Albero Binario di Ricerca è t.c. il valore codificato su ciascun nodo è: i) maggiore o uguale del valore codificato sul figlio sinistro del nodo stesso; ii) minore secco del valore codificato sul figlio destro

Visita anticipata e posticipata di un Albero binario

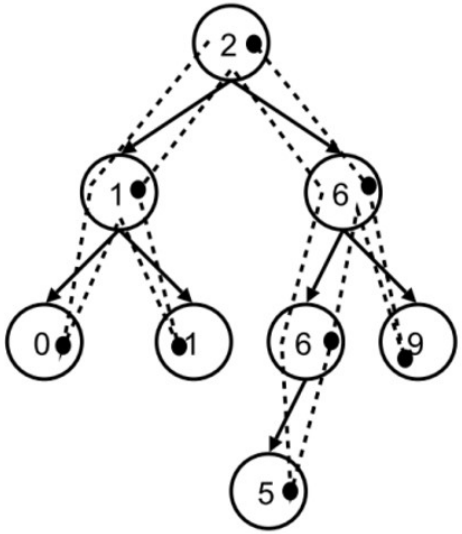
Visita Anticipata

- Ordine:
  - Radice (es: operazione stampa)
  - Sottoalbero sinistro
  - Sottoalbero destro



Visita Posticipata

- Ordine:
  - Sottoalbero sinistro
  - Sottoalbero destro
  - Radice (es: operazione stampa)



Rappresentazione di un Albero Binario in Python

Classe ‘Node’ e Inizializzazione

```
class Nodo:
        def __init__(self, value=None, next=None):
                self.value = value
                self.next = next
        def __str__(self):
                return str(self.value)
```

NOTA:
DEFINIZIONE
nodo lista linkata

```
class Node:
'''Rappresenta il nodo di un Albero'''
'''Metodo speciale init invocato quando un oggetto viene istanziato'''
  def __init__(self, data):
      self.right = None
      self.left = None
      self.data = data

  '''metodo speciale __str__ ha come scopo quello di restituire una rappresentazione
  di un oggetto in termini di stringa (usato per la stampa di un nodo). Se non si
  implementa la funzione __str__() per una classe, allora si usa l'implementazione
  dell'oggetto incorporato che effettivamente chiama la funzione __repr__().'''
  def __str__(self):
      return f'{self.data}’


  def __repr__(self):
      return f'{self.data}'
```

Inserimento ordinato

```
def insert(n, value):
 '''Insert value in a binary tree with root n Keyword arguments: n -- a Node object
 asd value --anything that can be compared with other values in tree'''
 if value <= n.data: #se il valore da inserire è minore del data attuale, vado a sx
    if n.left: #se esite già un figlio sx, richiamo la insert su tale nodo
       insert(n.left, value)
    else: #se non esiste già un figlio sx, ce lo metto con il valore attuale in data
       n.left = Node(value) #uso il costruttore con input il valore attuale
 else: #se il valore da inserire è maggiore del data attuale, vado a dx
    if n.right: #se esite già un figlio dx, richiamo la insert su tale nodo
       insert(n.right, value)
    else: #se non esiste già un figlio dx, ce lo metto con il valore attuale in data
       n.right = Node(value)
```

Inserire elementi random in un Albero Binario

Ricordando che la classe Node è definita nel modo seguente:

```
import random
class Node:
   def __init__(self, data):
       self.right = None
       self.left = None
       self.data = data
```

```
def populate_tree(n, nvalues, min_v=0, max_v=10):
   if n is None:
       n = Node(random.randint(min_v, max_v))
       for _ in range(nvalues):
          insert(n, random.randint(min_v, max_v))
          return n
```

Inserire elementi in un Albero Binario da console

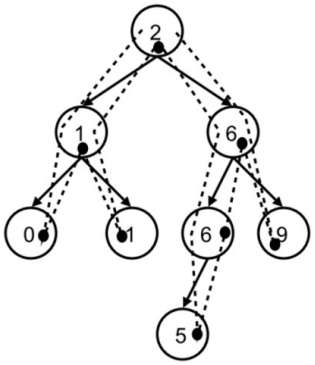
- Ricordando che la classe Node è definita nel modo seguente:

```
import random
class Node:
   def __init__(self, data):
       self.right = None
       self.left = None
       self.data = data
```

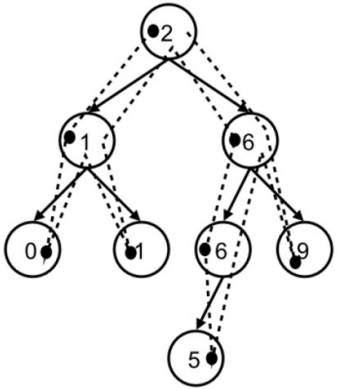
```
def populate_tree_from(values): #prende in ingresso una lista di valori
       print(values) #li stampa
       n = Node(values[0]) #inizializza l'albero inserendo il primo elemento
       for v in values[1:]: #inserisce tutti gli altri elementi
              insert(n, v)
       return n
```

Visita di un albero (funzioni esterne alla classe Node)

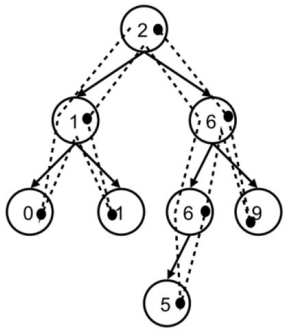
```
def visit(n):
   '''visita simmetrica'''
   if n:
     if n.left:
         visit(n.left)
     print(n.data)
     if n.right:
         visit(n.right)
```




```
def visit_pre(n):
   '''visita anticipata'''
   if n:
     print(n.data)
     if n.left:
         visit(n.left)
     if n.right:
         visit(n.right)
```

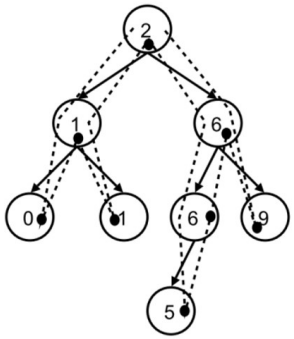



```
def visit_post(n):
'''visita posticipata'''
    if n:
       if n.left:
          visit(n.left)
       if n.right:
          visit(n.right)
       print(n.data)
```

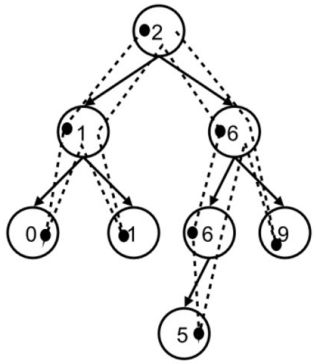


Visita di un albero (metodi della classe Node)

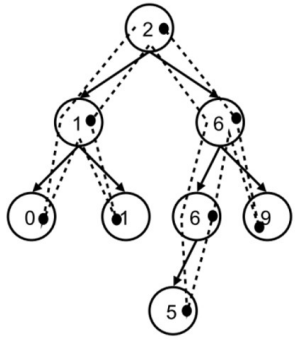
```
def visit(n):
'''visita simmetrica'''
if n:
  if n.left:
       n.left.visit()
  print(n.data)
  if n.right:
       n.right.visit()
  #nomeclasse.metodo
```




```
def visit_pre(n):
'''visita anticipata'''
if n:
  print(n.data)
  if n.left:
       n.left.visit()
  if n.right:
       n.right.visit()
```

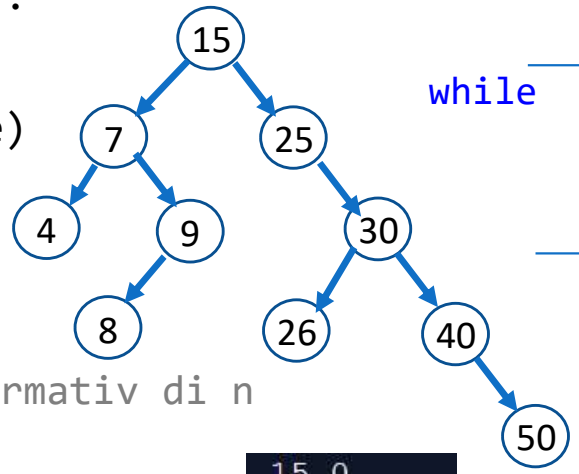



```
def visit_post(n):
'''visita posticipata'''
if n:
  if n.left:
       n.left.visit()
  if n.right:
       n.right.visit()
  print(n.data)
```

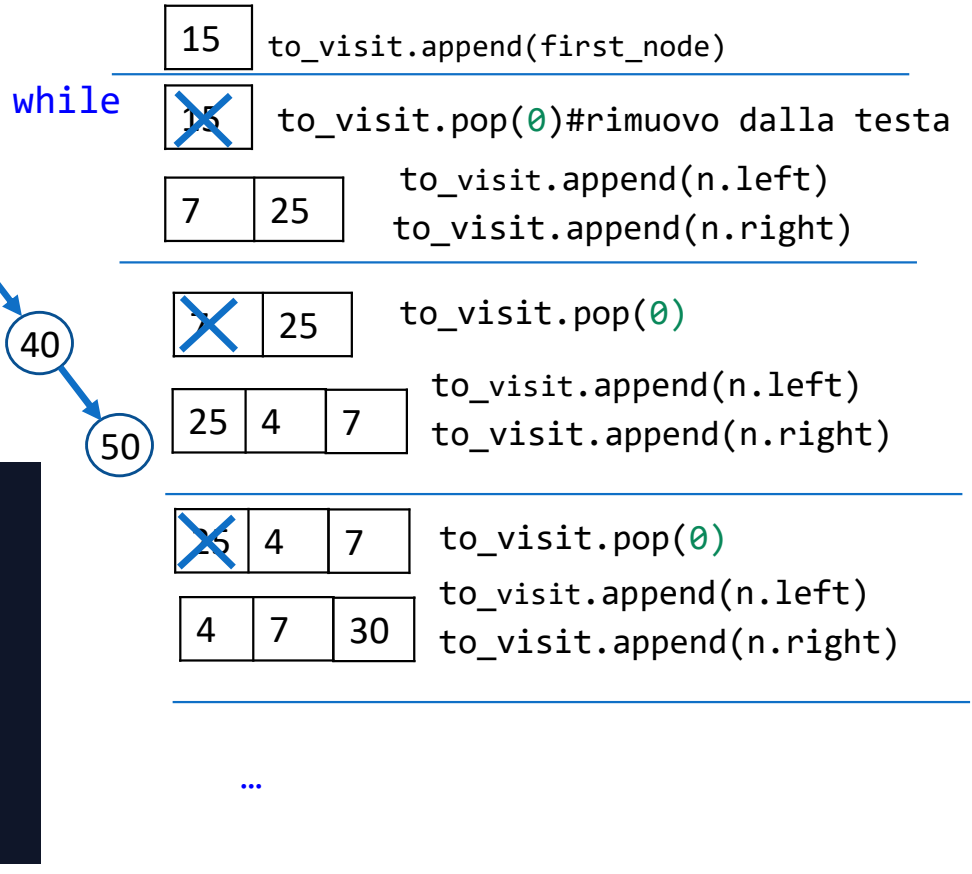


Visita in ampiezza

```
def visit_breadth(first_node):
    to_visit = []
    to_visit.append(first_node)
    while to_visit:
        #rimuovo dalla testa
        n = to_visit.pop(0)
        #stampo contenuto informativ di n
        print(n.data)
        if n.left:
            #inserico figlio sx
            to_visit.append(n.left)
        if n.right:
          #inserico figlio dx
          to_visit.append(n.right)
```



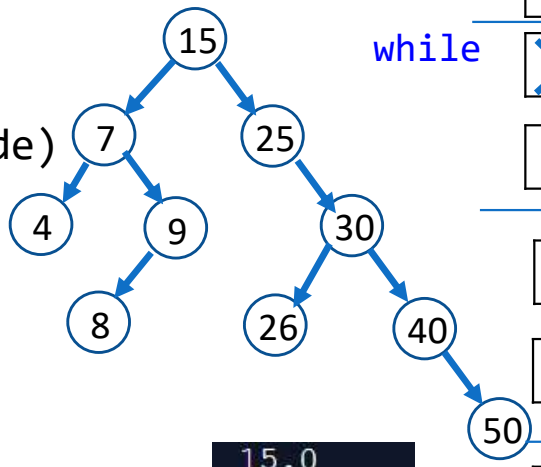

```
15.0
7.0
25.0
4.0
9.0
30.0
8.0
26.0
40.0
50.0
```



Esercitazione

Visita in profondità

```
def visit_depth(first_node):
    to_visit = []
    to_visit.append(first_node)

    while to_visit:
        n = to_visit.pop()
        print(n.data)
        if n.left:
            ìto_visit.append(n.left)
        if n.right:
            to_visit.append(n.right)
```




```
15.0
25.0
30.0
40.0
50.0
26.0
7.0
9.0
8.0
4.0
```

while

15	to_visit.append(first_node)
~~15~~	to_visit.pop()#rimuovo dalla coda
7 | 25	to_visit.append(n.left) to_visit.append(n.right)
7 | ~~25~~	to_visit.pop()#rimuovo coda
7 | 30	to_visit.append(n.left) to_visit.append(n.right)
7 | ~~30~~	to_visit.pop()#rimuovo coda
7 | 26 | 40	to_visit.append(n.left) to_visit.append(n.right)

…

Ricerca su un Albero Binario

- Su un albero binario di ricerca è possibile eseguire una ricerca in modo efficiente, evitando di visitare esaustivamente tutti i nodi
- L'algoritmo di ricerca è descritto in forma ricorsiva:
  - SE la radice contiene il valore da cercare, la ricerca termina con successo
  - SE il valore della radice è maggiore (minore) del target allora questo può trovarsi solo nel sottoalbero sinistro (destro) per cui l'esito della ricerca è quello della ricerca sul sottoalbero sinistro (destro)
  - SE la ricerca fallisce appena viene raggiunto un sottoalbero vuoto

```
def search(n, data):
    if n:
        if data < n.data:#cerco in sottoalbero sx
            return search(n.left, data)
        elif data > n.data:#cerco in sottoalbero dx
            return search(n.right, data)
        else:#la ricerca termina con successo
            return True, n
    else:
        return False, None
```

Esercizio (1)

- Scrivere una funzione che conti il numero dei nodi in un albero binario

```
def count_tree(n):
        if n:
                return 1 + count_tree(n.left) + count_tree(n.right)
        else:
                return 0
```

Esercizio (2)

- Scrivere una funzione che la somma totale dei valori dei nodi in un albero binario

```
def sum_tree(n):
        if n:
                return n.data + sum_tree(n.left) + sum_tree(n.right)
        else:
                return 0
```

Esercizio (3)

- Scrivere una funzione che restituisca la somma totale dei valori dei nodi e il numero totale dei nodi in un albero binario (tupla)

```
def sc_tree(n):
  if n:
    return    n.data + sum_tree(n.left) + sum_tree(n.right), 1 + count_tree(n.left) + count_tree(n.right)
  else:
    return 0, 0
```

Esercizio (4)

- Tenendo presente le funzioni precedenti, scrivere una funzione che restituisca il numero totale e la somma totale dei valori dei nodi in un albero binario

Sapendo che:

```
def sc_tree(n):
  if n:
    return    n.data + sum_tree(n.left) + sum_tree(n.right), 1 + count_tree(n.left) + count_tree(n.right)
  else:
    return 0, 0
```

```
def average_nodes(n):
        s, c = sc_tree(n)
        return s / c
```

Esercizio (5)

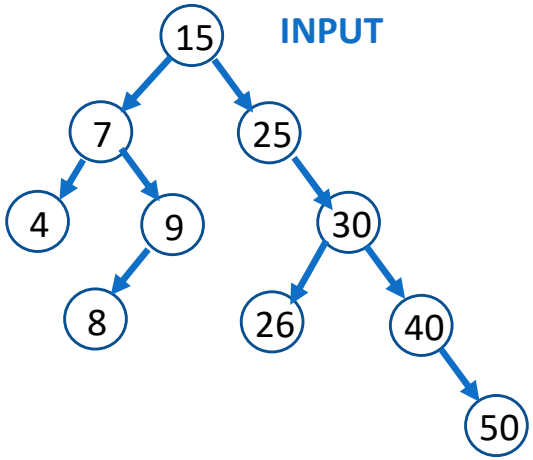
Scrivere una funzione booleana che, dato un nodo in input, restituisca True se è una foglia, False altrimenti

```
def is_leave(n):
    # rende true se sia il figlio destro che il sinistro di node
    # non esistono, ovvero rende True SE node non ha figli (è una foglia).
    # altrimenti rende False
    return n.left is None and n.right is None
```

Esercizio 6

Scrivere una funzione che prenda in ingresso un albero e renda la lista delle sue foglie

```
def get_leaves(n, leaves):
        if is_leave(n):
                leaves.append(n)
        else:
                if n.left:
                        get_leaves(n.left, leaves)
                if n.right:
                        get_leaves(n.right, leaves)
```



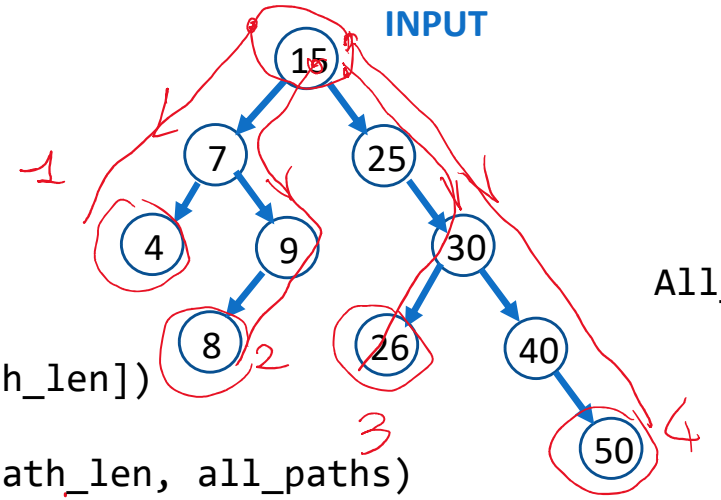
OUTPUT

[4.0, 8.0, 26.0, 50.0]

Esercizio (7) 15, 7,25,4,9,8,30,26,40,50

- Scrivere una funzione che prenda in ingresso un albero binario e renda una lista con tutti i possibili percorsi (path) per arrivare alle foglie (N foglie, N percorsi)

```
#path_len all'inizio è zero e path e all_paths è sono liste vuote
def print_paths(n, path, path_len, all_paths):
    if n is None:
        return
    if len(path) > path_len:
        path[path_len] = n
    else:
        path.append(n)
        path_len = path_len + 1
    if is_leave(n):
        all_paths.append(path[0:path_len])
    else:
        print_paths(n.left, path, path_len, all_paths)
        print_paths(n.right, path, path_len, all_paths)
```



path []

path | 15

path | 15 | 7

path | 15 | 7 | 4
path

All_path [[15,7,4]]

OUTPUT

[[15.0, 7.0, 4.0], [15.0, 7.0, 9.0, 8.0], [15.0, 25.0, 30.0, 26.0], [15.0, 25.0, 30.0, 40.0, 50.0]]

Esempio di main (1)

```
[…] #definizione di classe, metodi, funzioni o import

if __name__ == "__main__":
   choice = ' '
   root = None
     while choice != '0':
     print('1) Inserisci valori in albero')
     print('2) visita albero')
     print('3) visita albero in ampiezza')
     print('4) ricerca valore')
     print('5) somma i valori')
     print('6) conta gli elementi')
     print('7) restituire tupla con: numero totale e la somma totale dei valori dei nodi')
     print('8) stampa la media dei valori')
     print('9) rende true se l\'albero ha un solo nodo')
     print('10) rende la lista delle foglie')
     print('0) esci')
     choice = input('scelta: ')  […]
```

Esempio di main (2)

```
[…] #definizione di classe, metodi, funzioni o import
```

```
if __name__ == "__main__":
   choice = ' '
   root = None
   while choice != '0':
     […]
     choice = input('scelta: ')
     if choice == '1':
         vals_str = input('inserisci i valori separati da virgole: ')
         vals_str = vals_str.split(', ')
         vals = []
         for v in vals_str:
                 vals.append(float(v))
                 root = populate_tree_from(vals)
     elif choice == '2':
         visit(root)
     elif choice == '3':
         visit_breadth(root) […]
```

Esempio di main (3)

```
[…] #definizione di classe, metodi, funzioni o import
if __name__ == "__main__":
   choice = ' '
   root = None
   while choice != '0':
     […]
     choice = input('scelta: ')
     […]
     elif choice == '4':
         data = input('valore?: ')
         found, value = search(root, float(data))
     if found:
         print(f'trovato: {value}')
     else:
         print('non trovato!')
     elif choice == '5':
         #number = count_tree(root)
         number = sum_tree(root)
         print('Somma: ',number)
         nodo = input('inserisci i valori separati da virgole: ')
     elif choice == '6':
         print(count_tree(root))
```

Esempio di main (4)

```
[…] #definizione di classe, metodi, funzioni o import
if __name__ == "__main__":
   choice = ' '
   root = None
   while choice != '0':
     […]
     choice = input('scelta: ')
     […]
     elif choice == '7’:
         print(sc_tree(root))
     elif choice == '8’:
         print(average_nodes(root))
     elif choice == '9’:
         print(is_leave(root))
     elif choice == '10’:
         leaves = []
         get_leaves(root, leaves)
         print(leaves)
     […]
```

Esempio di main (5)

```
[…] #definizione di classe, metodi, funzioni o import
if __name__ == "__main__":
   choice = ' '
   root = None
   while choice != '0':
     […]
     choice = input('scelta: ')
     […]
     elif choice == '11’:
         path = []
         all_paths=list()
         print_paths(root, path, 0, all_paths)
         print(all_paths)
     elif choice == '0’:
         pass
     else:
         print('valore errato: '+ choice) #if none of the above user entered something
     wrong
```

Esempio di possibile main – (1)

```
[…]
#definizioni…
if __name__ == "__main__":
   choice = ' '
   root = None
   while choice != '0':
   print('1) Inserisci valori in albero')
   print('2) visita albero')
   print('3) visita albero in ampiezza')
   print('4) ricerca valore')
   print('5) somma i valori')
   print('6) conta gli elementi')
   print('7) restituire tupla con: numero totale e la somma totale dei valori dei nodi')
   print('8) stampa la media dei valori')
   print('9) rende true se l\'albero ha un solo nodo')
   print('10) rende la lista delle foglie')
   print('0) esci')
   choice = input('scelta: ')
```

Esempio di possibile main – (2)

```
[…]
if choice == '1’:
        vals_str = input('inserisci i valori separati da virgole: ‘)
        vals_str = vals_str.split(',’)
        vals = []
        for v in vals_str:
                vals.append(float(v))
        root = populate_tree_from(vals)
elif choice == '2’:
        visit(root)
elif choice == '3’:
        visit_breadth(root)
[…]
```

Esempio di possibile main – (3)

```
[…]
elif choice == '4’:
        data = input('valore?: ‘)
        found, value = search(root, float(data))
        if found:
                print(f'trovato: {value}’)
        else:
                print('non trovato!’)
elif choice == '5’:
        number = sum_tree(root)
        print('Somma: ',number)
        nodo = input('inserisci i valori separati da virgole: ')
elif choice == '6’:
        print(count_tree(root))
[…]
```

Esempio di possibile main – (4)

```
[…]
elif choice == '7’:
        print(sc_tree(root))
elif choice == '8’:
        print(average_nodes(root))
elif choice == '9’:
        print(is_leave(root))
elif choice == '10’:
        leaves = []
        get_leaves(root, leaves)
        print(leaves)
[…]
```

Esempio di possibile main – (5)

```
[…]
elif choice == '11’:
        path = []
        all_paths=list()
        print_paths(root, path, 0, all_paths)
        print(all_paths)
elif choice == '0’:
        pass
else:
        print('valore errato: '+ choice) #if none of the above user entered
        something wrong
[…]
```

Algoritmi di ordinamento in una Lista

FARE algoritmi di ordinamento da altro libro

+ vedi cosa ha fatto Lorenzo

Ordinamento (1)

- L'ordinamento è una delle operazioni più frequenti nella elaborazione di dati
- Ci sono vari algoritmi di ordinamento e tecniche che permettono di confrontarne le prestazioni
- Dopo aver ordinato una lista, vi si possono facilmente effettuare ricerche, per verificare la presenza di specifici elementi

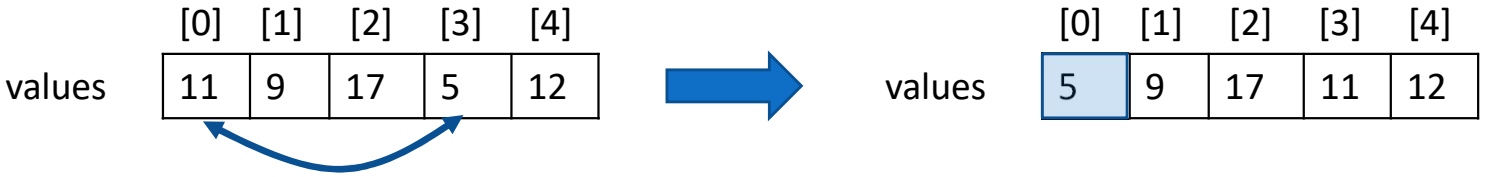
Ordinamento per selezione o Selection Sort (1)

- Problema: Data una sequenza di elementi memorizzati in una Lista, ordinare i valori dal più piccolo al più grande
- L'algoritmo Selection-sort sposta gli elementi di una raccolta di dati in modo che, al termine, siano memorizzati in qualche ordine specifico
- L'algoritmo Selection-sort ordina gli elementi di una lista cercando ripetutamente l'elemento minore della zona terminale della lista non ancora ordinata, spostandolo all'inizio della stessa
- Partiamo dall'ordinamento di una lista di numeri interi

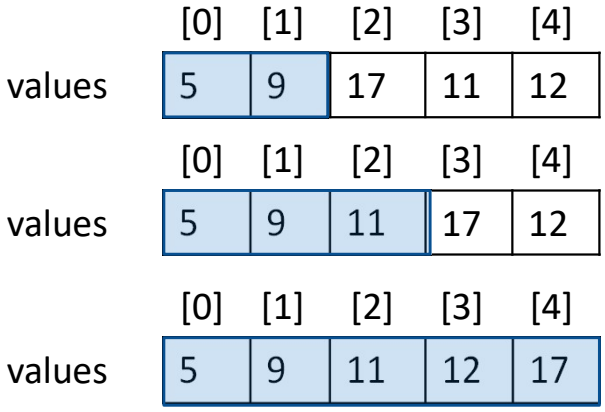
	[0]	[1]	[2]	[3]	[4]
values	11	9	17	5	12

- Una prima fase consiste nella ricerca dell'elemento minimo (5)
- Si deve spostare l'elemento minio all'inizio della lista, dove c'è un altro elemento memorizzato (11)
- Non è possibile spostare semplicemente values[3] in values[0], si perderebbe il valore 11. Non sappiamo dove andrà l'11, ma sappiamo che nn starà in values[0], quindi posticipiamo il problema mettendolo al posto del 5 in values[3]

Ordinamento per selezione o Selection Sort (2)



- In questo modo il primo elemento è nel posto giusto
- Ripetendo lo stesso procedimento si ha la seguente sequenza:



- Questo algoritmo ordinerà una qualunque lista di numeri interi
- Questo algoritmo è corretto ma ha prestazioni deludenti se eseguito su grandi insiemi di dati

Selection Sort in Python (1)

```
def minimuPosition(values, start):
   minPos = start #la prima volta che entro inizializzo minPos a 0
   #divido la lista in due
   #creco il minimo tra i valori che stanno a destra del valore che
   #attualmente ha posizione minima, ovvero che hanno minPos >=
   start+1
   for i in range(start + 1, len(values)):
     #parto da values[i], con i=start + 1, (all'inizio values[1])
     #confronto ognuno dei valori (values[i])
     #con l'attuale valore minimo (values[minPos])
     #l'indice dell'elemento con il valore più piccolo rispetto
     #al minimo attuale viene messo in minPos e restituito
     if values[i] < values[minPos]:

       return minPosminPos = i
       print('posizione minima: ',i,' valore: ',values[i])
```

```
def selectionSort(values):
   for i in range(len(values)):
     minPos = minimuPosition(values, i)
     temp = values[minPos]
     values[minPos] = values[i]
     values[i] = temp
```

Selection Sort in Python (2)

```
def minimuPosition(values, start):
   minPos = start
   for i in range(start + 1, len(values)):
     if values[i] < values[minPos]:
       return minPosminPos = i


def selectionSort(values):
   for i in range(len(values)):
     minPos = minimuPosition(values, i)
     #si scambiano i due elementi:
     temp = values[minPos] #variabile temporanea per memorizzare values[minPos]
     values[minPos] = values[i] #sovrascrivo values[minPos] con il minimo (values[i])
     values[i] = temp #completo lo scambio usando la variabile temporanea temp
```

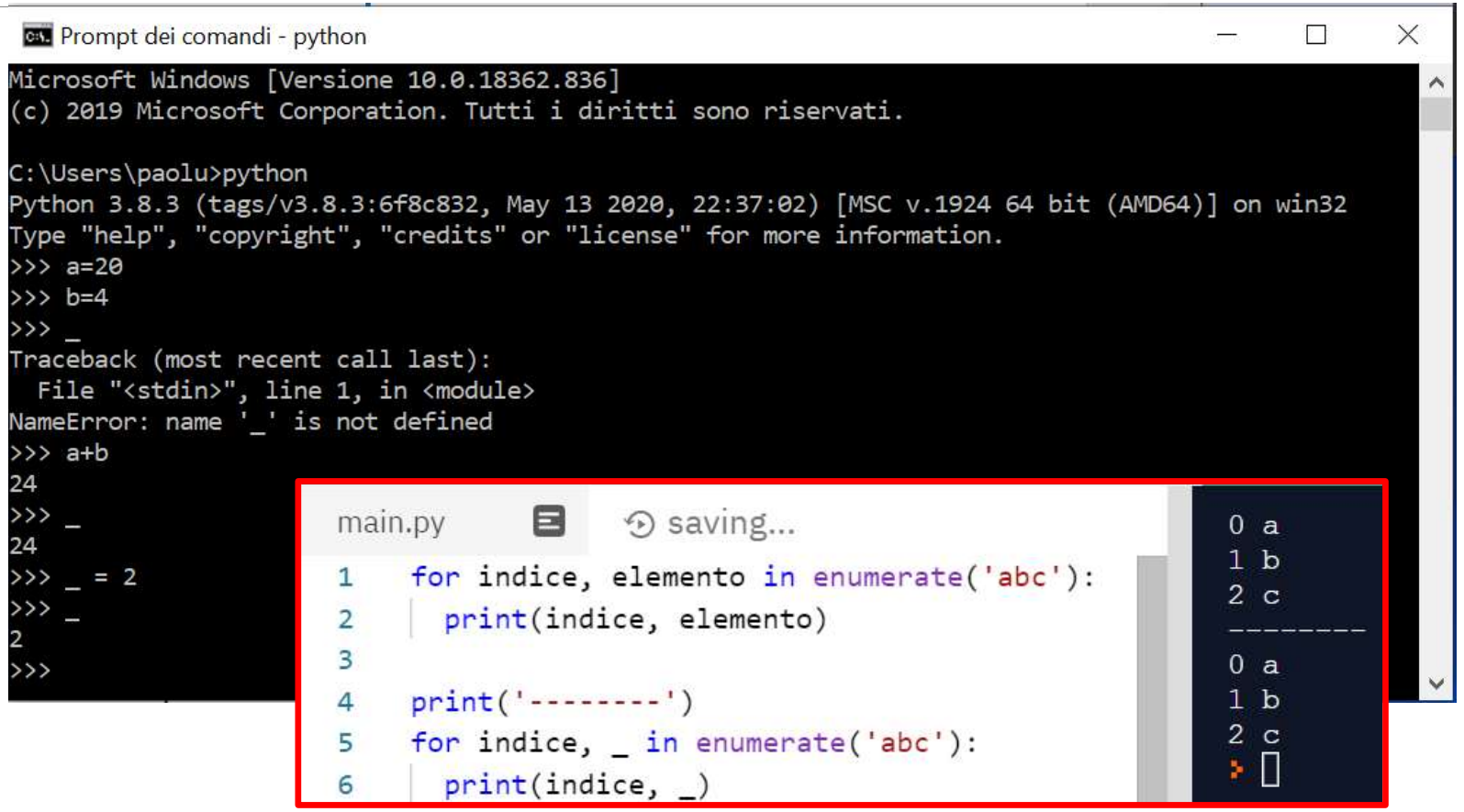
Main() che fa uso del selection_sort

```
if __name__ == "__main__":
        n=10
        values = []
        for i in range(n):
                values.append(randint(1,100))
        print(values)

        selectionSort(values)
        print(values)
```

```
[9, 7, 3, 59, 41, 49, 10, 52, 72, 70]
[3, 7, 9, 10, 41, 49, 52, 59, 70, 72]
>
```

Uso di _ e __

- '__' serve, come sappiamo, per contrassegnare le parole chiave in Python (es: '__main__') e distinguerle dalla semplici variabili
- '_ 'restituisce il valore dell'ultima espressione eseguita



Selection Sort – definizione alternativa

```
def selection_sort(values):
    for i, _ in enumerate(values):
        _, idm = max_index(values, i)
        swap(values, i, idm)
```

Con:

```
def max_index(values, start):
    M = values[start]
    id_max = start
    for i in range(start, len(values)):
        if M > values[i]:
            M = values[i]
            id_max = i
    return M, id_max
```

```
def swap(l, i1, i2):
    t = l[i1]
    l[i1] = l[i2]
    l[i2] = t
```

Prestazioni del Selection Sort (1)

- Per rilevare le prestazioni di un programma, un metodo potrebbe essere quello di eseguirlo e misurare il tempo trascorso per la esecuzione. Ma non sarebbe una metodologia efficiente perché la maggior parte dei programmi vengono eseguiti molto velocemente
- Inoltre, una certa quantità di tempo viene usata ad esempio per caricare il programma dal disco alla memoria o per visualizzare i risultati sullo schermo
- Per misurare in modo più accurato l'algoritmo utilizziamo allora la funzione di libreria time(), del modulo time che restituisce il numero di secondi (sotto forma di valori in virgola mobile) che sono trascorsi dalla mezzanotte del 1 gennaio 1970 – timestamp
- Ovviamente non ci interessa il numero assoluto di secondi ma la differenza tra il timestamp di fine esecuzione e di inizio, che ci fornisce la durata dell'intervallo temporale misurata in secondi

Misurazione delle prestazioni temporali di un algoritmo (1)

```
if __name__ == "__main__":
        n=1000
        values = []
        for i in range(n):
                values.append(randint(1,100))
        print(values)
        starttime = time()
        selectionSort(values)
        endtime = time()
        print('Size: %d Elapsed time: %3f seconds' % (n, endtime -starttime ))
        starttime = time()
        selection_sort(values)
        endtime = time()
        print('Size: %d Elapsed time: %3f seconds' % (n, endtime -starttime ))
```

OUTPUT

[….. (mille valori random) , 77, 18, 96, 84, 4, 59, 20, 36, 4, 14, 42, 4, 100, 20, 33, 47, 69, 6, 1, 29, 15, 20]
Size: 1000 Elapsed time: 0.025898 seconds
Size: 1000 Elapsed time: 0.018949 seconds

Misurazione delle prestazioni temporali di un algoritmo (2)

```
if __name__ == "__main__":
   firstsize = int(input('Enter first list size: '))
   #numero di liste su cui fare il confronto
   numberOfLists = int(input('Enter number of lists: '))
   for k in range(1, numberOfLists+1):
     size = firstsize*k
     values = []
   #genera una lista casuale
   for i in range(size):
     values.append(randint(1,100))
     starttime = time()
     selectionSort(values)
     endtime = time()
     print('Size: %d Elapsed time: %3f seconds' % (size, endtime -starttime ))
```

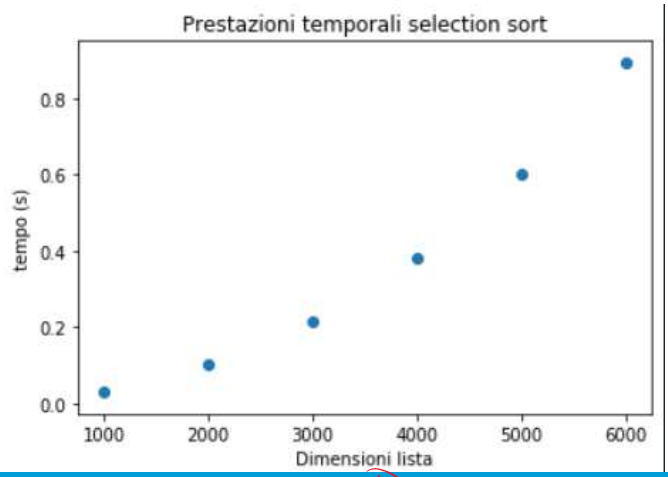
OUTPUT

```
Enter first list size: 1000

Enter number of lists: 6
Size: 1000 Elapsed time: 0.015619 seconds
Size: 2000 Elapsed time: 0.085817 seconds
Size: 3000 Elapsed time: 0.230367 seconds
Size: 4000 Elapsed time: 0.385497 seconds
Size: 5000 Elapsed time: 0.616587 seconds
Size: 6000 Elapsed time: 0.870414 seconds
```

Misurazione delle prestazioni temporali di un algoritmo (3)

- Se si avvia la misurazione del tempo prima e dopo l'esecuzione di un algoritmo si misura il tempo impiegato per ottenere il risultato voluto. Nel caso di un algoritmo di ordinamento in una lista, si misura il tempo necessario per ordinare ida ti senza tenere conto del tempo impiegato per leggere i dati forniti dall'utente o inseriti nella lista o per la visualizzazione dei risultati
- In tabella e in figura i risultati ottenuti con Intel(R) Core(TM) a 2GHz su sistema Operativo Windows 10. Su altri dispositivi le prestazioni possono essere diverse
- Si noti che raddoppiando la dimensione delle liste, il tempo che occorre per ordinarli è 'circa il doppio'

```
Enter first list size: 1000
Enter number of lists: 6

Size: 1000 Elapsed time: 0.031075 seconds
Size: 2000 Elapsed time: 0.100269 seconds
Size: 3000 Elapsed time: 0.216266 seconds
Size: 4000 Elapsed time: 0.381343 seconds
Size: 5000 Elapsed time: 0.599177 seconds
Size: 6000 Elapsed time: 0.891536 seconds
```



Analisi delle prestazioni del SelectionSort (1)

- Come passo successivo alla valutazione temporale, contiamo le operazioni che il programma deve eseguire per ordinare una lista
- In realtà non conosciamo le istruzioni di codice macchina che vengono usate per il calcolo dell'algoritmo, ne' quali sono le istruzioni che impiegano più tempo di altre, possiamo allora fare una semplificazione contando il numero di visite (in lettura o in scrittura) che vengono fatte sulla lista in esame
- Ciascuna visita richiede più o meno la stessa quantità di lavoro di altre operazioni (come l'incremento di indici o il confronto di valori)
- Supponiamo che N sia la dimensione della lista

Analisi delle prestazioni del SelectionSort (2)

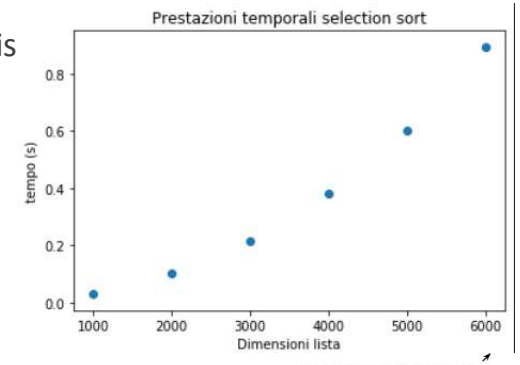
- Supponiamo che N sia la dimensione della lista
- Per prima cosa si deve trovare il più piccolo di N numeri: questo richiede la visita degli N elementi della lista
- Poi si scambiano gli elementi, operazione che richiede due visite
- Nel passo successivo, per trovare il minimo, si devono visitare N-1 elementi della lista. Nel passo ancora successivo, N-2. Finché non si arriva all'ultimo passo in cui si visitano solo 2 elementi tra i quali si deve trovare il minimo
- Ciascun passo richiede 2 visite per scambiare gli elementi, quindi il numero totale delle visite ris

$$N + 2 + (N-1) + 2 + (N-2) + 2 + \cdots + 2 = (N + (N-1) + \cdots + 2) + (N-1) * 2$$

$$= (2 + \cdots + (N-1) + N) + (N-1) * 2$$

$$= \frac{N(N-1)}{2} - 1 + (N-1) * 2$$

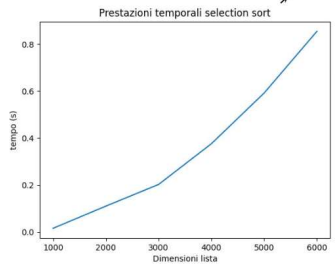
Perché: $1 + 2 + 3 + \cdots + (N-1) + N = \frac{N(N-1)}{2}$



- Sviluppando le moltiplicazioni e raccogliendo N come fattore comune, si trova che il numero delle visite è:

$$= \frac{1}{2}N^2 + \frac{5}{2}N - 3$$

- Come risultato si ottiene una quadratica, questo è il motivo per cui il grafico somiglia ad una parabola



Analisi delle prestazioni del SelectionSort (3)

- Numero delle visite è:

$$= \frac{1}{2}N^2 + \frac{5}{2}N - 3$$

- Semplificando ulteriormente l'analisi, in caso di dimensioni elevate della lista, ad esempio 1000 (o 2000), si ha che:

- $\frac{1}{2}N^2 = 500000 \quad e \quad \frac{5}{2}N - 3 = 1666{,}6$

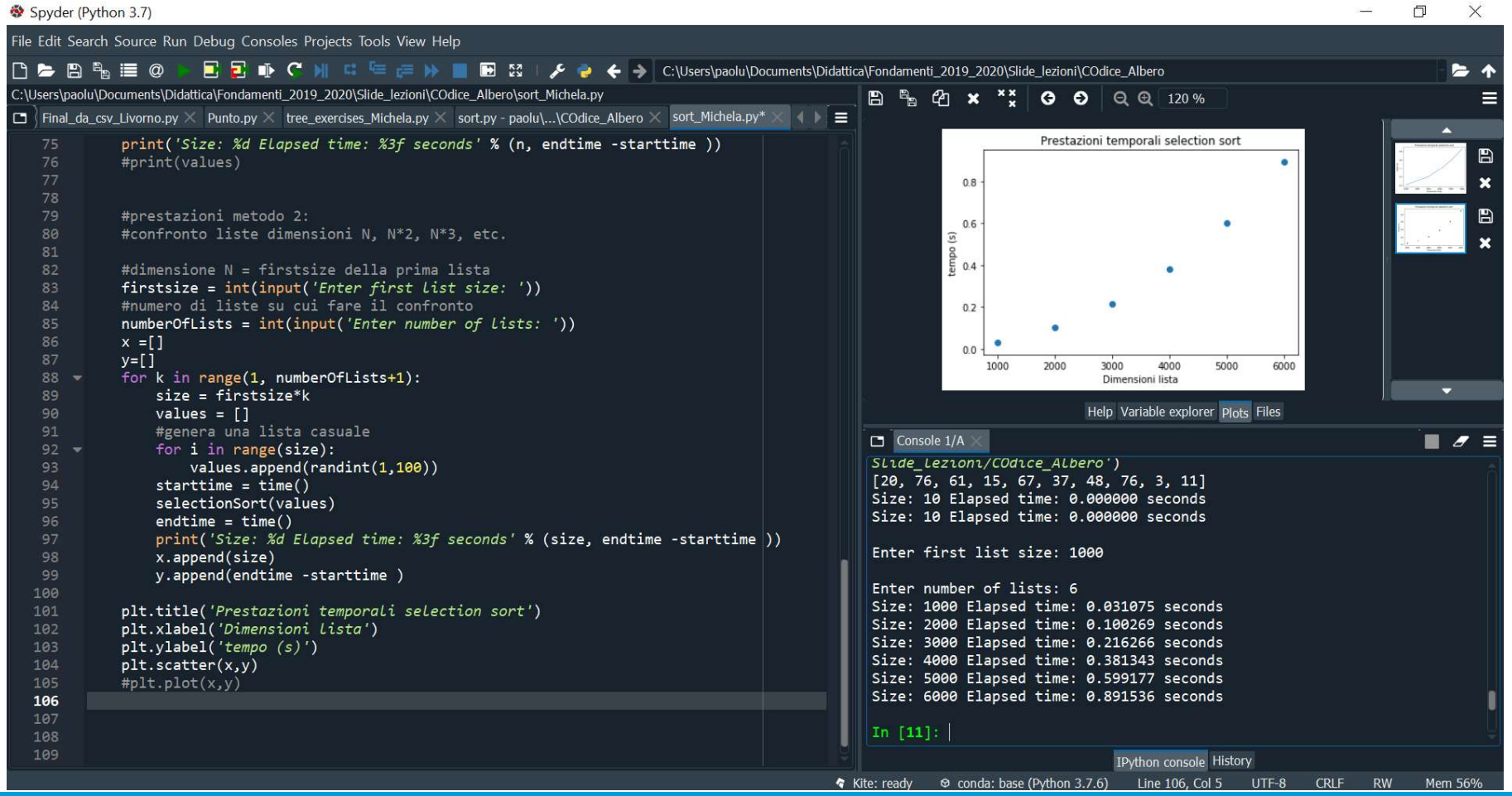
- Si nota che il fattore $(\frac{5}{2}N - 3)$ Non influisce più di tanto nel calcolo generale rispetto a $\frac{1}{2}N^2$ e può quindi essere ignorato, così come il fattore costante ½ (di $\frac{1}{2}N^2$)

- Possiamo ad esempio confrontare l'ordinamento di una lista di 2000 elementi con quello di una lista di 1000 elementi, ottenendo come risultato che per ordinare la lista di 4000 ci vuole un numero di visite pari a 4 volte quelle necessarie per una lista di 1000 elementi:

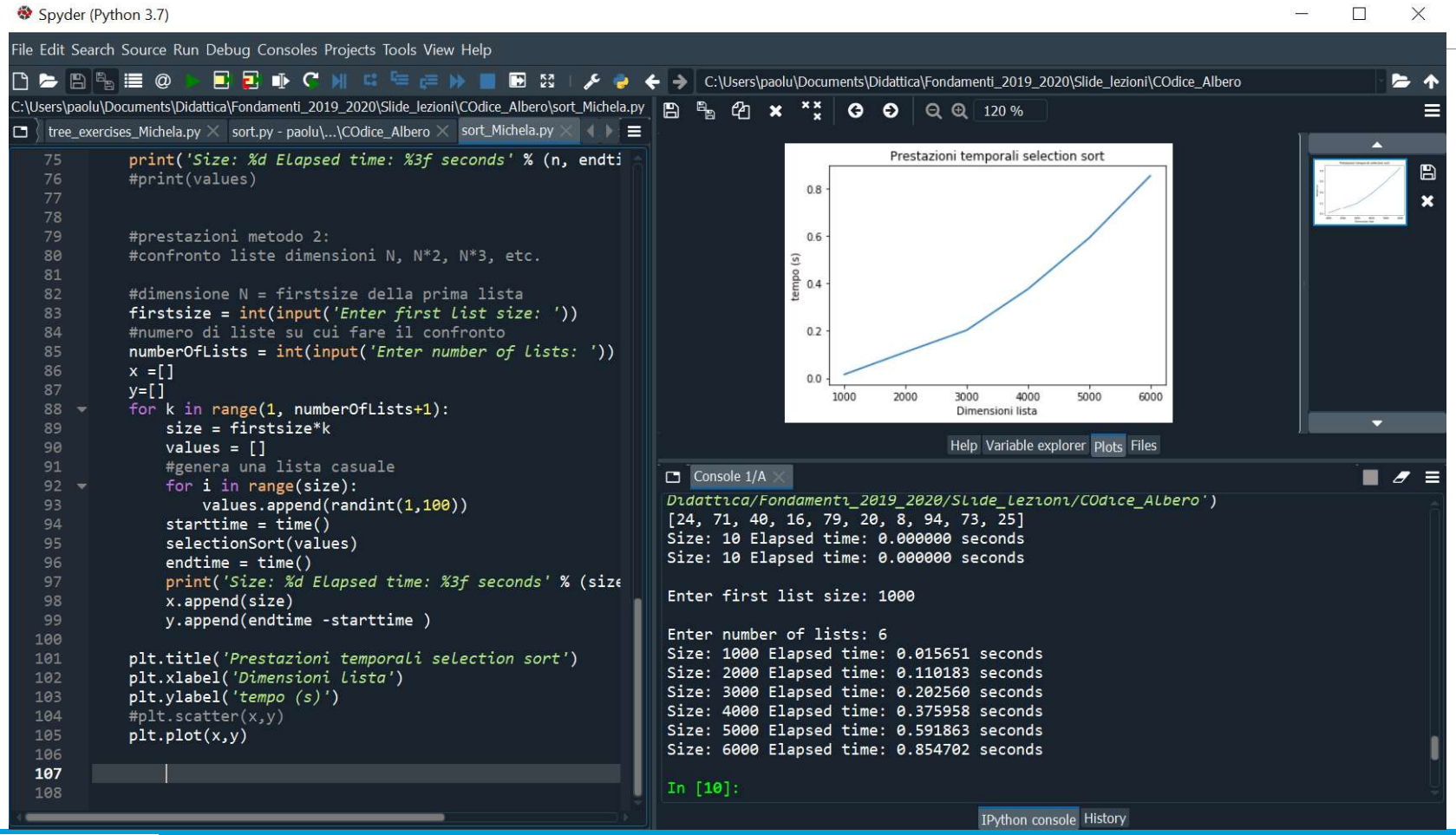
$\left(\frac{\frac{1}{2}2000^2}{\frac{1}{2}1000^2}\right) = 4$, da cui si può dire che 'il numero delle visite è dell'ordine di N^2 , ovvero **O(N^2)**

Come ottenere i grafici visti con Python

Graphics with matplotlib: visualization from Spyder (1)

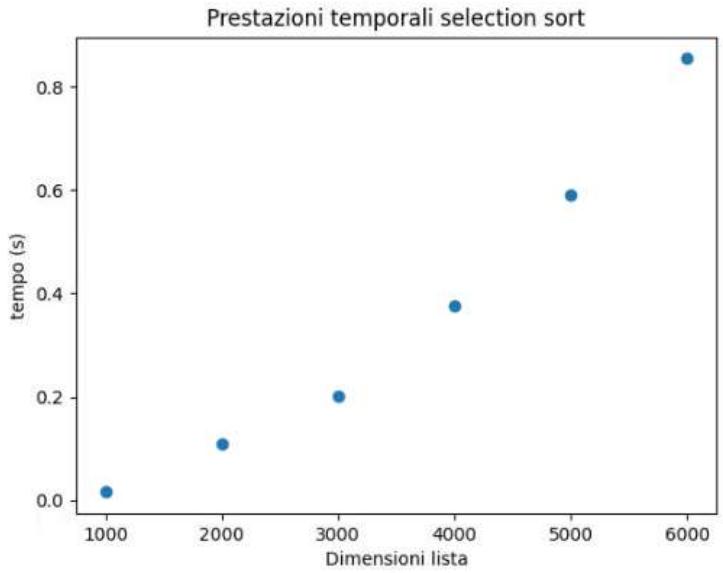


Graphics with matplotlib: visualization from Spyder (2)

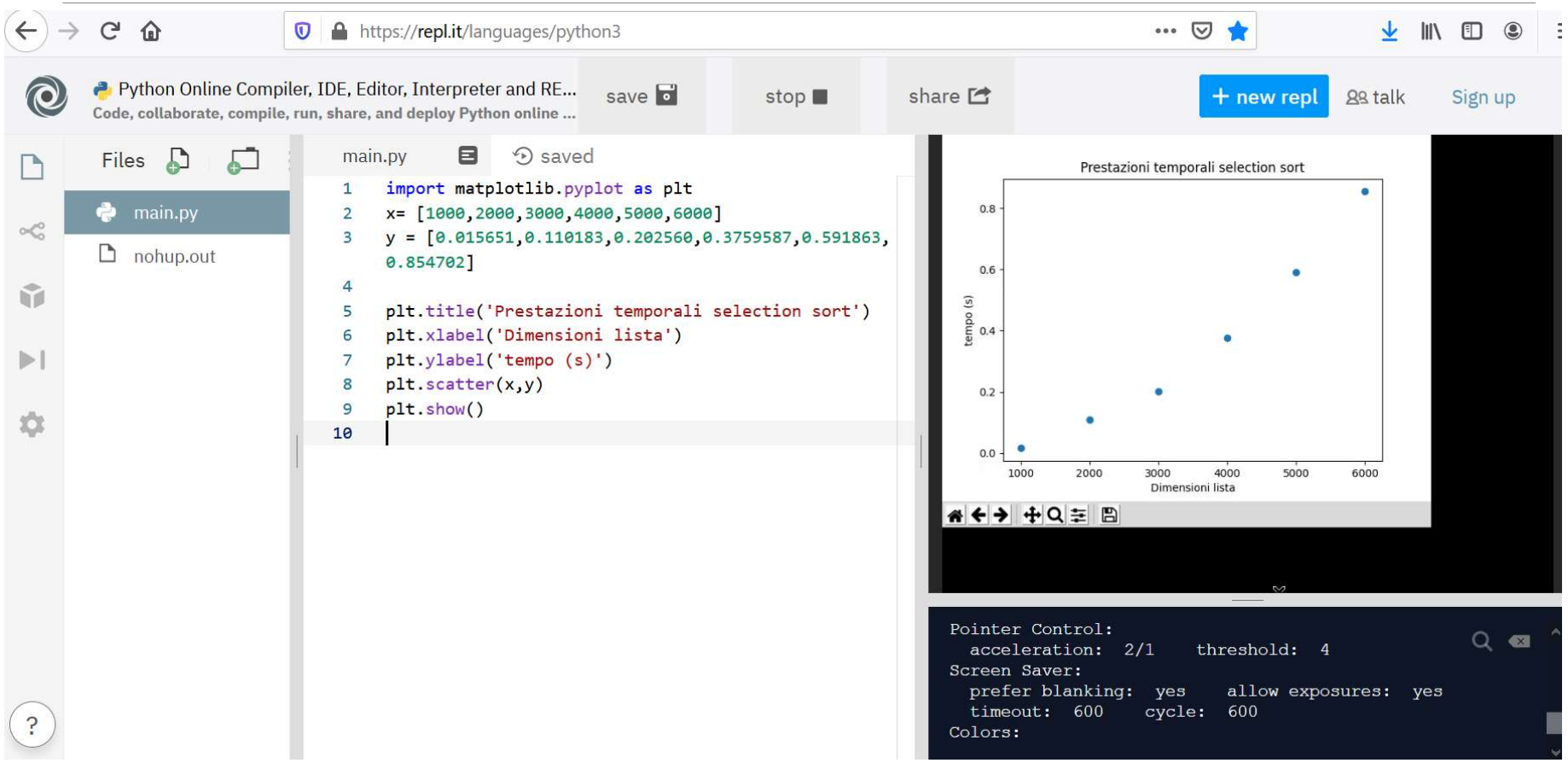


Usare matplotlib: alcuni esempi – type: scatter (1)

```
import matplotlib.pyplot as plt
x= [1000,2000,3000,4000,5000,6000]
y = [0.015651,0.110183,0.202560,0.3759587,0.591863,0.854702]

plt.title('Prestazioni temporali selection sort')
plt.xlabel('Dimensioni lista')
plt.ylabel('tempo (s)')
plt.scatter(x,y)
plt.show()
```

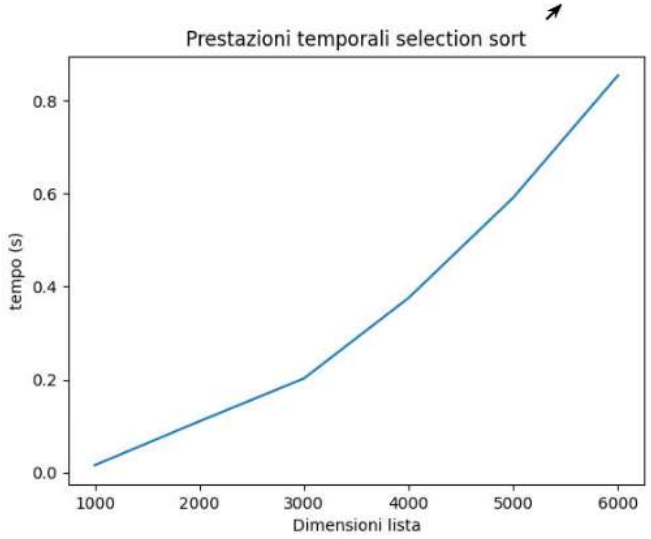


Usare matplotlib: alcuni esempi – type: scatter (2)

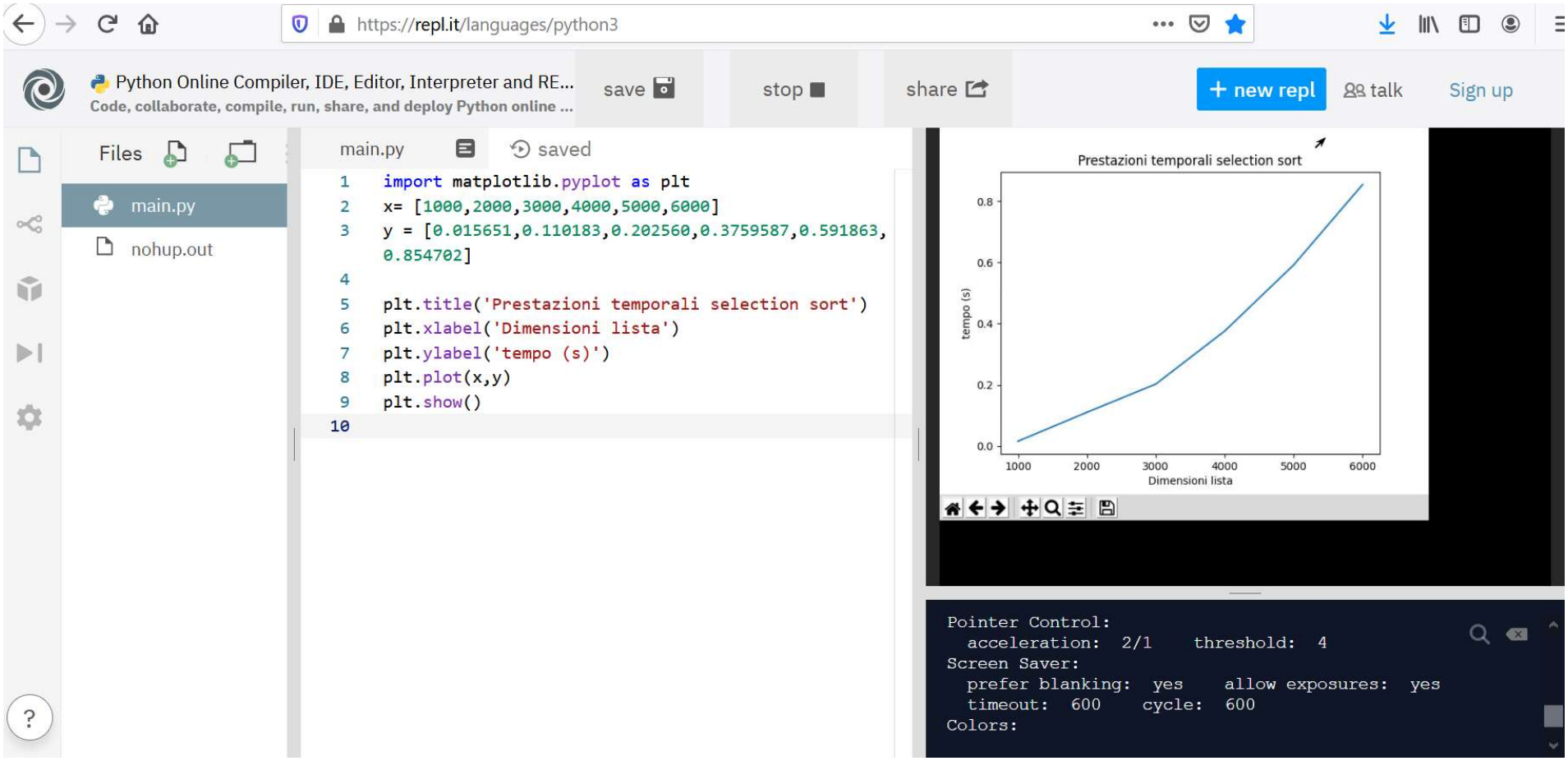


Usare matplotlib: alcuni esempi – type: plot (1)

```
import matplotlib.pyplot as plt
x= [1000,2000,3000,4000,5000,6000]
y = [0.015651,0.110183,0.202560,0.3759587,0.591863,0.854702]

plt.title('Prestazioni temporali selection sort')
plt.xlabel('Dimensioni lista')
plt.ylabel('tempo (s)')
plt.plot(x,y)
plt.show()
```



Usare matplotlib: alcuni esempi – type: plot (2)



Usare matplotlib e selectionsort (1)

```
import matplotlib.pyplot as plt
from random import randint
from time import time

def minimuPosition(values, start):
            minPos = start
            for i in range(start + 1, len(values)):
                        if values[i] < values[minPos]:
                        minPos = i
            return minPos

def selectionSort(values):
            for i in range(len(values)):
                        minPos = minimuPosition(values, i)
                        temp = values[minPos] #scambia i due elementi
                        values[minPos] = values[i]
            values[i] = temp
[…]
```

Usare matplotlib e selectionsort (2)

```
import matplotlib.pyplot as plt

from random import randint

from time import time

def minimuPosition(values, start):

    minPos = start

    for i in range(start + 1, len(values)):

          if values[i] < values[minPos]:

                minPos = i

    return minPos

def selectionSort(values):

    for i in range(len(values)):

          minPos = minimuPosition(values, i)

          temp = values[minPos] #scambia i due elementi

          values[minPos] = values[i]

          values[i] = temp

#prestazioni metodo 2: confronto liste dimensioni N, N*2, N*3, etc.

firstsize = int(input('Enter first list size: ‘)) #dimensione N = firstsize della prima lista

numberOfLists = int(input('Enter number of lists: ‘)) #numero di liste su cui fare il confronto

x =[]

y=[]

for k in range(1, numberOfLists+1):

      size = firstsize*k

      values = []

      for i in range(size): #genera una lista casuale

           values.append(randint(1,100))

      starttime = time()

      selectionSort(values)

      endtime = time()

      print('Size: %d Elapsed time: %3f seconds' % (size, endtime -
            starttime ))

      x.append(size)

      y.append(endtime -starttime )

plt.title('Prestazioni temporali selection sort')

plt.xlabel('Dimensioni lista')

plt.ylabel('tempo (s)')

plt.scatter(x,y) #plt.plot(x,y)
```

Usare matplotlib e selectionsort (3)

Spyder (Python 3.7)

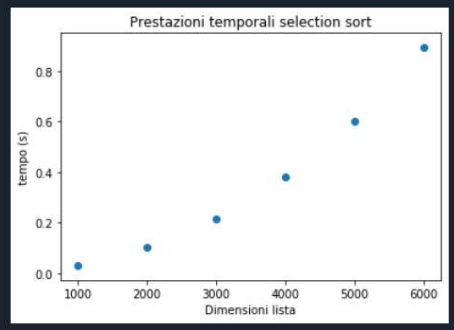
File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\paolu\Documents\Didattica\Fondamenti_2019_2020\Slide_lezioni\COdice_Albero

C:\Users\paolu\Documents\Didattica\Fondamenti_2019_2020\Slide_lezioni\COdice_Albero\sort_Michela.py

Final_da_csv_Livorno.py | Punto.py | tree_exercises_Michela.py | sort.py - paolu\...\COdice_Albero | sort_Michela.py*

```
75      print('Size: %d Elapsed time: %3f seconds' % (n, endtime -starttime ))
76      #print(values)
77
78
79      #prestazioni metodo 2:
80      #confronto liste dimensioni N, N*2, N*3, etc.
81
82      #dimensione N = firstsize della prima lista
83      firstsize = int(input('Enter first list size: '))
84      #numero di liste su cui fare il confronto
85      numberOfLists = int(input('Enter number of lists: '))
86      x =[]
87      y=[]
88      for k in range(1, numberOfLists+1):
89          size = firstsize*k
90          values = []
91          #genera una lista casuale
92          for i in range(size):
93              values.append(randint(1,100))
94          starttime = time()
95          selectionSort(values)
96          endtime = time()
97          print('Size: %d Elapsed time: %3f seconds' % (size, endtime -starttime ))
98          x.append(size)
99          y.append(endtime -starttime )
100
101     plt.title('Prestazioni temporali selection sort')
102     plt.xlabel('Dimensioni lista')
103     plt.ylabel('tempo (s)')
104     plt.scatter(x,y)
105     #plt.plot(x,y)
106
107
108
109
```



Help | Variable explorer | Plots | Files

Console 1/A

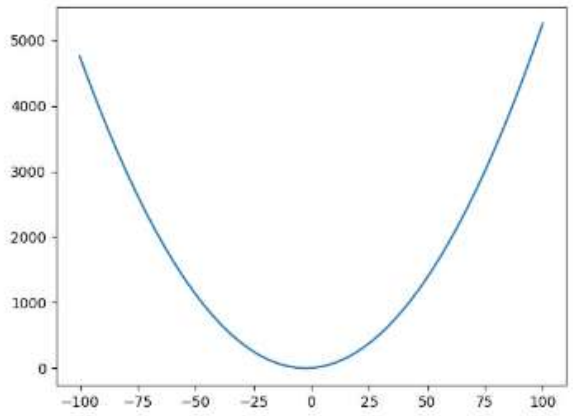
```
Slide_lezioni/COdice_Albero')
[20, 76, 61, 15, 67, 37, 48, 76, 3, 11]
Size: 10 Elapsed time: 0.000000 seconds
Size: 10 Elapsed time: 0.000000 seconds

Enter first list size: 1000

Enter number of lists: 6
Size: 1000 Elapsed time: 0.031075 seconds
Size: 2000 Elapsed time: 0.100269 seconds
Size: 3000 Elapsed time: 0.216266 seconds
Size: 4000 Elapsed time: 0.381343 seconds
Size: 5000 Elapsed time: 0.599177 seconds
Size: 6000 Elapsed time: 0.891536 seconds

In [11]:
```

IPython console | History

Kite: ready | conda: base (Python 3.7.6) | Line 106, Col 5 | UTF-8 | CRLF | RW | Mem 56%

Uso di numphy

```
import numpy as np
import matplotlib.pyplot as plt

if __name__ == "__main__":
   # sia la quadatica: (1/2)*N^2 + (5/2)*N + 3
   # 3 + 2.5*x + 0.5 x^2
   # K + A*x + B*x^2
   K = 3
   A = 2.5
   B = 0.5
   x = np.linspace(-100,100.0,50,endpoint=True)
   y = K + A*x + B*x*x
   plt.plot(x,y)
   plt.show()
```



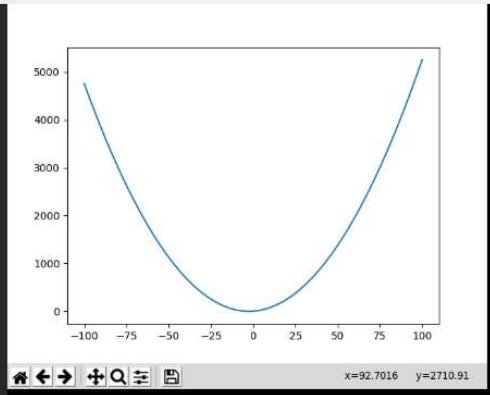
https://repl.it/languages/python3

Python Online Compiler, IDE, Editor, Interpreter and REPL
Code, collaborate, compile, run, share, and deploy Python online from ...

save | stop | share | + new repl | talk | Sign up

main.py — saved

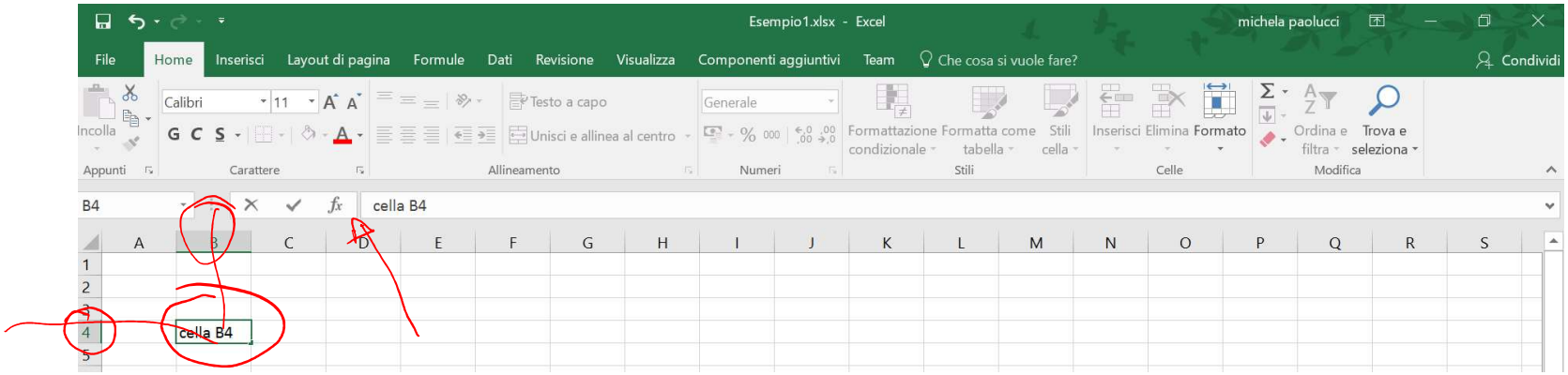
```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  if __name__ == "__main__":
5      # sia la quadatica: (1/2)*N^2 + (5/2)*N + 3
6      # 3 + 2.5*x + 0.5 x^2
7      # K + A*x + B*x^2
8      K = 3
9      A = 2.5
10     B = 0.5
11     x = np.linspace(-100,100.0,50,endpoint=True)
12     y =  K + A*x + B*x*x
13     plt.plot(x,y)
14     plt.show()
15
```




```
Keys:  off
  auto repeat delay:  660    repeat rate:  25
  auto repeating keys:  00feffffdffffbbf
                        fadfffffffdfe5ef
                        fffffffffffffff
                        fffffffffffffff
  bell percent:  50    bell pitch:  400    bell duration:  100
Pointer Control:
  acceleration:  2/1    threshold:  4
Screen Saver:
  prefer blanking:  yes    allow exposures:  yes
  timeout:  600    cycle:  600
```

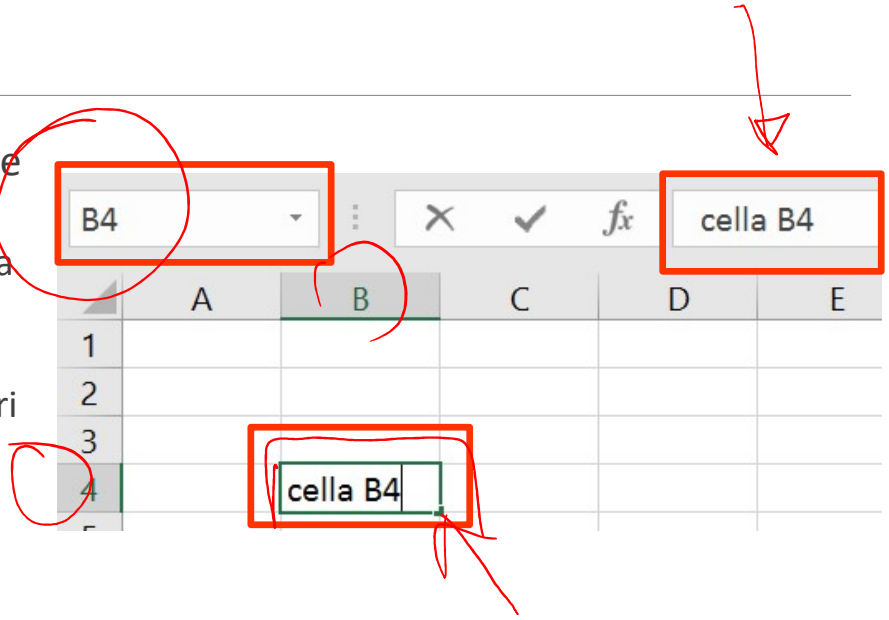
EXCEL

Introduzione (1)



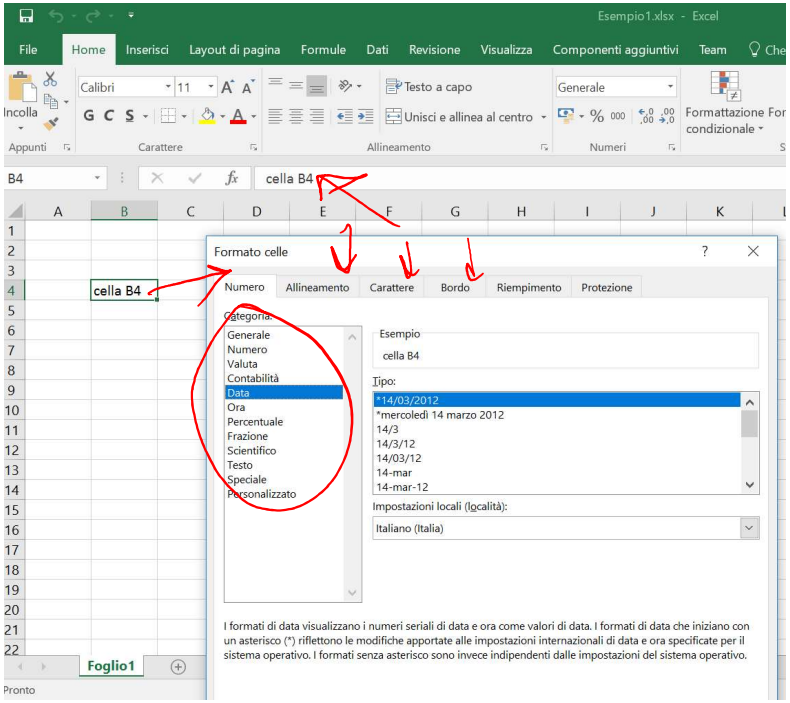
Celle (1)

- Ogni foglio di Excel è diviso in righe e colonne che hanno un nome:
  - il nome delle righe corrisponde al numero della riga sul foglio
  - Il nome delle colonne ad una lettera alfabetica (quando sono finite le lettere, si usano due caratteri invece di uno, per esempio AX)
- I nomi delle righe e delle colonne sono detti INTESTAZIONI
- L'intersezione tra una riga e una colonna è detto CELLA
- Se si decide di lavorare su una cella, si evidenzia rispetto alle altre



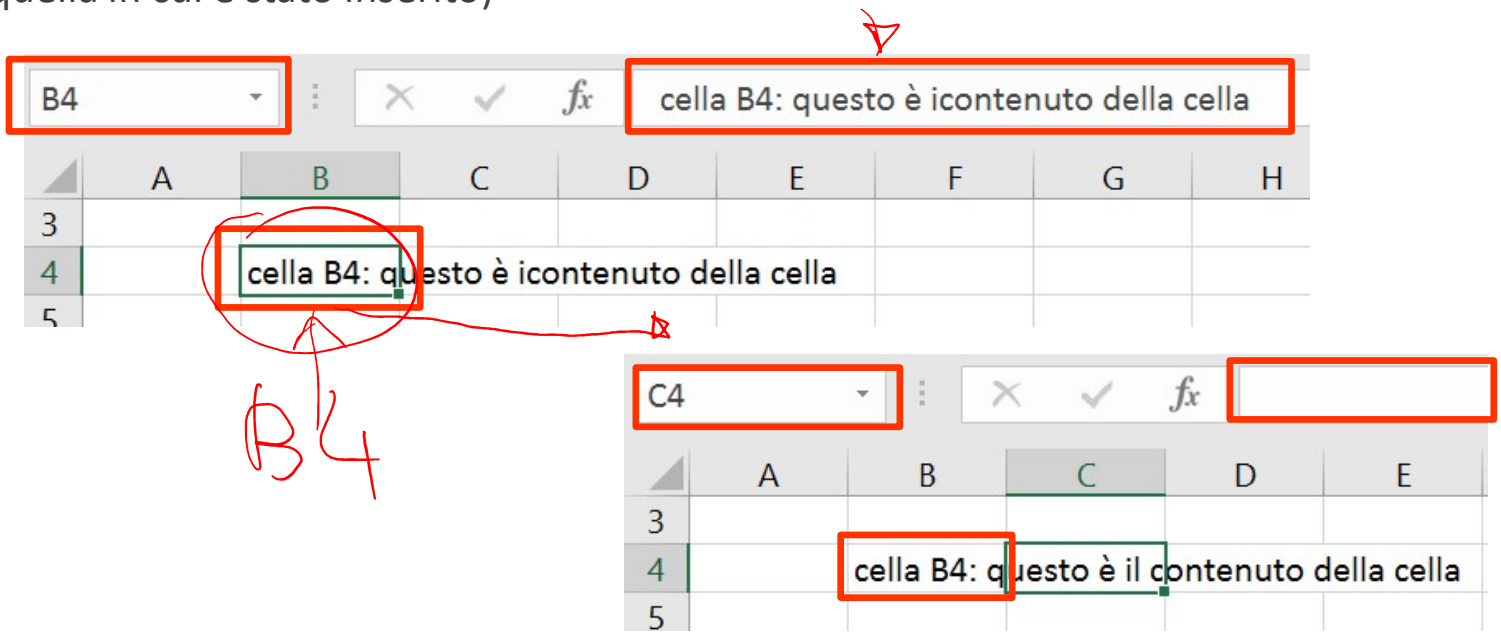
Celle (2)

- Per spostarsi tra le celle si usa il tasto ‘tabulazione’ (il cursore si sposta di una cella a destra)
- Ogni cella può contenere diversi tipi di informazioni:
  - Date, numeri, testo, etc.
- E’ possibile impostare il formato di ogni cella (Selezionare cella > tasto destro > Formato celle), ovvero decidere il modo in cui visualizzare un dato:
  - Come scrivere una data (14/03/2012 o 14/03, …)
  - Bordo di una cella
  - Tipo di carattere
  - Etc.
- Dimensione delle celle
  - Le celle hanno una dimensione standard, se un testo ‘esce’ da tale dimensione e la cella alla sua destra è libera, allora excel lo visualizza per intero MA è sempre contenuto in una unica cella (quella in cui è stato inserito)



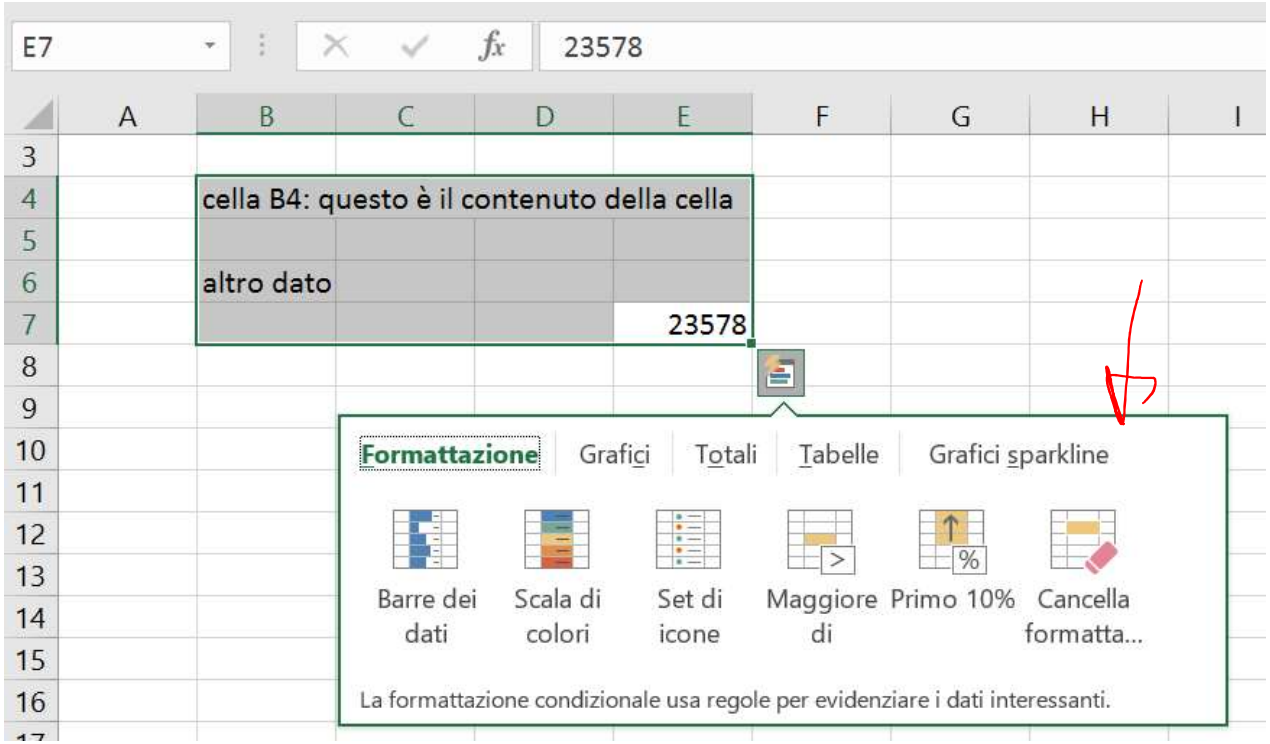
Celle (3)

- Dimensione delle celle
- Le celle hanno una dimensione standard, se un testo 'esce' da tale dimensione e la cella alla sua destra è libera, allora excel lo visualizza per intero MA è sempre contenuto in una unica cella (quella in cui è stato inserito)



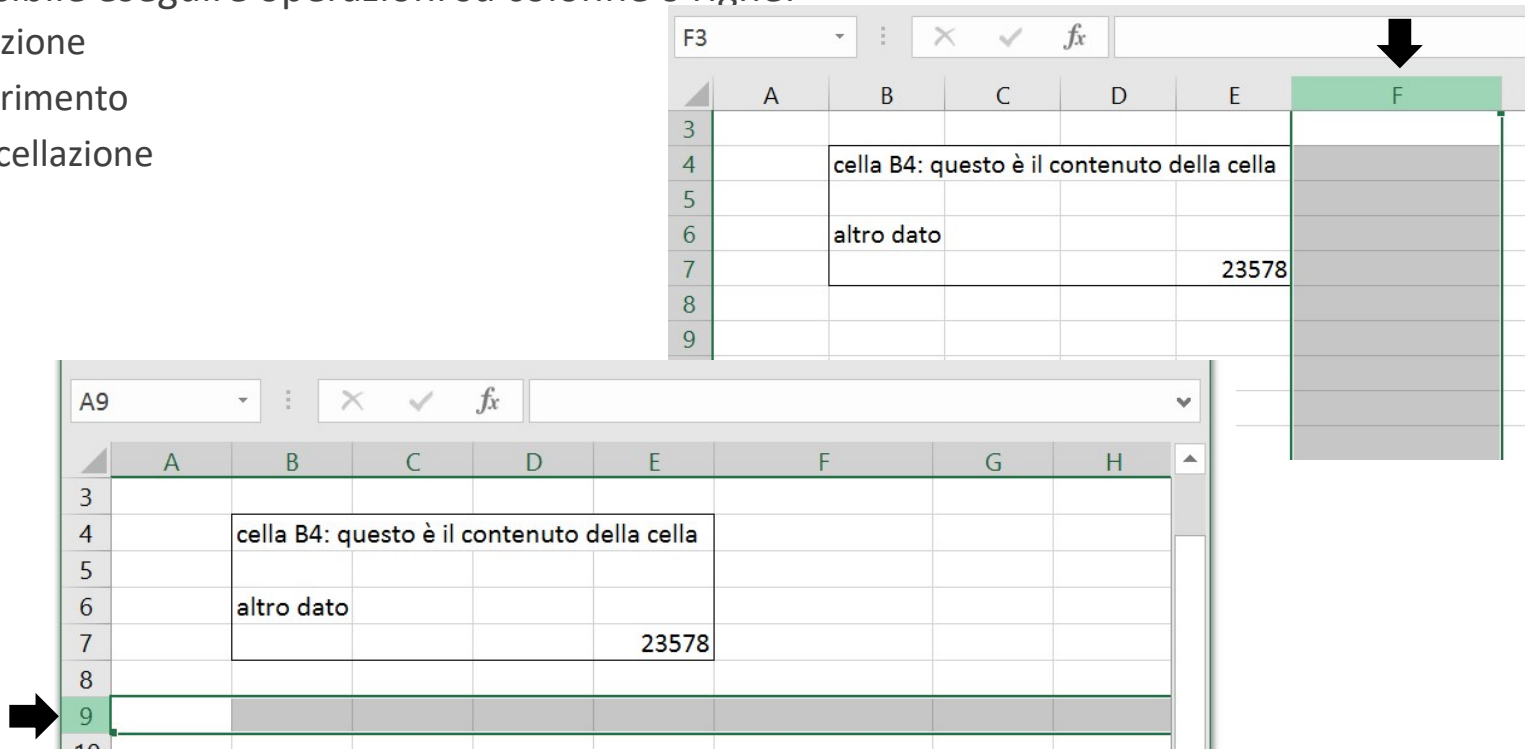
Selezione celle (1)

- E' possibile selezionare più celle insieme (Shift + frecce) per effettuare sulle celle selezionate delle operazioni (anche solo di visualizzazione, ad esempio mettere un bordo, etc.)



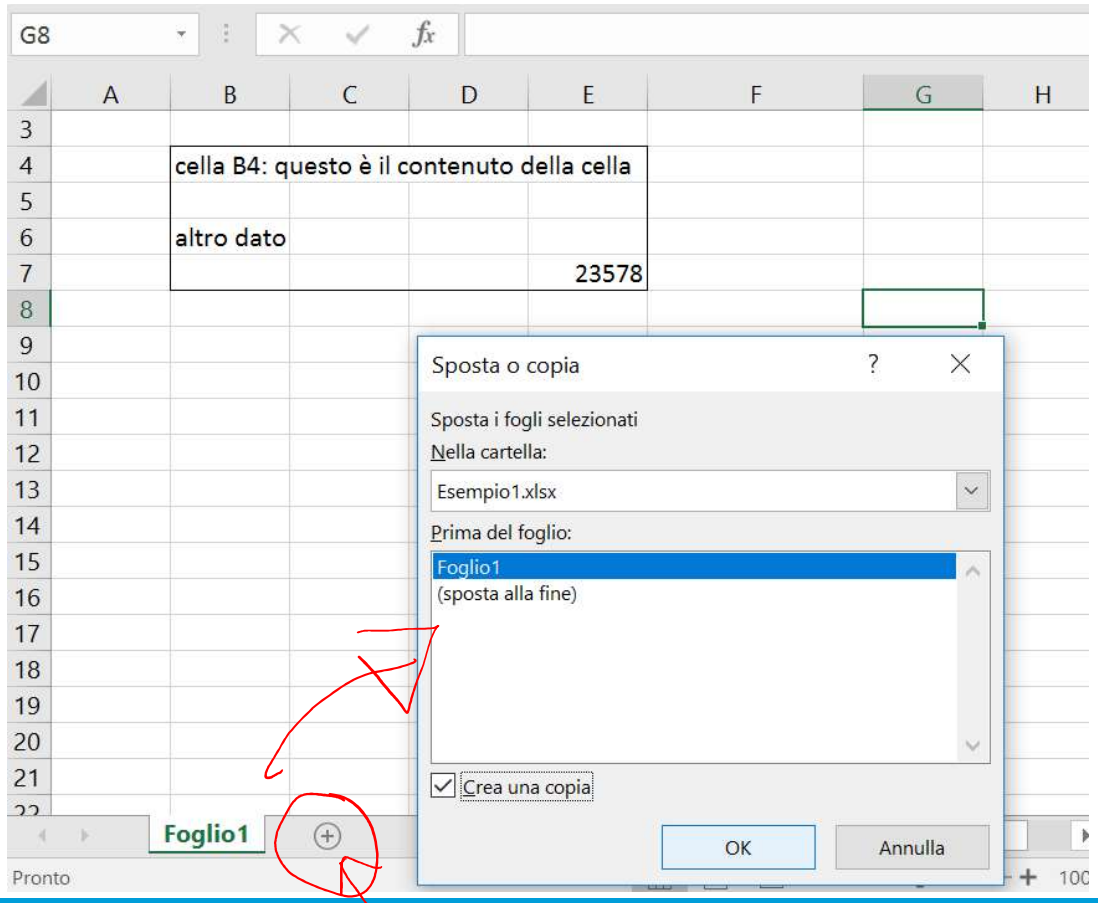
Selezione celle (2)

- E' possibile eseguire operazioni su colonne o righe:
  - Selezione
  - Inserimento
  - Cancellazione
  - Etc.



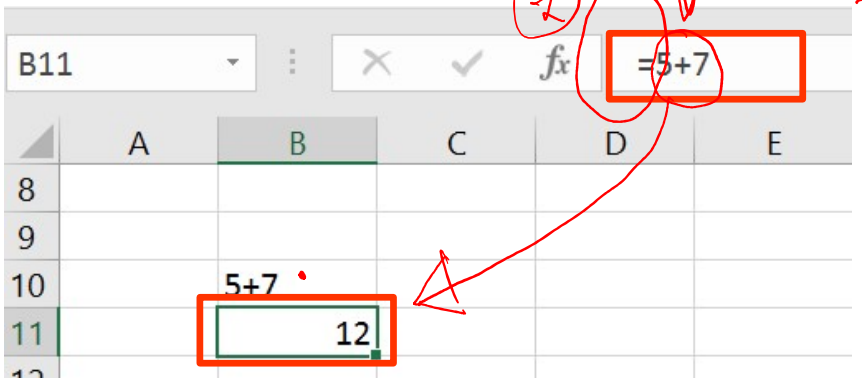
Fogli di lavoro

- Un file excel può contenere più fogli di lavoro
- Copia
- Aggiunta nuovo
- Etc.



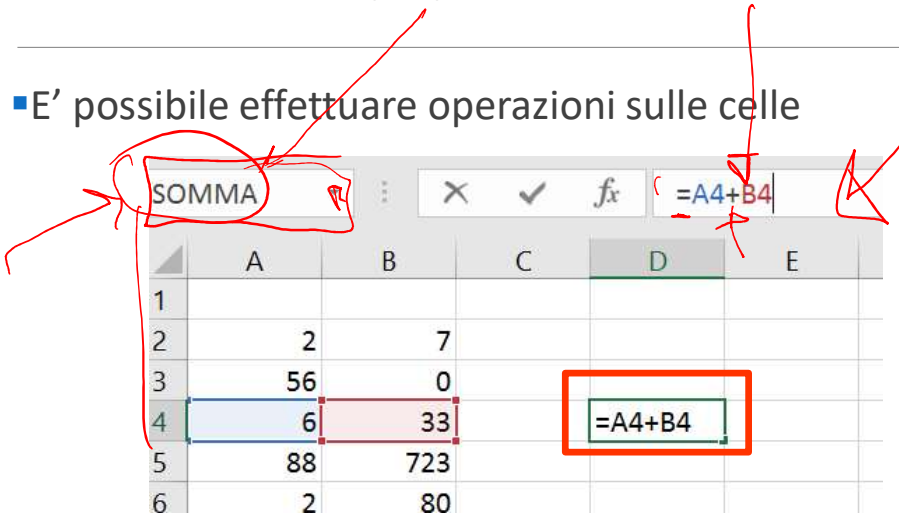
Formule (1)

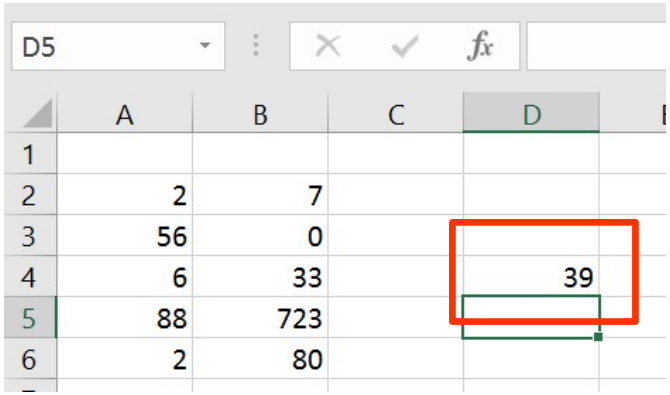
- Uno degli usi dei fogli di calcolo è quello di fornire il risultato di alcune operazioni
- Se si vuole eseguire la somma tra due numeri:
  - Digitare '=5+7' (senza spazi) e cliccare invio
- Si possono usare le varie operazioni:
  - Somma (+)
  - Sottrazione (-)
  - Moltiplicazione (*)
  - Divisione (/)
  - Elevamento a potenza (^)

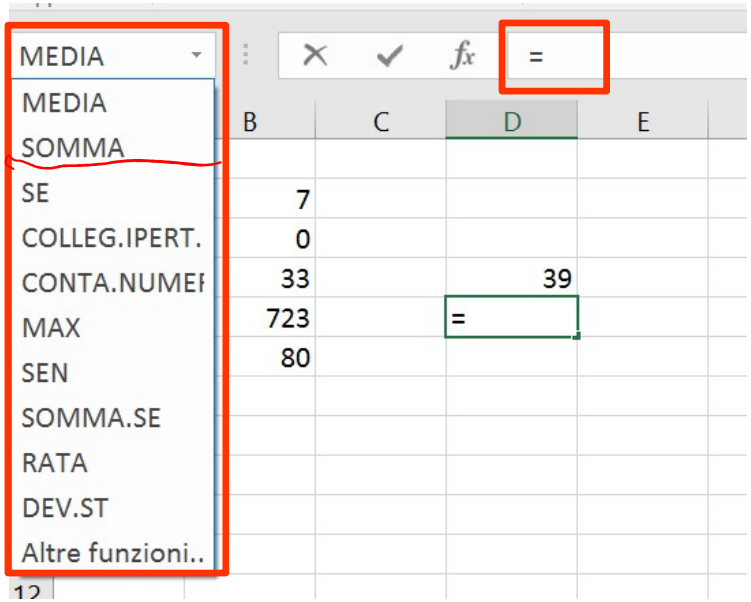


Formule (2)

- E' possibile effettuare operazioni sulle celle

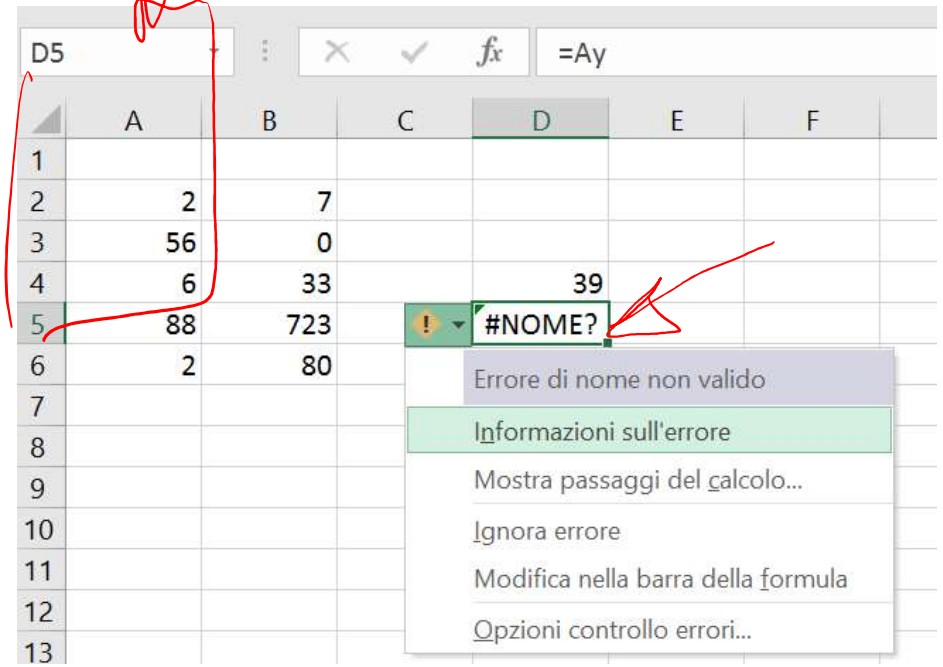




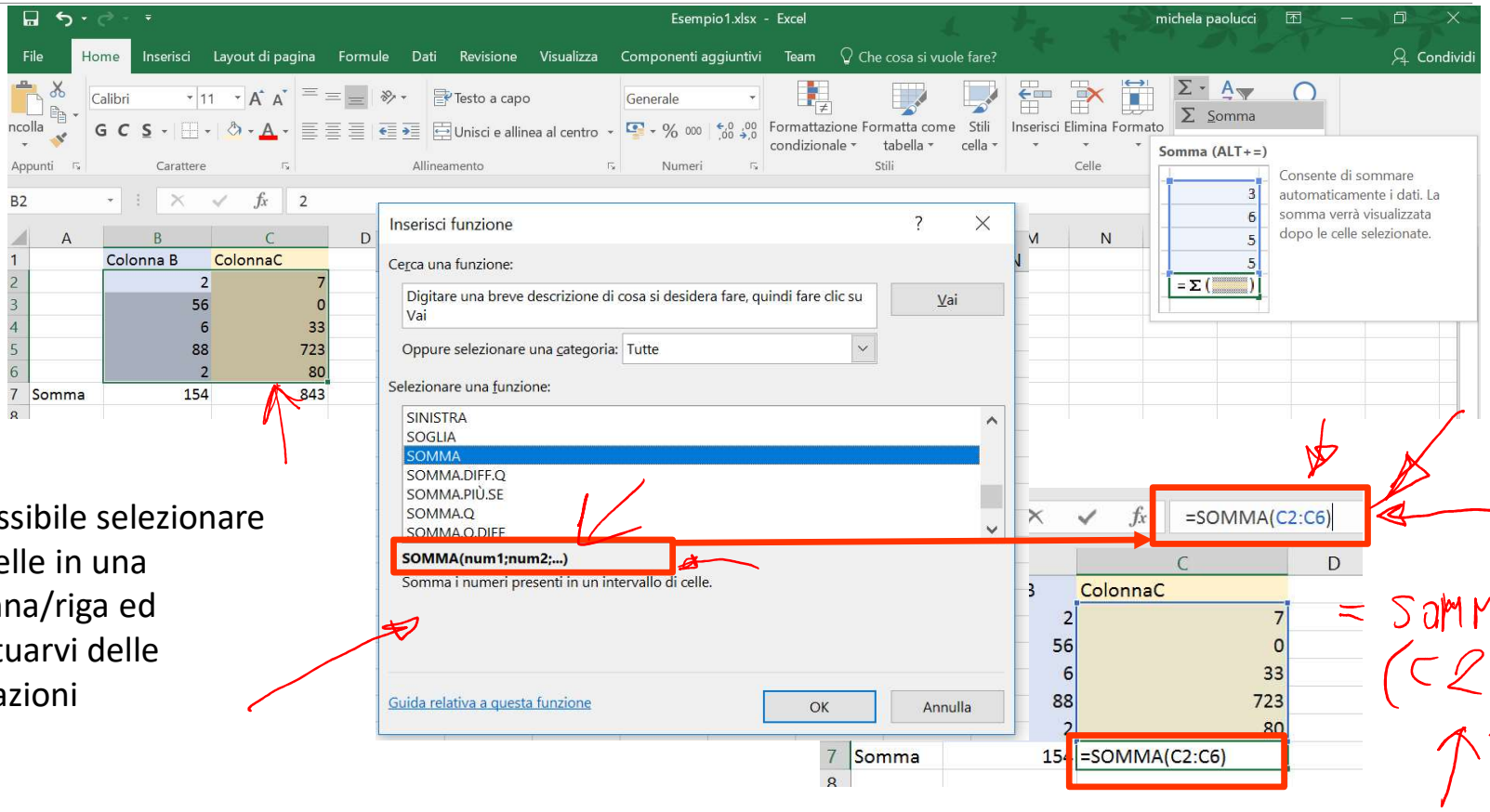


Formule (3)

- Se si scrive in modo errato una formula, nella cella corrispondente si trova la stringa '#NOME?' e una serie di possibili indicazioni relative al possibile tipo di errore effettuato



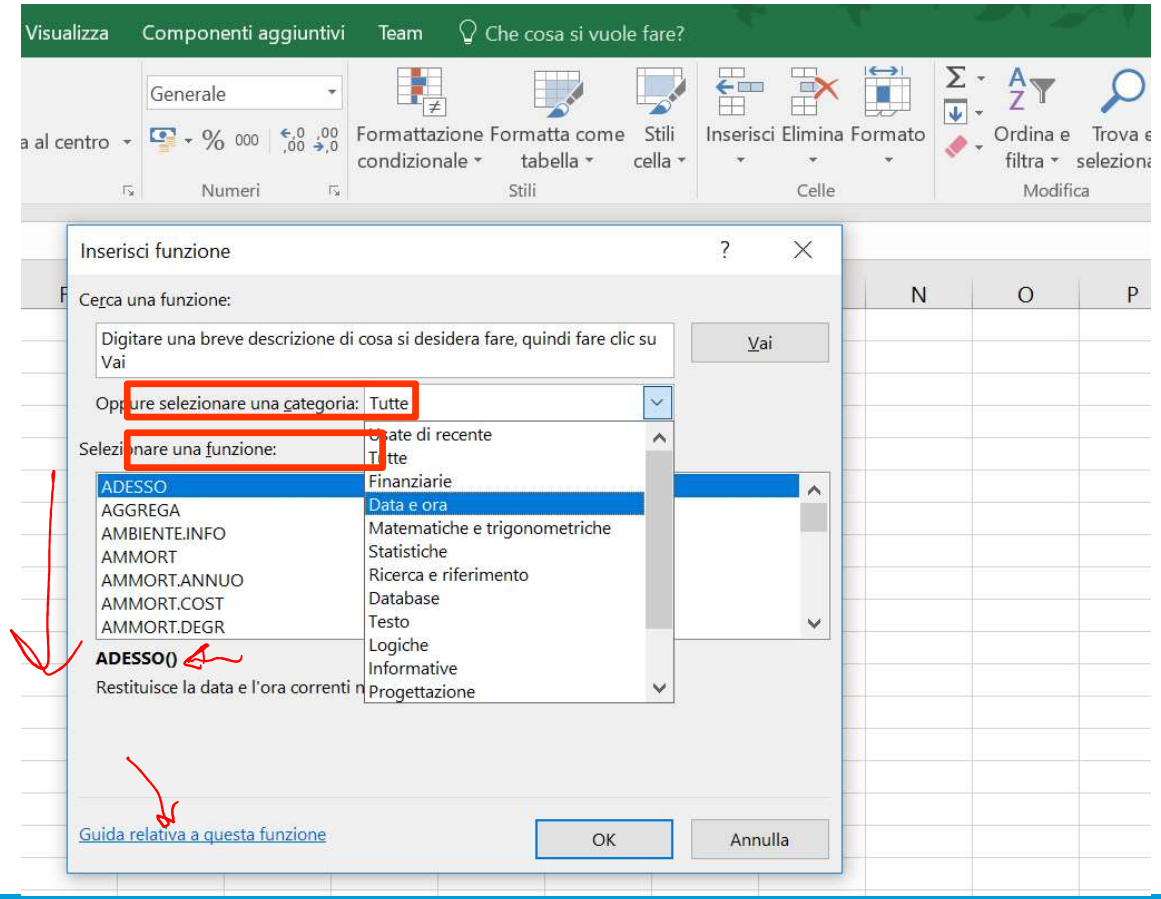
Formule (4)



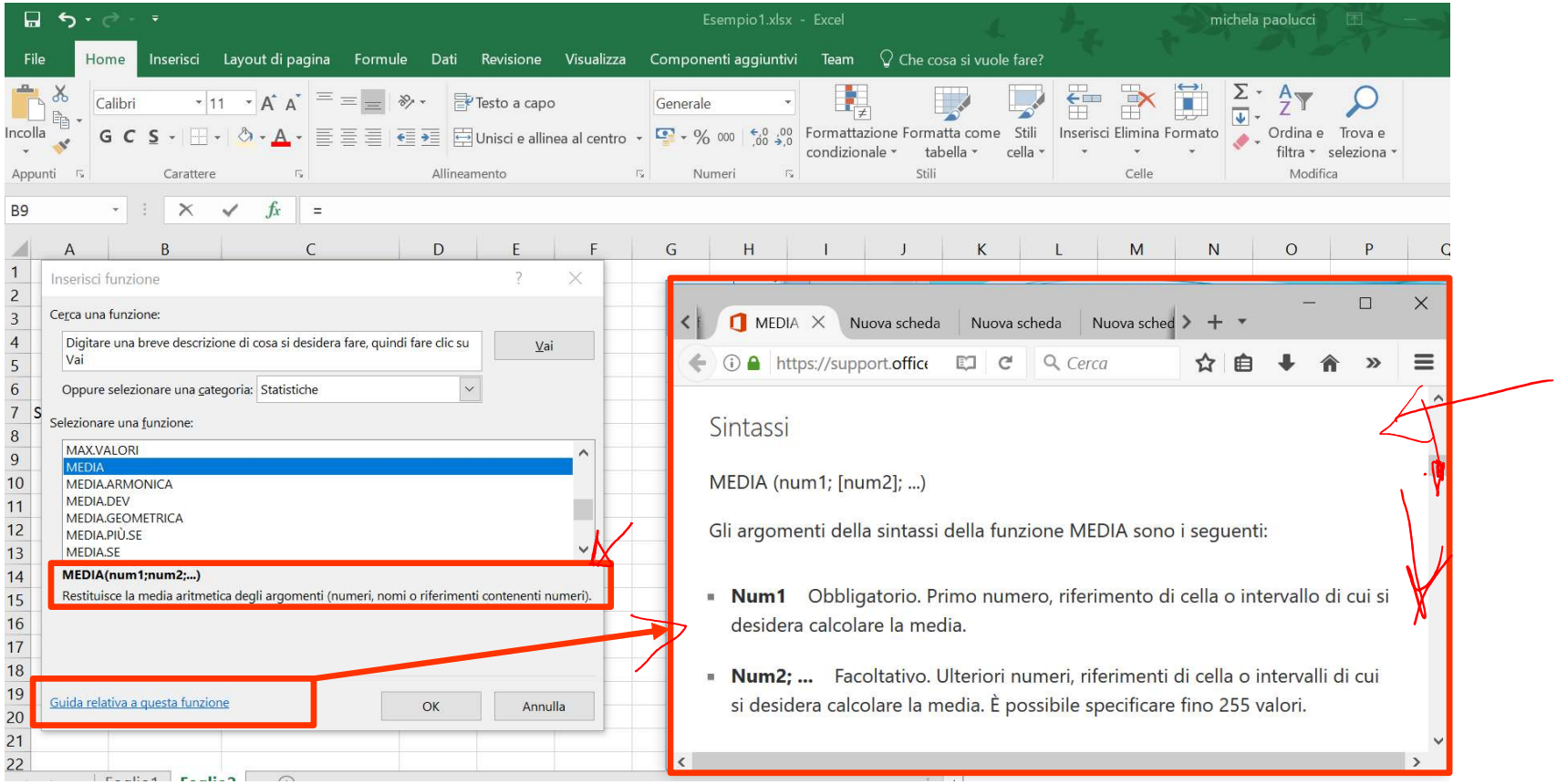
- E' possibile selezionare più celle in una colonna/riga ed effettuarvi delle operazioni

Funzioni (1)

- Excel prevede una serie di funzioni per la esecuzione di alcune operazioni
  - SOMMA
  - ARROTONDA
  - MEDIA
  - SEN
  - COS
  - Etc.

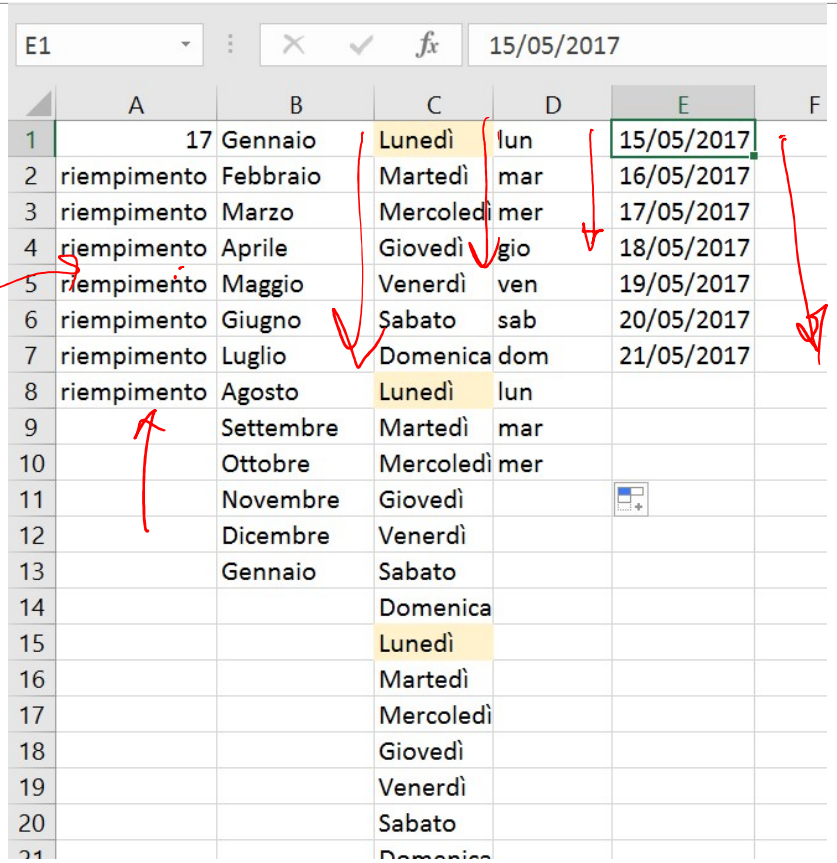


Funzioni (2)


Esempio1.xlsx - Excel
michela paolucci
File
Home
Inserisci
Layout di pagina
Formule
Dati
Revisione
Visualizza
Componenti aggiuntivi
Team
Che cosa si vuole fare?
Incolla
Calibri
11
Testo a capo
Unisci e allinea al centro
Generale
Formattazione condizionale
Formatta come tabella
Stili cella
Inserisci
Elimina
Formato
Ordina e filtra
Trova e seleziona
Appunti
Carattere
Allineamento
Numeri
Stili
Celle
Modifica
B9
=
Inserisci funzione
Cerca una funzione:
Digitare una breve descrizione di cosa si desidera fare, quindi fare clic su Vai
Vai
Oppure selezionare una categoria:
Statistiche
Selezionare una funzione:
MAX.VALORI
MEDIA
MEDIA.ARMONICA
MEDIA.DEV
MEDIA.GEOMETRICA
MEDIA.PIÙ.SE
MEDIA.SE
MEDIA(num1;num2;...)
Restituisce la media aritmetica degli argomenti (numeri, nomi o riferimenti contenenti numeri).
Guida relativa a questa funzione
OK
Annulla
MEDIA
Nuova scheda
Nuova scheda
https://support.offic
Cerca
Sintassi
MEDIA (num1; [num2]; ...)
Gli argomenti della sintassi della funzione MEDIA sono i seguenti:
Num1 Obbligatorio. Primo numero, riferimento di cella o intervallo di cui si desidera calcolare la media.
Num2; ... Facoltativo. Ulteriori numeri, riferimenti di cella o intervalli di cui si desidera calcolare la media. È possibile specificare fino 255 valori.

Selezionare/Copiare/Trascinare celle

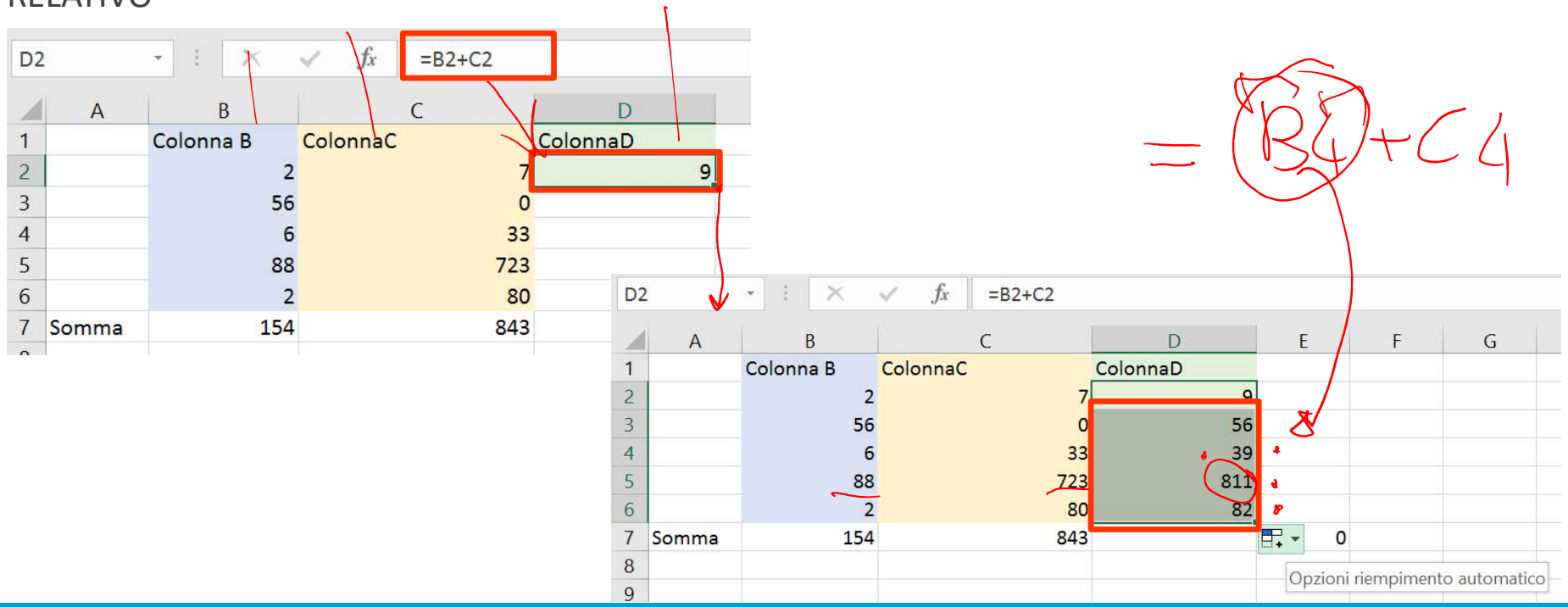
- Si possono selezionare celle e copiarle o tagliarle, incollandole in altre zone del foglio
- In alcuni casi predefiniti, il riempimento automatico compie azioni particolari
- Si selezionano una o più celle (in base alla operazione che si vuole fare), click tasto sinistro e si trascina



	A	B	C	D	E	F
1	17	Gennaio	Lunedì	lun	15/05/2017	
2	riempimento	Febbraio	Martedì	mar	16/05/2017	
3	riempimento	Marzo	Mercoledì	mer	17/05/2017	
4	riempimento	Aprile	Giovedì	gio	18/05/2017	
5	riempimento	Maggio	Venerdì	ven	19/05/2017	
6	riempimento	Giugno	Sabato	sab	20/05/2017	
7	riempimento	Luglio	Domenica	dom	21/05/2017	
8	riempimento	Agosto	Lunedì	lun		
9		Settembre	Martedì	mar		
10		Ottobre	Mercoledì	mer		
11		Novembre	Giovedì			
12		Dicembre	Venerdì			
13		Gennaio	Sabato			
14			Domenica			
15			Lunedì			
16			Martedì			
17			Mercoledì			
18			Giovedì			
19			Venerdì			
20			Sabato			

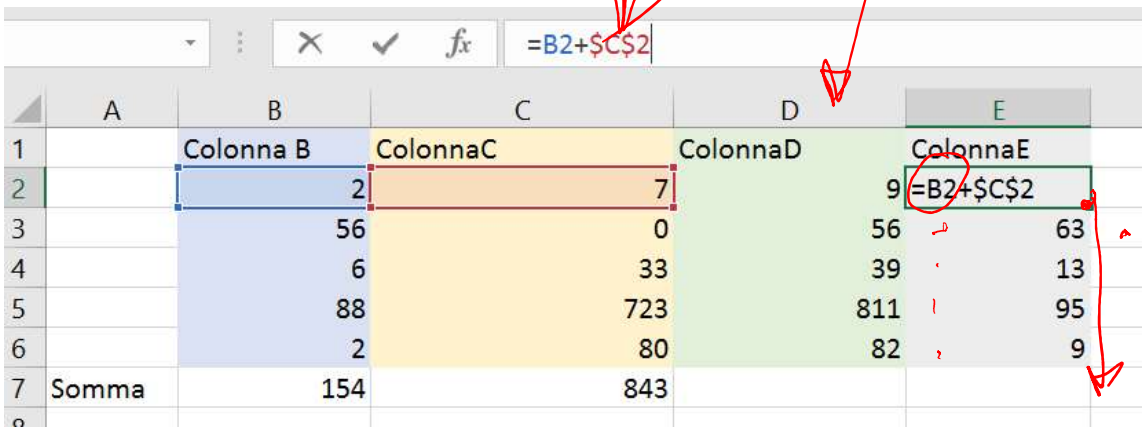
Riferimenti assoluti e relativi (1)

- E' possibile copiare anche le formule (sempre trascinando)
- Nelle celle D3-D6 excel memorizza la stessa formula contenuta in D2 per RIFERIMENTO RELATIVO



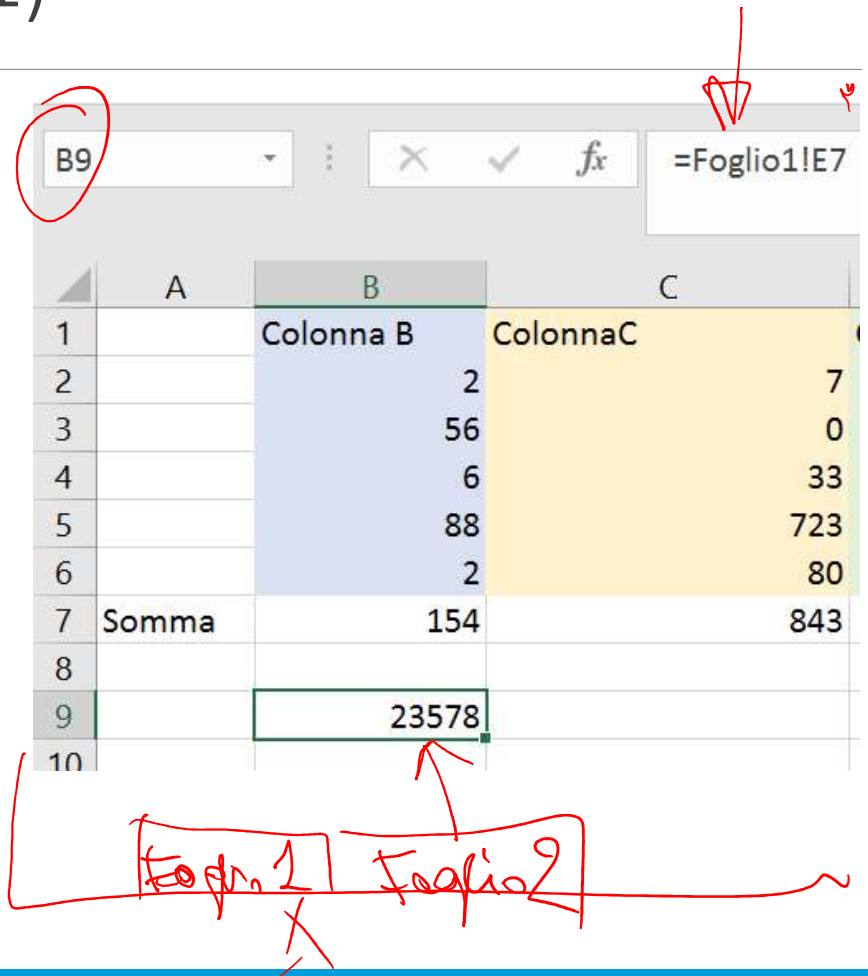
Riferimenti assoluti e relativi (2)

- Per scrivere un riferimento ASSOLUTO si usa il carattere '$'
- Formula:
  - B2+C2
- Formula con riferimento ASSOLUTO:
  - B2+$C$2
- Nelle celle D3-D6 excel memorizza la stessa formula contenuta in D2 per RIFERIMENTO ASSOLUTO



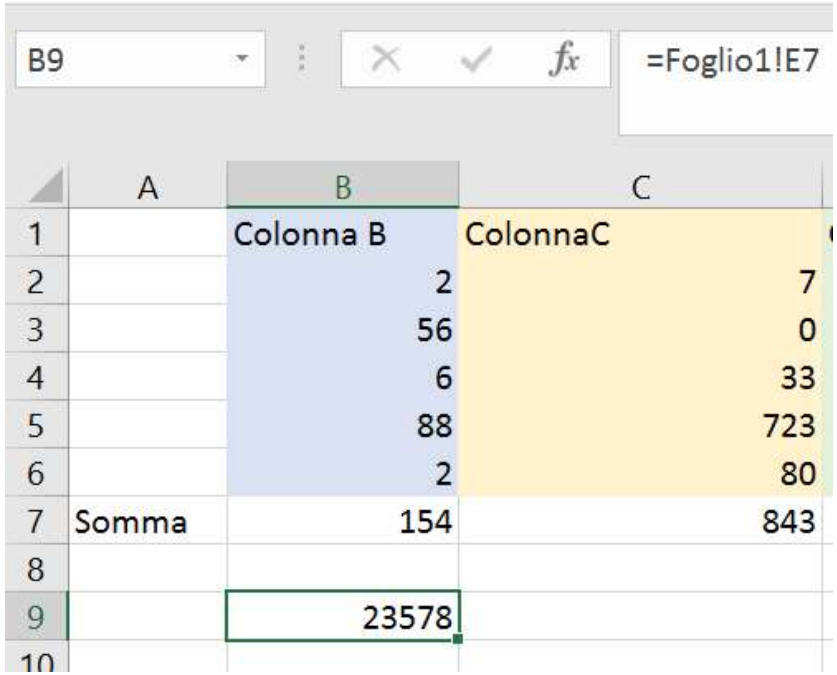
Collegamenti tra fogli (1)

- Nelle formule è possibile fare riferimento alle celle di altri fogli di lavoro, con la sintassi seguente:
  - = NOMEFOGLIO ! NOMECELLA
- E' possibile anche comporre formule:
- = NOMEFOGLIO1 ! NOMECELLA1 + NOMEFOGLIO2 ! NOMECELLA2



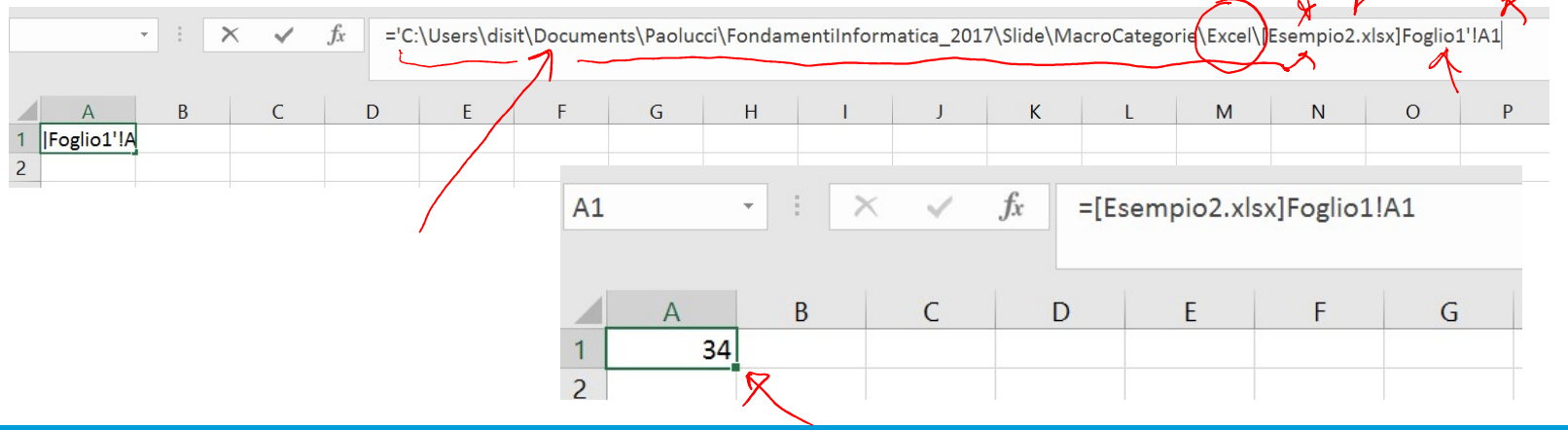
Collegamenti tra fogli (2)

- Nelle formule è possibile fare riferimento alle celle di altri fogli di lavoro, con la sintassi seguente:
  - = NOMEFOGLIO ! NOMECELLA



Collegamenti tra fogli (3)

- Quando c'è bisogno di riferirsi a celle che si trovano in altri file, si devono indicare le seguenti informazioni:
  - Nome del file di origine;
  - Percorso completo del file: nome della directory in cui si trova il file, per esempio C:\DOCUMENTI
  - Nome del foglio di lavoro
  - Riferimento alle celle
- ='PERCORSO\DIRECTORY\[FILE.xls]FOGLIO'!RIFERIMENTOCELLE

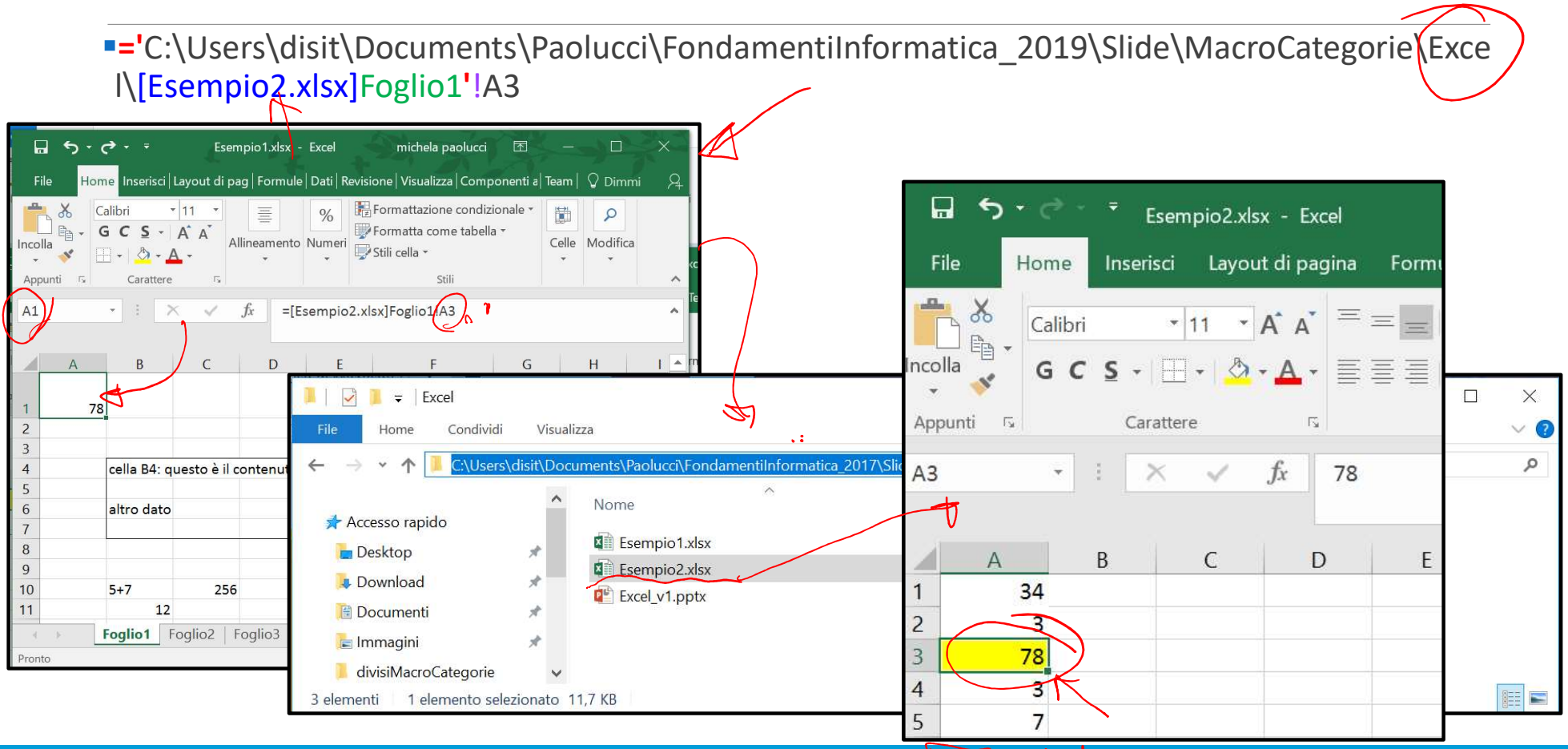


Collegamenti tra fogli (3)

- Quando c'è bisogno di riferirsi a celle che si trovano in altri file, si devono indicare le seguenti informazioni:
  - Nome del file di origine;
  - Percorso completo del file: nome della directory in cui si trova il file, per esempio C:\DOCUMENTI
  - Nome del foglio di lavoro
  - Riferimento alle celle
- =PERCORSO\DIRECTORY**[FILE.xls]**FOGLIO**!**RIFERIMENTOCELLE
- Esempio (cosa scrivere nella cella):
- **='**C:\Users\disit\Documents\Paolucci\FondamentiInformatica_2019\Slide\MacroCategorie\Excel\[Esempio2.xlsx]Foglio1**'!**A1

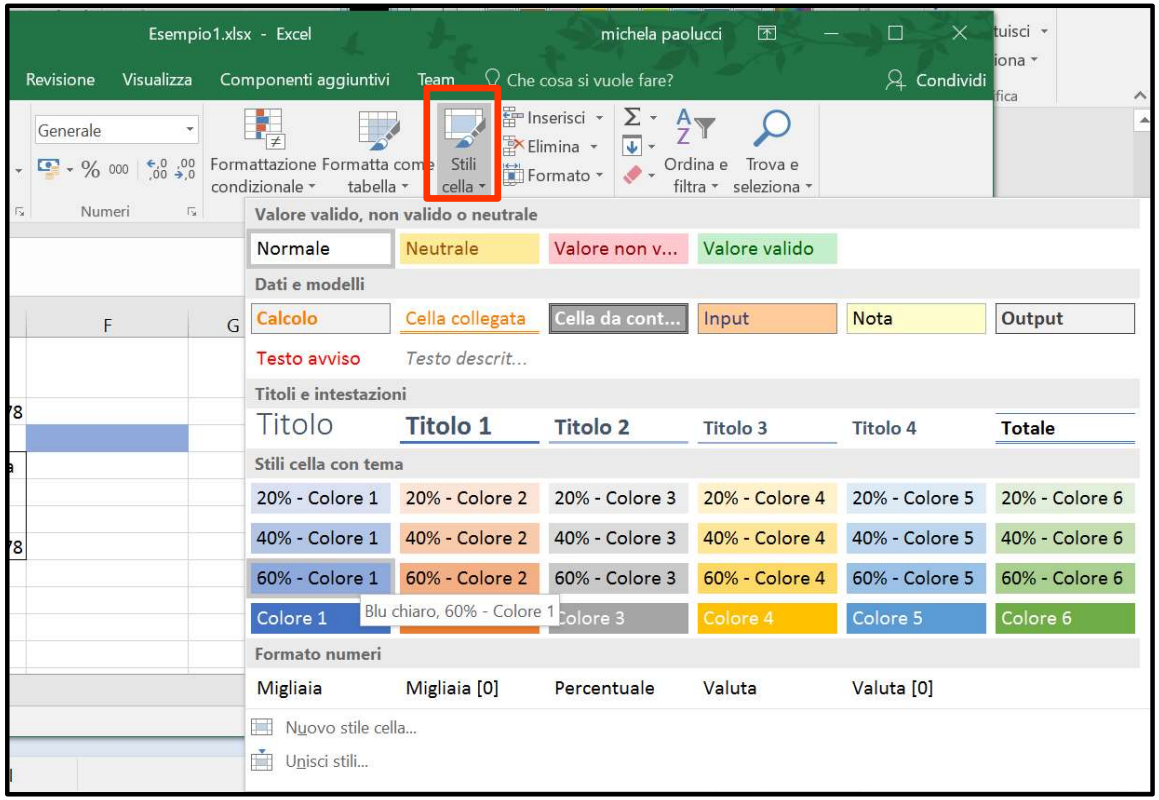
Collegamenti tra fogli (4)

- ='C:\Users\disit\Documents\Paolucci\FondamentiInformatica_2019\Slide\MacroCategorie\Excel\[Esempio2.xlsx]Foglio1'!A3



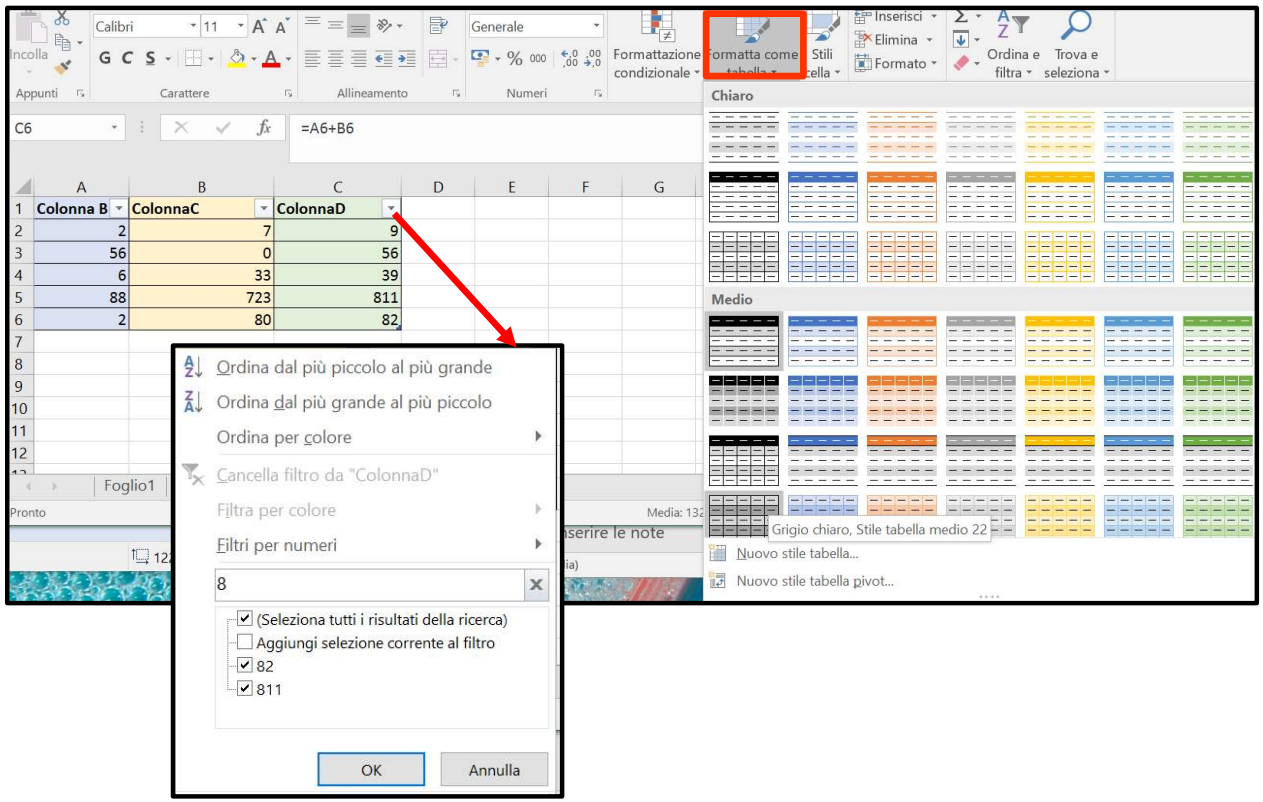
Formattazione (1)

- Insieme di operazioni che riguardano la vista, ovvero la modalità con cui le informazioni vengono visualizzate
- Stile di una cella

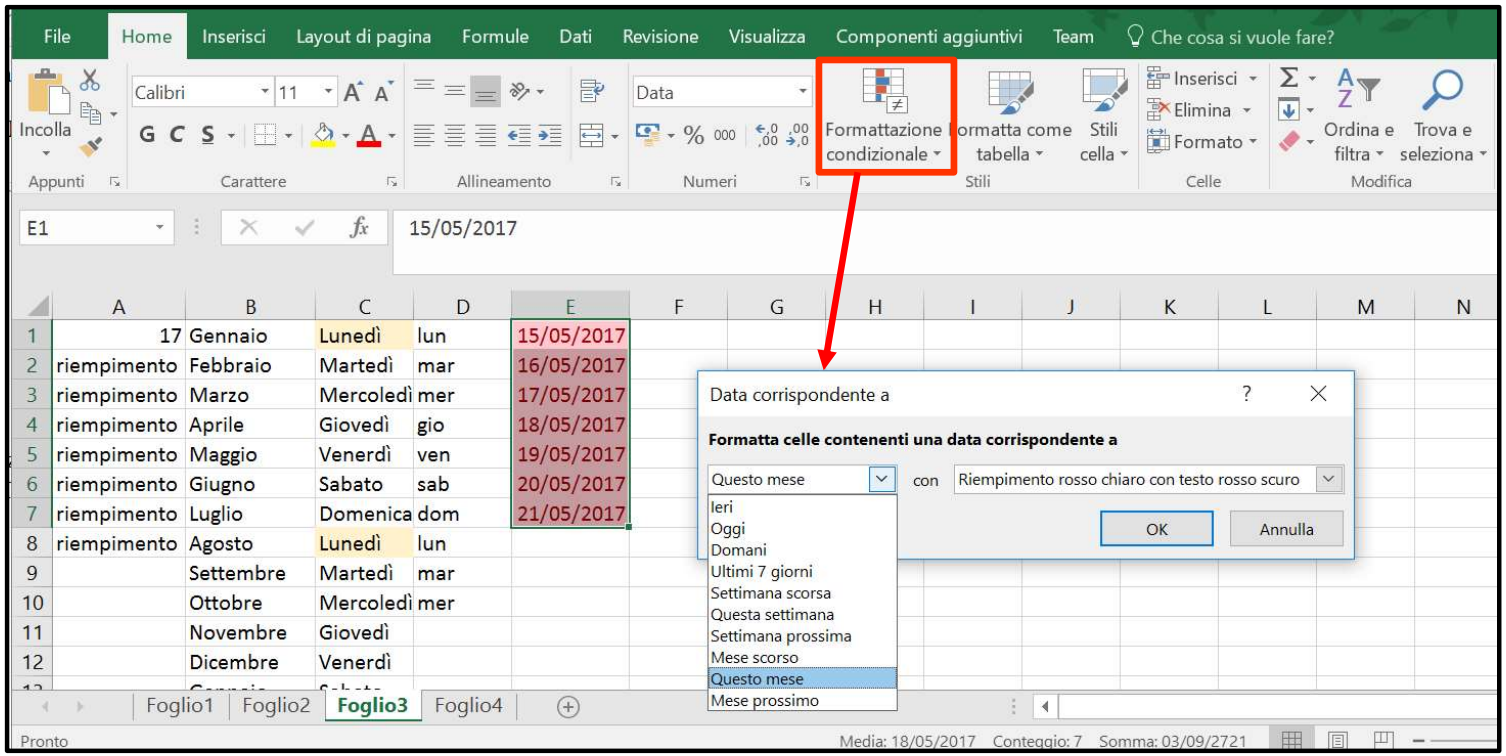


Formattazione (2)

- Insieme di operazioni che riguardano la vista, ovvero la modalità con cui le informazioni vengono visualizzate
- Stile di una tabella



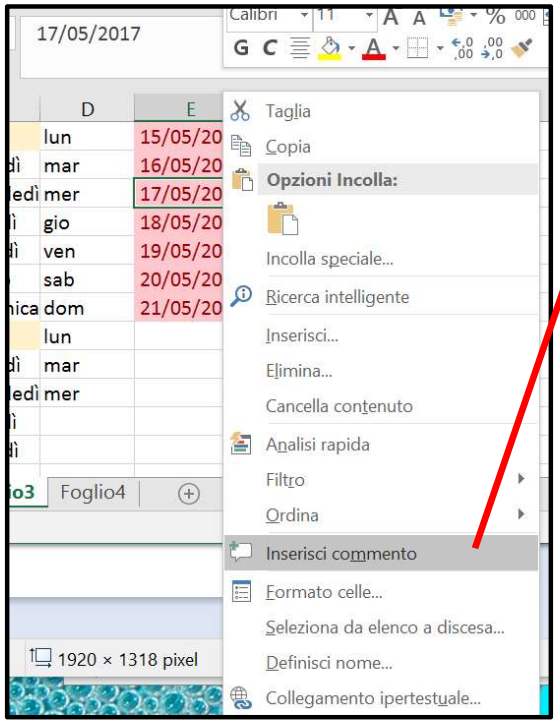
Formattazione (3)

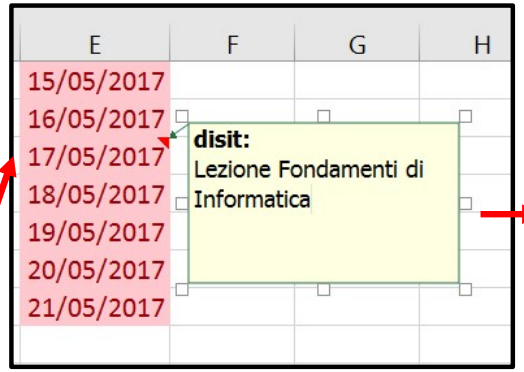


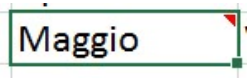
	A	B	C	D	E
1	17	Gennaio	Lunedì	lun	15/05/2017
2	riempimento	Febbraio	Martedì	mar	16/05/2017
3	riempimento	Marzo	Mercoledì	mer	17/05/2017
4	riempimento	Aprile	Giovedì	gio	18/05/2017
5	riempimento	Maggio	Venerdì	ven	19/05/2017
6	riempimento	Giugno	Sabato	sab	20/05/2017
7	riempimento	Luglio	Domenica	dom	21/05/2017
8	riempimento	Agosto	Lunedì	lun	
9		Settembre	Martedì	mar	
10		Ottobre	Mercoledì	mer	
11		Novembre	Giovedì		
12		Dicembre	Venerdì		

Commenti

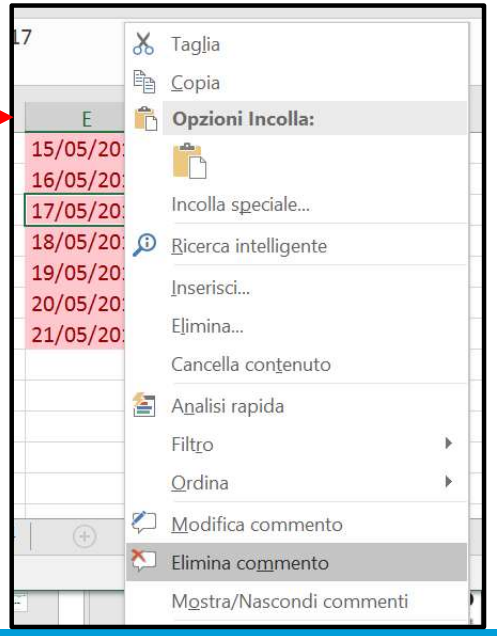
- Selezione cella > Tasto destro > crea commento







- Cella con commento

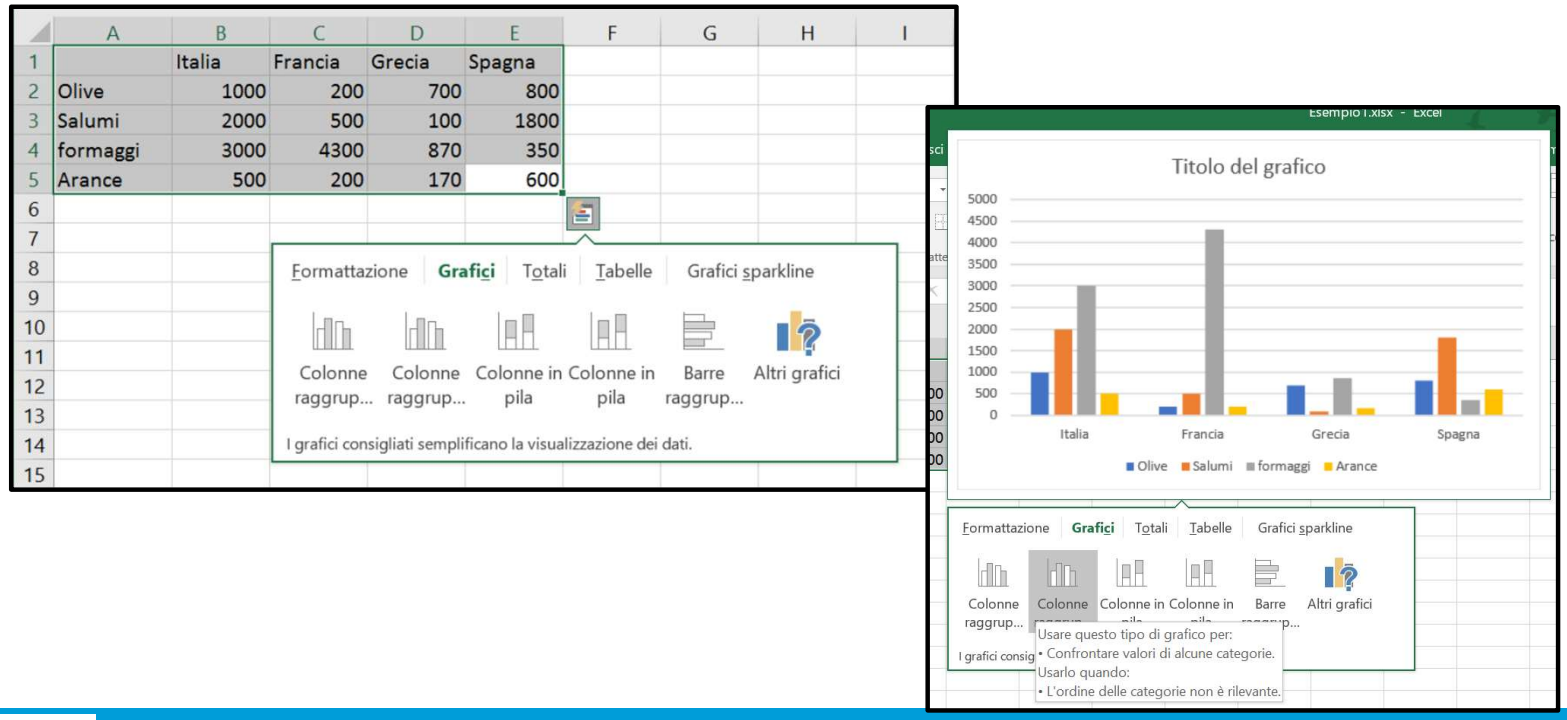


Grafici (1)

- I grafici sono immagini che rappresentano i dati contenuti nelle tabelle
- In generale, risulta molto più semplice e immediato consultare un grafico che una tabella
- Esistono varie tipologie di grafici. La scelta del tipo di grafico da usare dipende:
  - Dai dati che abbiamo a disposizione
  - Dal tipo di informazione che vogliamo 'far emergere' tramite l'uso del grafico
- Alcuni dei grafici più usati sono:
  - Grafici a torta e istogrammi

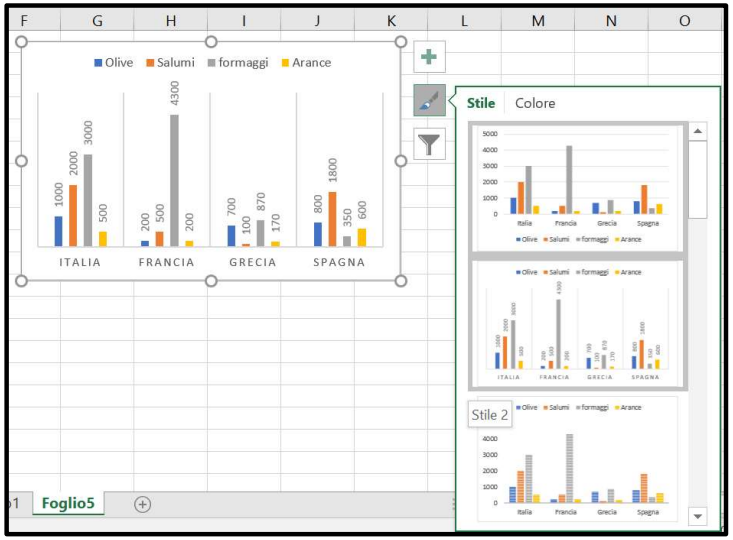
Grafici (2)

- Creazione rapida di un grafico:
  - Selezione tabella > click strumento analisi rapida >grafici



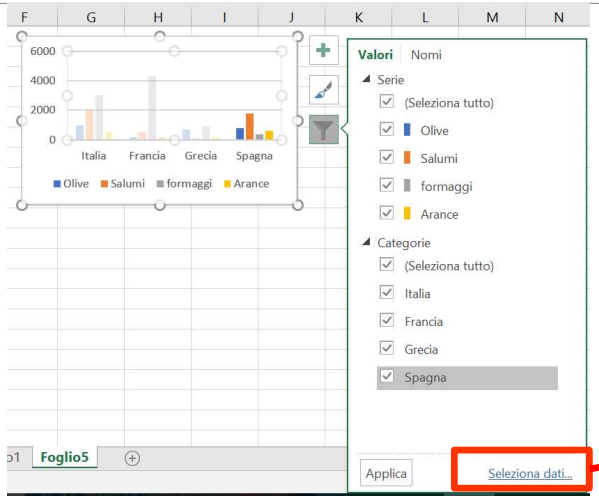
Grafici (3)

- Modifica del grafico:
  - Elementi aggiuntivi: legenda, etichette, titolo, etc.
  - Stile tabella



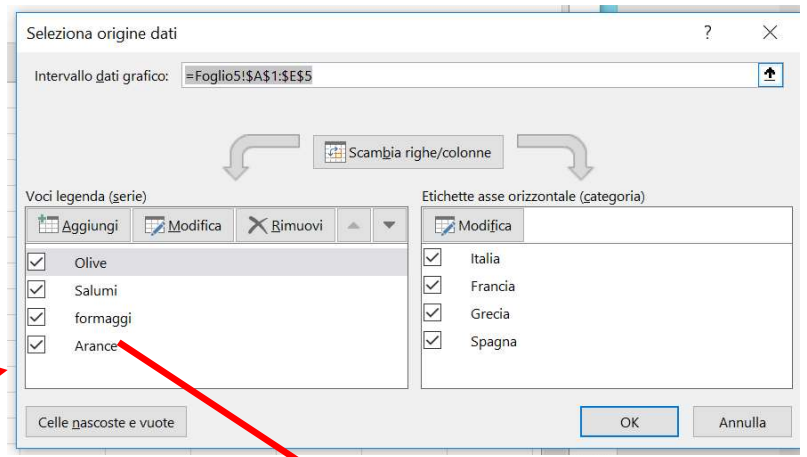
Grafici (4)

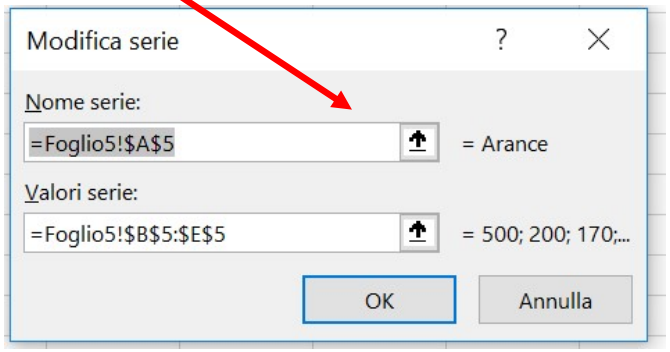
- Filtri e Modifica ai dati da visualizzare nel grafico



	A	B	C	D	E
1		Italia	Francia	Grecia	Spagna
2	Olive	1000	200	700	800
3	Salumi	2000	500	100	1800
4	formaggi	3000	4300	870	350
5	Arance	500	200	170	600

- Tabella: =Foglio5!$A$1:$E$5

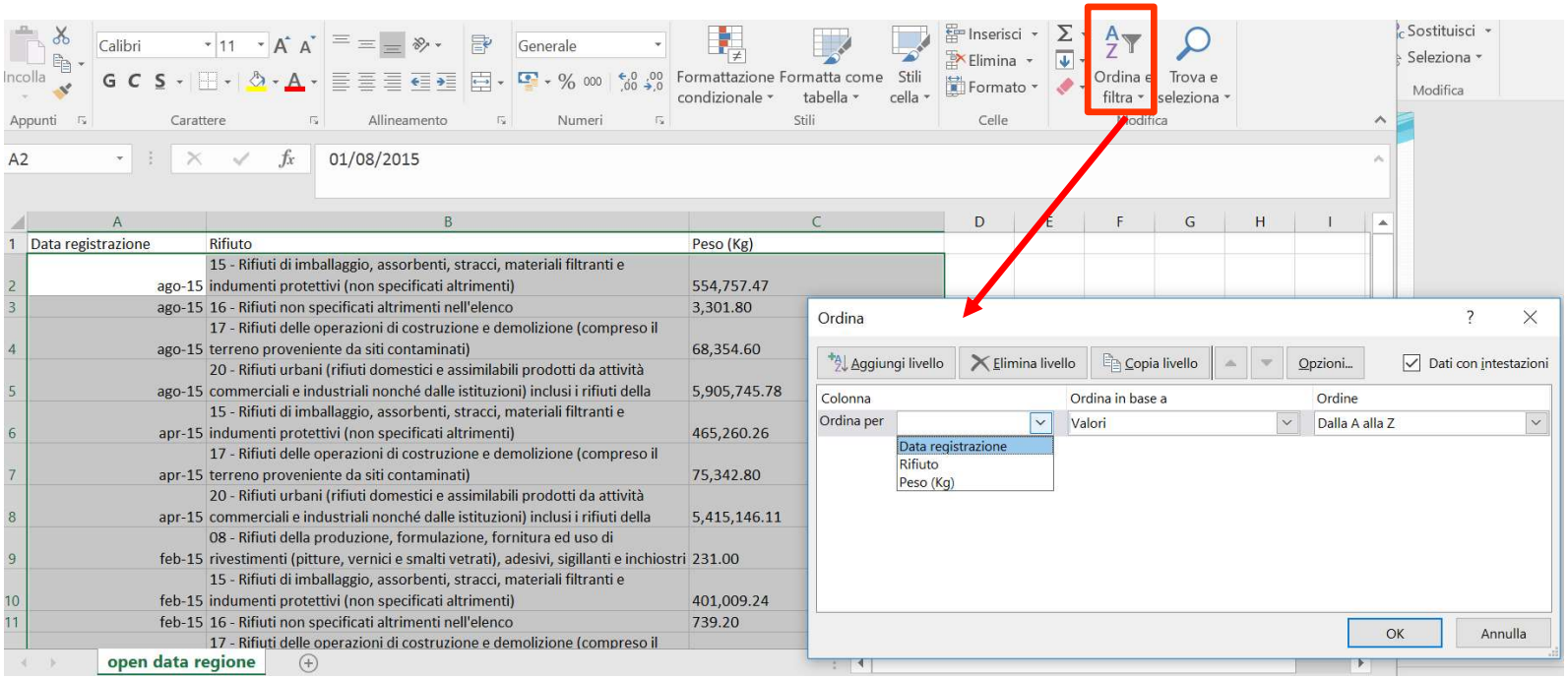




- Nome serie: =Foglio5!$A$2
- Valori serie: =Foglio5!$B$2:$E$2

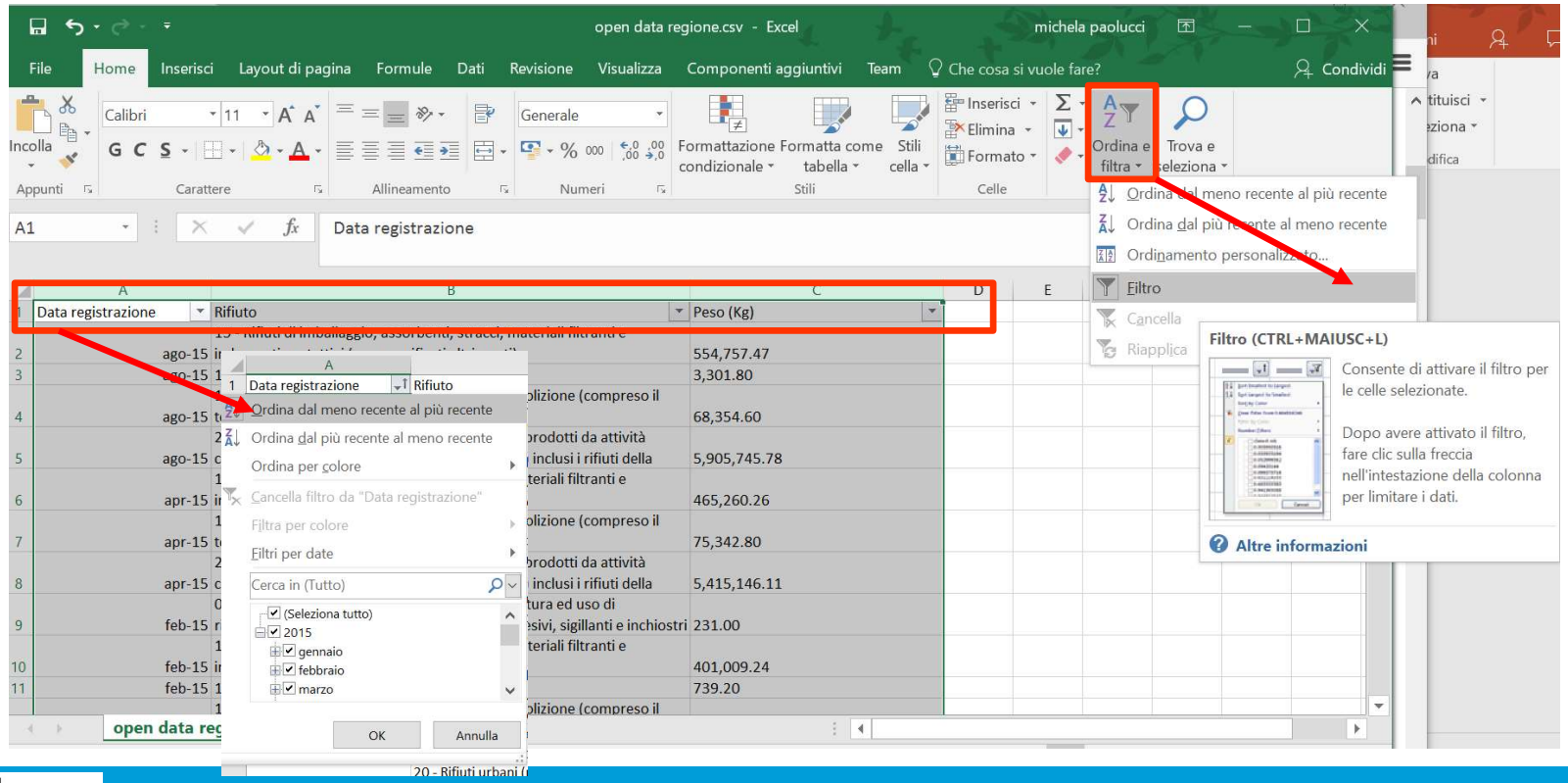
Ordinare i dati

- Menu > Ordina e filtra
  - Ordina per colonna, per riga, etc.



Filtrare i dati

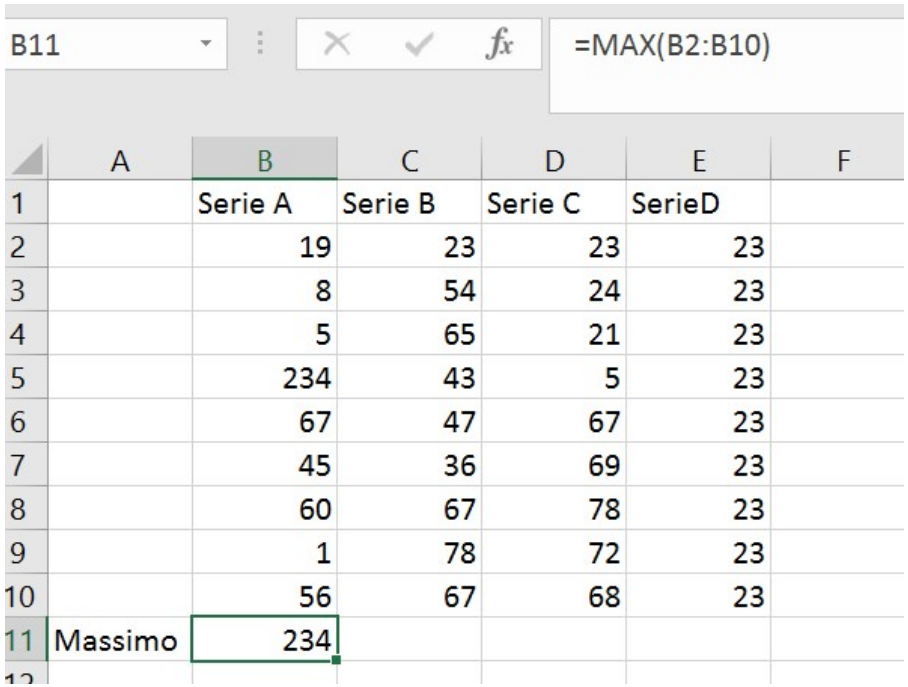
- Menu > Ordina e filtra
  - Filtro per colonna, per riga, etc.



Funzioni – Massimo e Minimo (1)

=MAX(B2:B10)

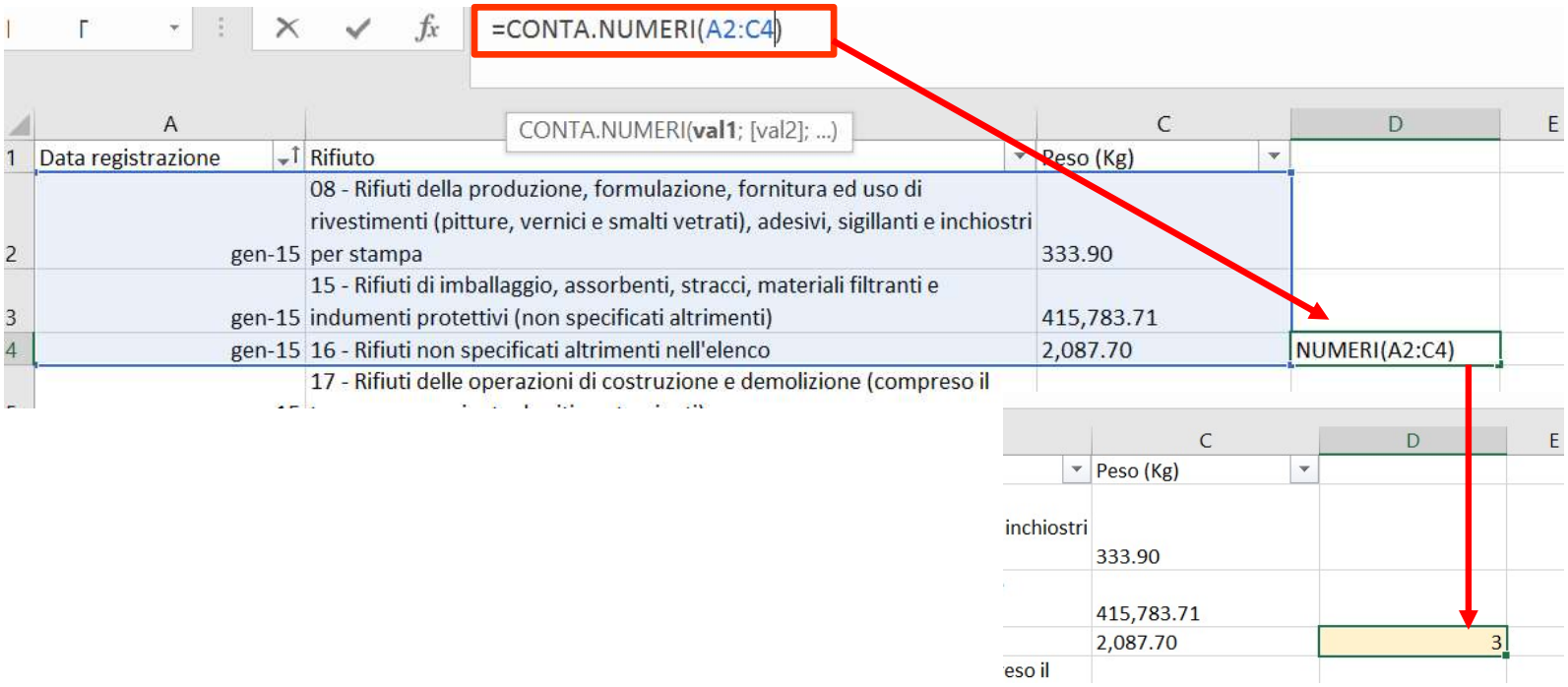
- E’ possibile trascinare la formula sulle altre serie (celle C11 e E11)



	A	B	C	D	E	F
1		Serie A	Serie B	Serie C	SerieD	
2		19	23	23	23	
3		8	54	24	23	
4		5	65	21	23	
5		234	43	5	23	
6		67	47	67	23	
7		45	36	69	23	
8		60	67	78	23	
9		1	78	72	23	
10		56	67	68	23	
11	Massimo	234				

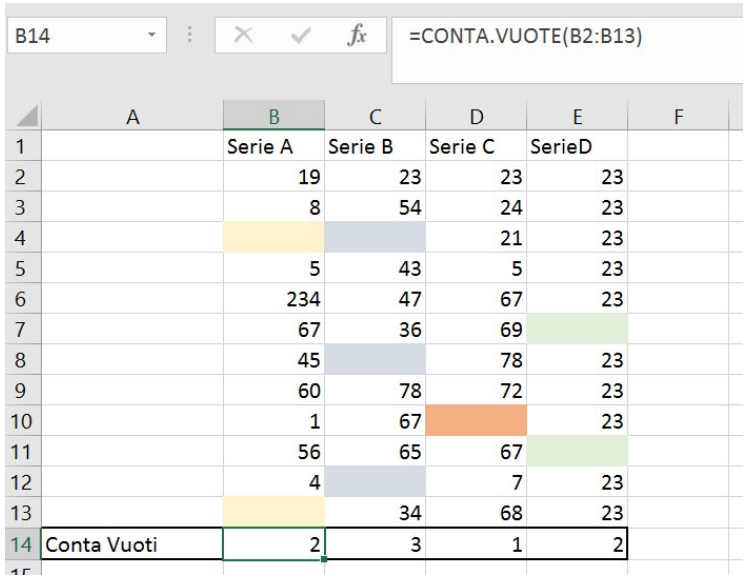
Funzioni di conteggio (1)

- La funzione CONTA.NUMERI conteggia il numero di celle di un dato intervallo che contengono numeri (incluse le date)



Funzioni di conteggio (2)

- La funzione CONTA.VUOTE conteggia il numero di celle vuote di un dato intervallo
- E' possibile trascinare la formula sulle altre serie (celle C14 e E14)

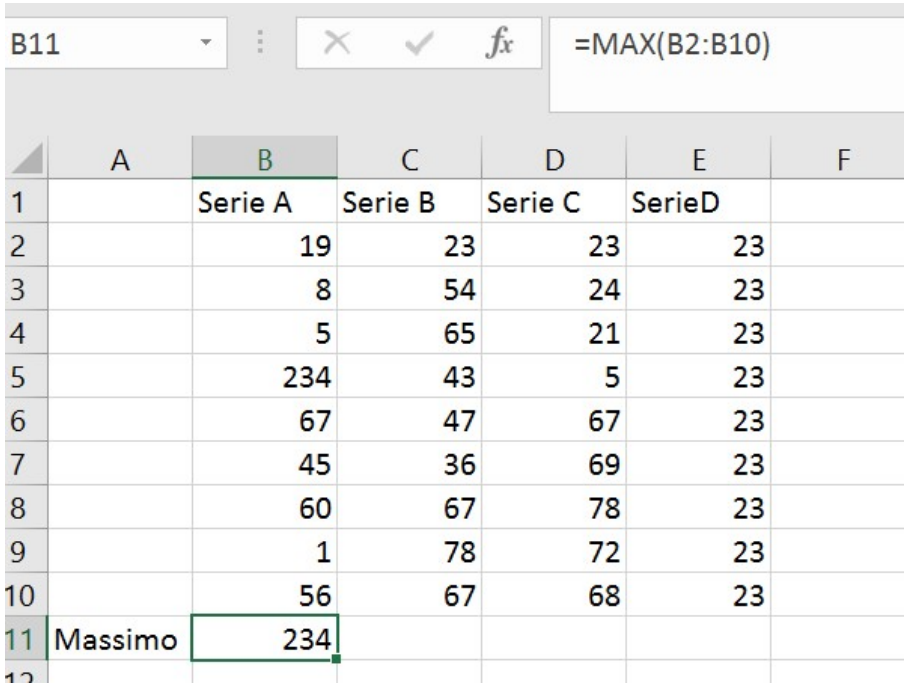


	A	B	C	D	E	F
1		Serie A	Serie B	Serie C	SerieD	
2		19	23	23	23	
3		8	54	24	23	
4				21	23	
5		5	43	5	23	
6		234	47	67	23	
7		67	36	69		
8		45		78	23	
9		60	78	72	23	
10		1	67		23	
11		56	65	67		
12		4		7	23	
13			34	68	23	
14	Conta Vuoti	2	3	1	2	

Funzioni – Massimo e Minimo (1)

=MAX(B2:B10)

- E' possibile trascinare la formula sulle altre serie (celle C11 e E11)

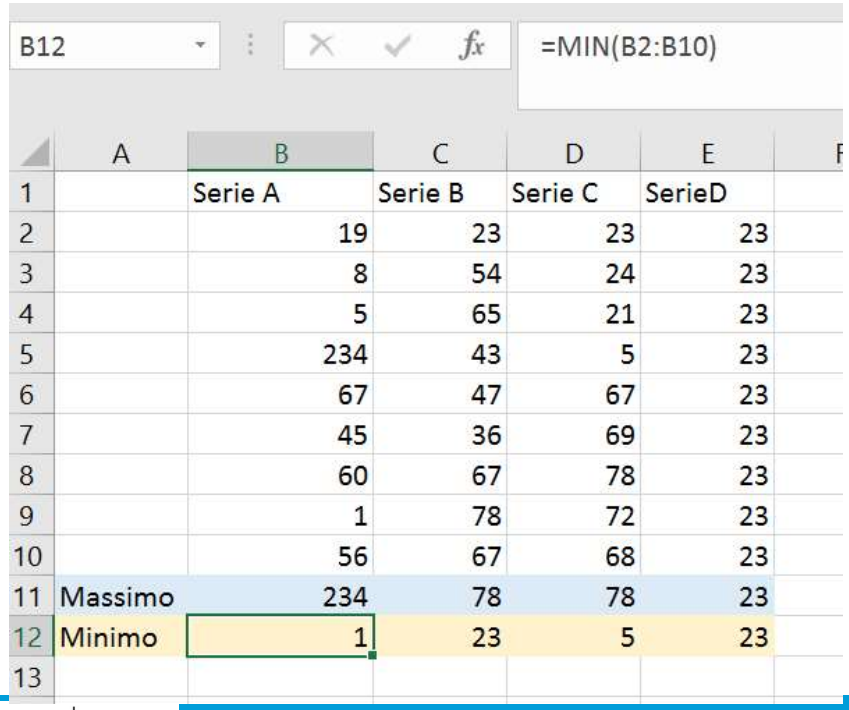


	A	B	C	D	E	F
1		Serie A	Serie B	Serie C	SerieD	
2		19	23	23	23	
3		8	54	24	23	
4		5	65	21	23	
5		234	43	5	23	
6		67	47	67	23	
7		45	36	69	23	
8		60	67	78	23	
9		1	78	72	23	
10		56	67	68	23	
11	Massimo	234				

Funzioni – Massimo e Minimo (2)

=MIN(B2:B10)

- E' possibile trascinare la formula sulle altre serie (celle C12 e E12)



	A	B	C	D	E
1		Serie A	Serie B	Serie C	SerieD
2		19	23	23	23
3		8	54	24	23
4		5	65	21	23
5		234	43	5	23
6		67	47	67	23
7		45	36	69	23
8		60	67	78	23
9		1	78	72	23
10		56	67	68	23
11	Massimo	234	78	78	23
12	Minimo	1	23	5	23
13					

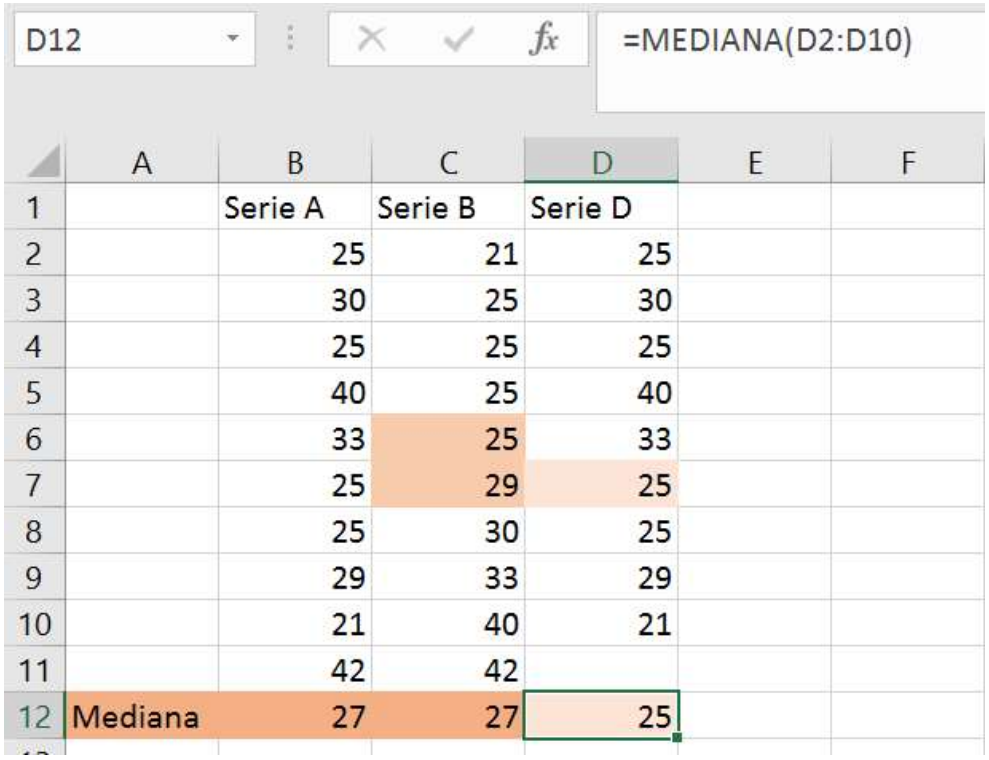
Funzioni – Mediana, Moda (1)

- La MEDIANA è quel valore che divide in due parti uguali la distribuzione di una serie di dati

- La MODA è il valore che ha la frequenza più elevata in una distribuzione

Funzioni – Mediana, Moda (2)

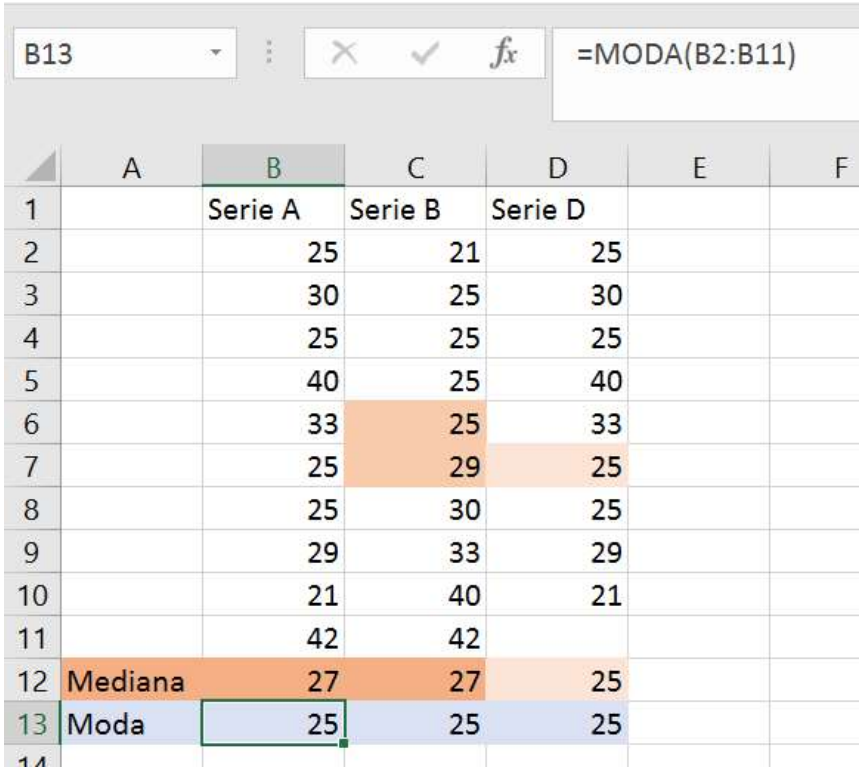
- La MEDIANA è quel valore che divide in due parti uguali la distribuzione di una serie di dati
- Se ci sono due numeri in posizione centrale, restituisce la media dei due numeri (da C2 a C11 sono 10 valori, numero pari)



	A	B	C	D	E	F
1		Serie A	Serie B	Serie D		
2		25	21	25		
3		30	25	30		
4		25	25	25		
5		40	25	40		
6		33	25	33		
7		25	29	25		
8		25	30	25		
9		29	33	29		
10		21	40	21		
11		42	42			
12	Mediana	27	27	25		

Funzioni – Mediana, Moda (3)

- La MODA è il valore che ha la frequenza più elevata in una distribuzione
- E' il valore che compare più frequentemente in una serie numerica



	A	B	C	D	E	F
1		Serie A	Serie B	Serie D		
2		25	21	25		
3		30	25	30		
4		25	25	25		
5		40	25	40		
6		33	25	33		
7		25	29	25		
8		25	30	25		
9		29	33	29		
10		21	40	21		
11		42	42			
12	Mediana	27	27	25		
13	Moda	25	25	25		

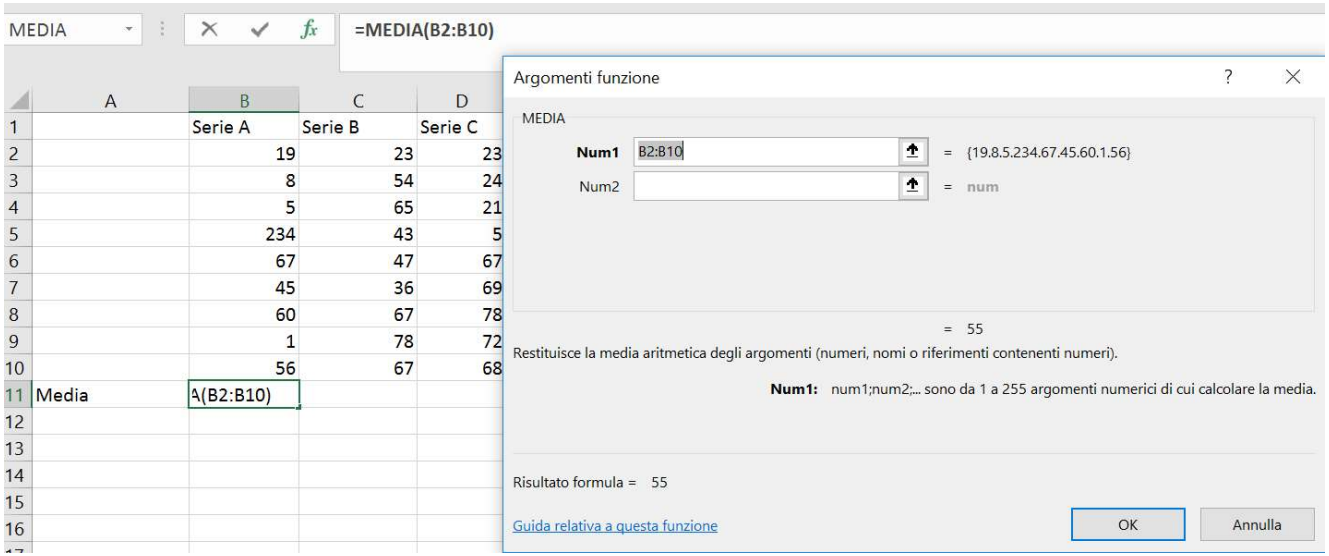
Funzioni – Varianza e deviazione standard (1)

- Varianza e deviazione standard (o scarto quadratico medio) sono strumenti per analizzare gli indici di dispersione dei dati. Servono per valutare quanto una serie di valori sia più o meno uniforme rispetto alla sua media
- Le funzioni usate sono:
  - MEDIA
  - VAR.C
  - DEV.ST.C

Funzioni – Varianza e deviazione standard (2)

=MEDIA(B2:B10)

- E’ possibile trascinare la formula sulle altre serie (celle C11 e D11)

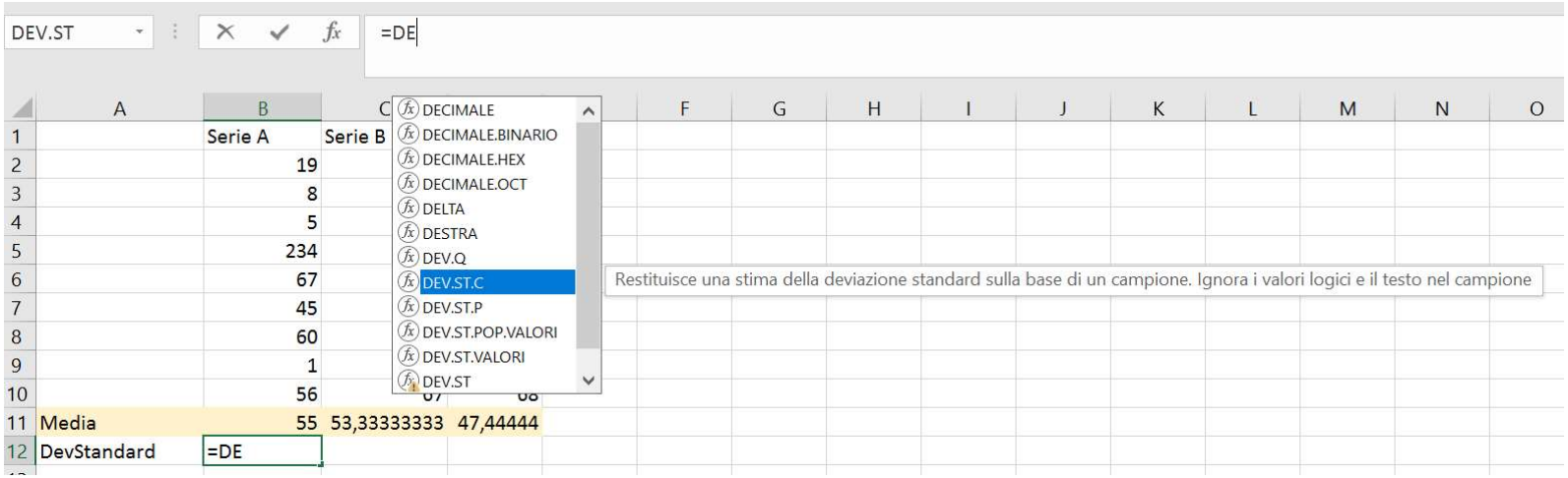


	A	B	C	D
1		Serie A	Serie B	Serie C
2		19	23	23
3		8	54	24
4		5	65	21
5		234	43	5
6		67	47	67
7		45	36	69
8		60	67	78
9		1	78	72
10		56	67	68
11	Media	A(B2:B10)		
12				
13				
14				
15				
16				

Funzioni – Varianza e deviazione standard (3)

=DEV.ST.C(B2:B10)

- E’ possibile trascinare la formula sulle altre serie (celle C12 e D12)



NOTE:

- DEV.ST.C si applica per il calcolo su un campione
- DEV.ST.P si applica se si considera un'intera popolazione.

Funzioni matematiche e trigonometriche

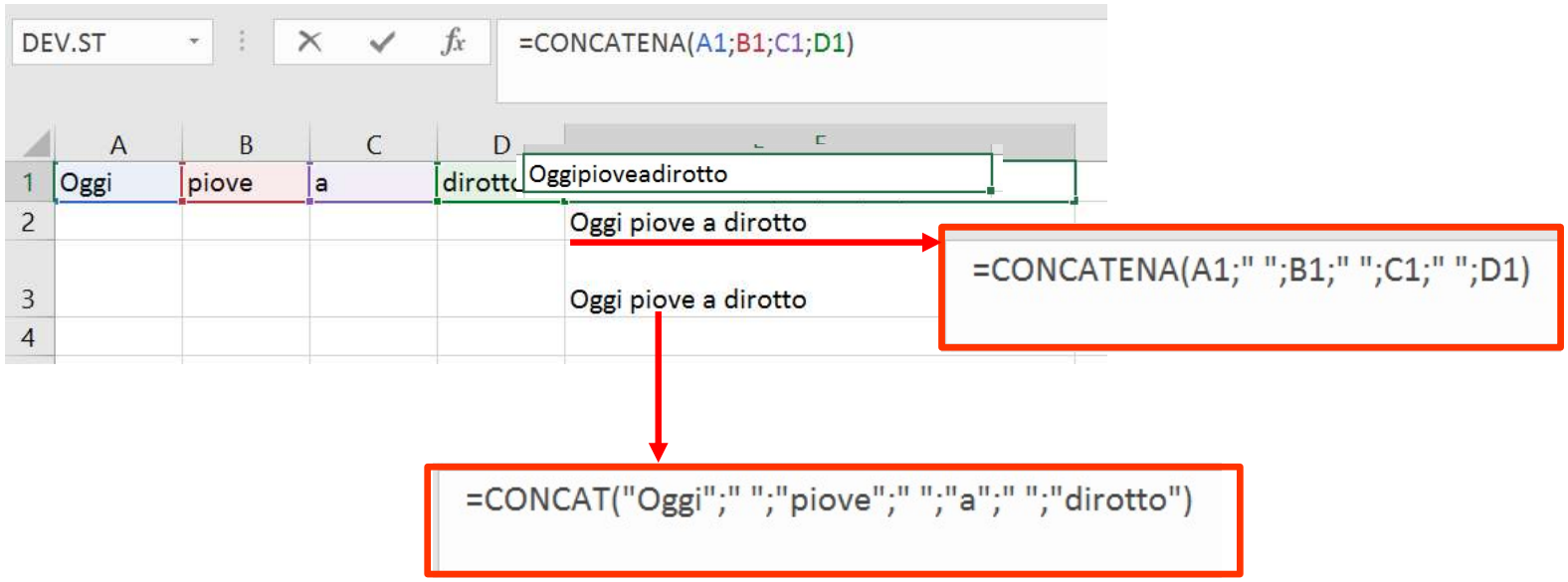
- SIN, COS, COT, TAN, …
- ABS, SOMMA, FATTORIALE, EXP, MCD, MCM, DISPARI, PARI, PRODOTTO, POTENZA, …
- ARROTONDA.ECCESSO, ARROTONDA.DIFETTO, …
- INT,
- MATR.DETERMINANTE, MATR.INVERSA, MATR.PRODOTO, …

Funzioni statistiche

- MEDIA, CONTA.VUOTE, CORRELAZIONE, CONTA.VALORI, DEV.ST.C, DEV.ST.P, VAR.C, VAR.P., ...
- MAX, MIN, MEDIANA, NORMALIZZATO, …

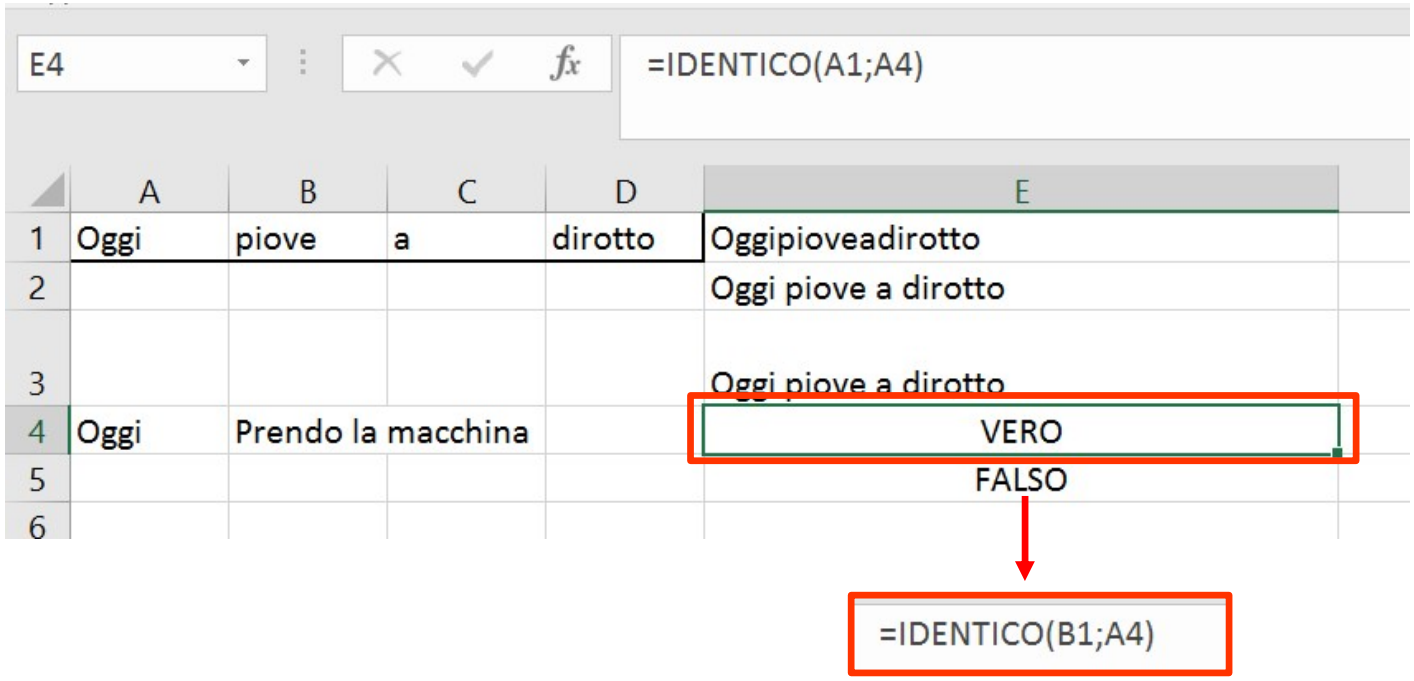
Lavorare con le stringhe (1)

- CONCATENA(testo1;testo2;…)



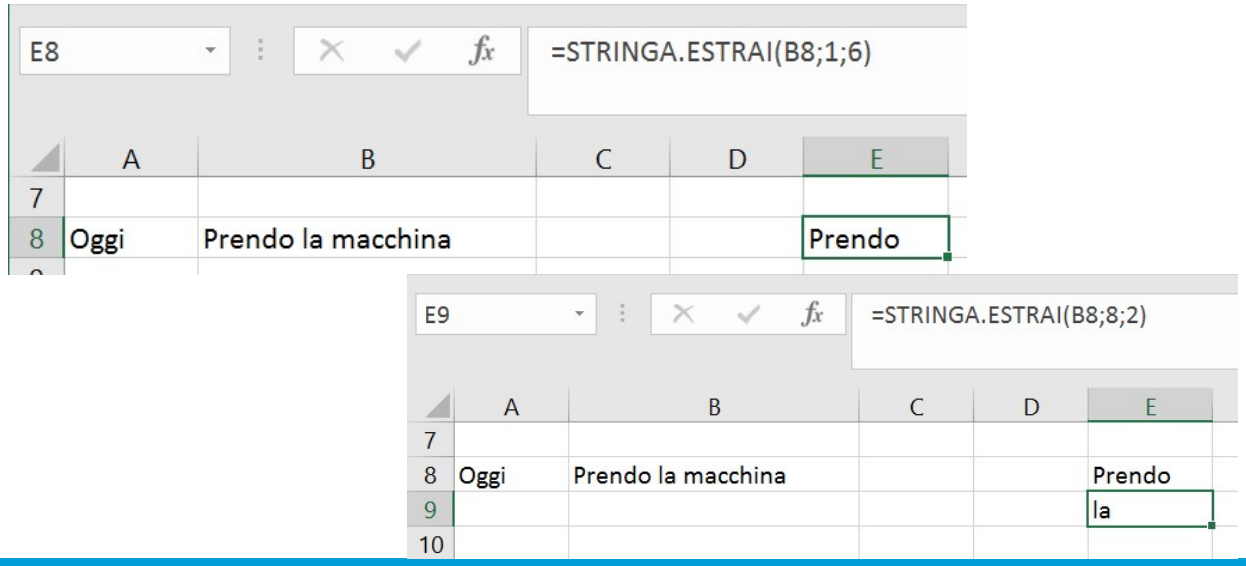
Lavorare con le stringhe (2)

- IDENTICO(testo1;testo2)



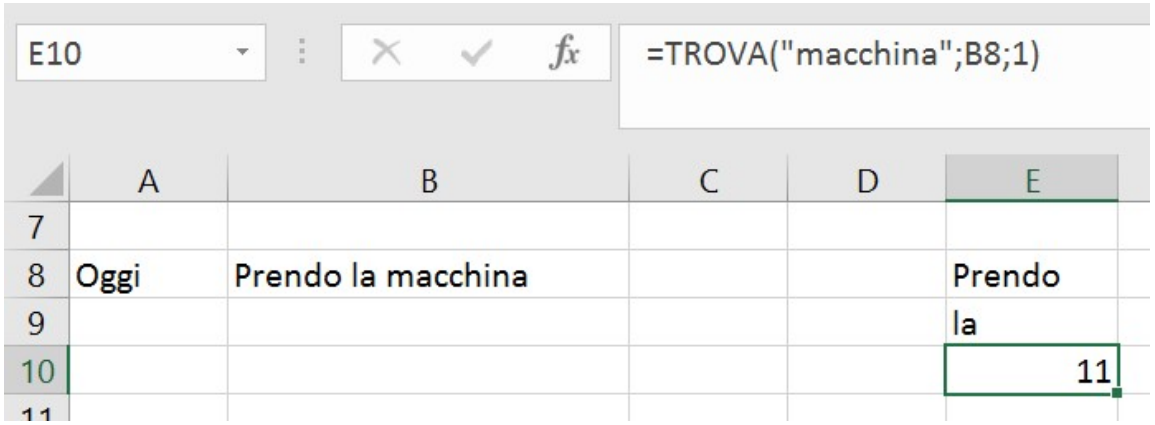
Lavorare con le stringhe (3)

- STRINGA.ESTRAI(testo; inizio; num_caratteri)
  - inizio = carattere da cui partire
  - num_caratteri = numero di caratteri da estrarre dalla stringa
- Esempi:
  - STRINGA.ESTRAI(‘‘Prendo la macchina’’;1;6)
  - STRINGA.ESTRAI(B8;1;6)
  - STRINGA.ESTRAI(B8;8;2)



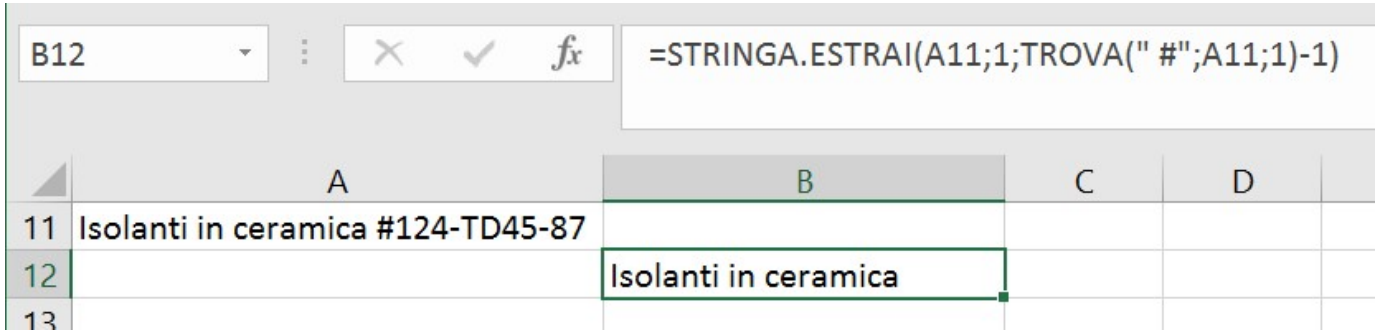
Lavorare con le stringhe (4)

- TROVA(testo; stringa; inizio)
  - testo = testo da cercare
  - Stringa = stringa in cui cercare il testo
  - Inizio = numero del carattere della stringa da cui iniziare a cercare
- Consente di individuare una stringa di testo all'interno di una seconda stringa di testo e restituisce il numero corrispondente alla posizione iniziale della prima stringa di testo dal primo carattere della seconda stringa di testo



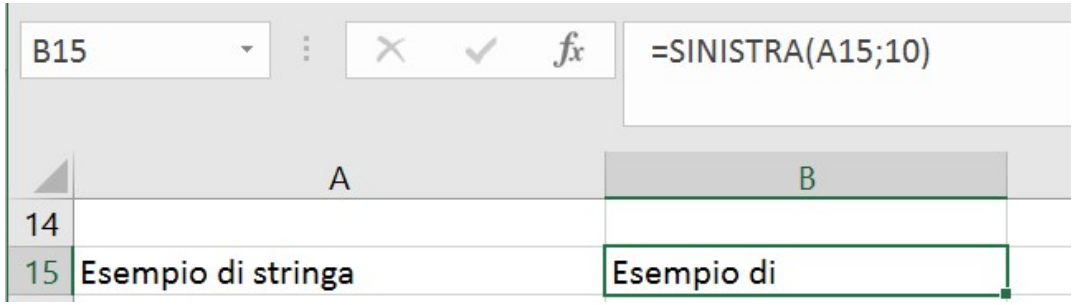
Lavorare con le stringhe (5)

- Composizione di funzioni
- STRINGA.ESTRAI(testo; inizio; num_caratteri) e TROVA(testo; stringa; inizio)
- ESEMPIO:
  - STRINGA.ESTRAI(A11;1;TROVA(" #";A11;1)-1)
    - Estrae testo dalla posizione 1 alla posizione di "#" nella cella A2 (Isolanti in ceramica)



Lavorare con le stringhe (6)

- SINISTRA(testo; [num_caratt])
  - testo = testo di partenza
  - num_caratteri = numero di caratteri da estrarre
- Restituisce il primo carattere o i primi caratteri di una stringa di testo in base al numero di caratteri specificato a partire da sinistra

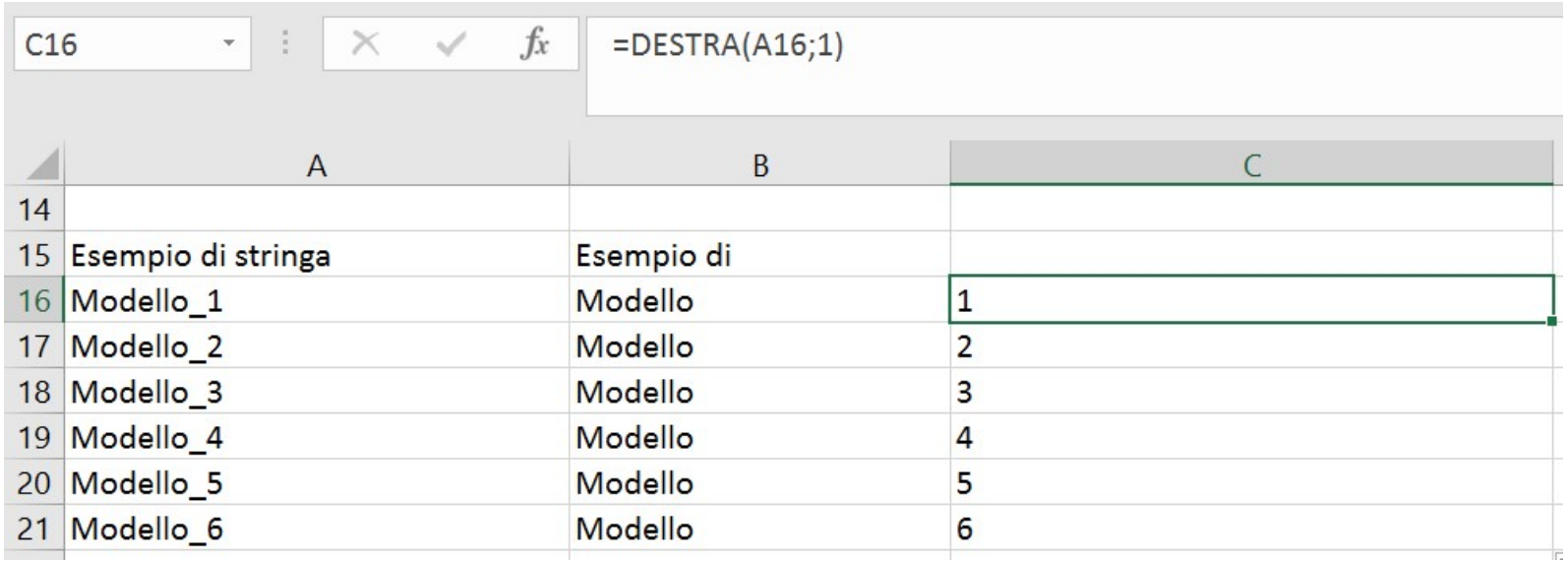


Lavorare con le stringhe (7)

- DESTRA(testo; [num_caratt])
  - testo = testo di partenza
  - num_caratteri = numero di caratteri da estrarre
- Restituisce il primo carattere o i primi caratteri di una stringa di testo in base al numero di caratteri specificato a partire da destra

Lavorare con le stringhe (8)

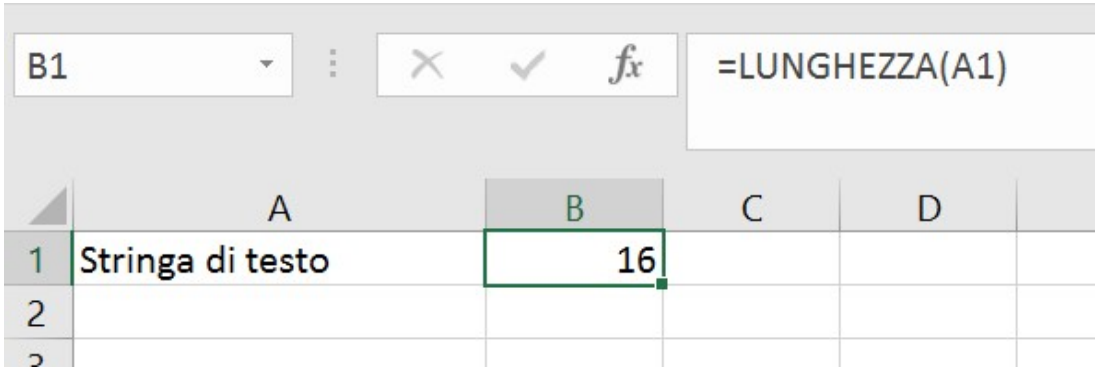
- Copiare formule…



	A	B	C
14			
15	Esempio di stringa	Esempio di	
16	Modello_1	Modello	1
17	Modello_2	Modello	2
18	Modello_3	Modello	3
19	Modello_4	Modello	4
20	Modello_5	Modello	5
21	Modello_6	Modello	6

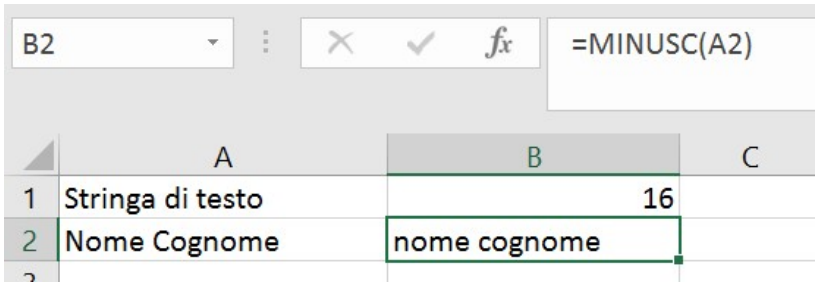
Lavorare con le stringhe (9)

- LUNGHEZZA(testo)
  - Restituisce il numero di caratteri di una stringa di testo (spazi compresi)

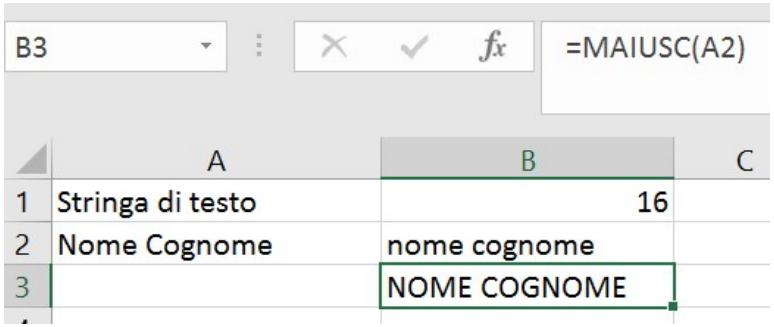


Lavorare con le stringhe (10)

- MINUSC(testo)
- Converte in minuscolo tutte le lettere maiuscole contenute in una stringa di testo

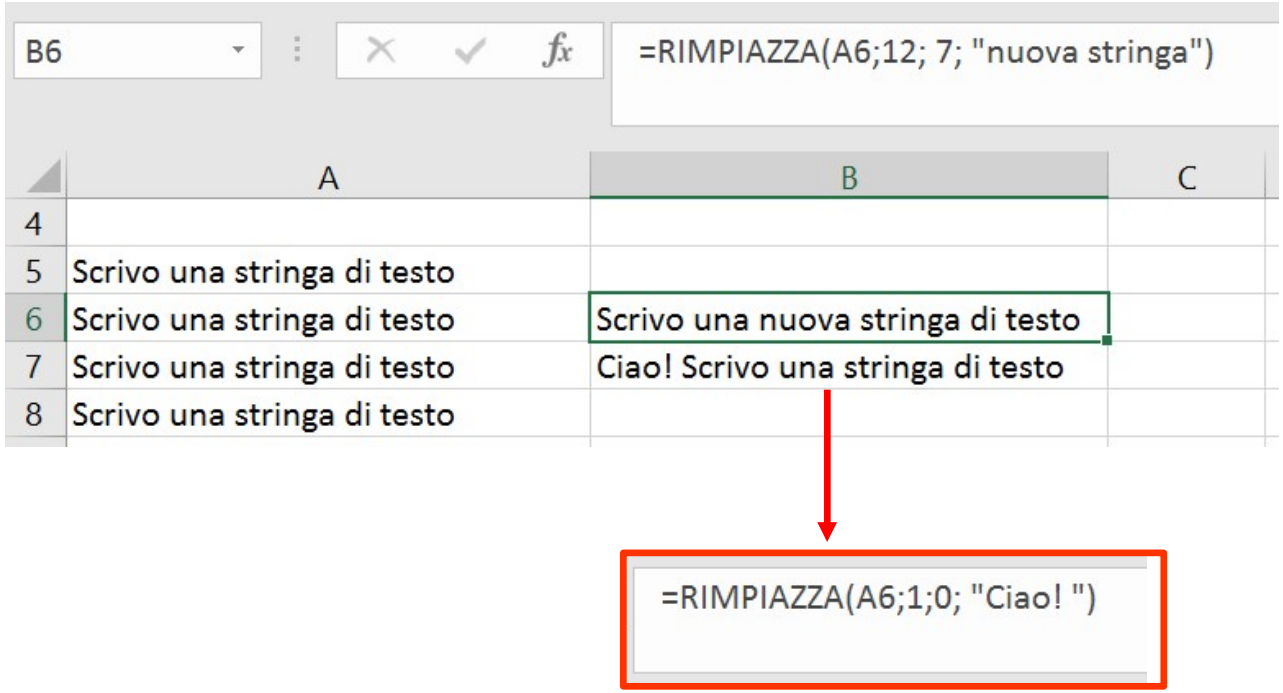


- MAIUSC(testo)
- Converte in minuscolo tutte le lettere maiuscole contenute in una stringa di testo)



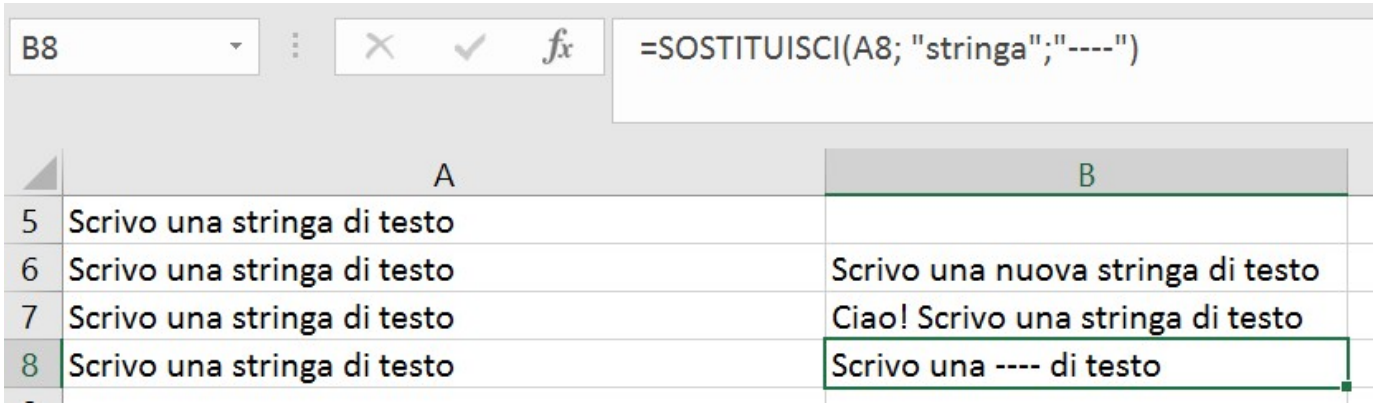
Lavorare con le stringhe (11)

- RIMPIAZZA(testo_prec; inizio; num_caratt; nuovo_testo)
- Sostituisce parte di una stringa di testo con una stringa di testo diversa, in base al numero di caratteri specificati.



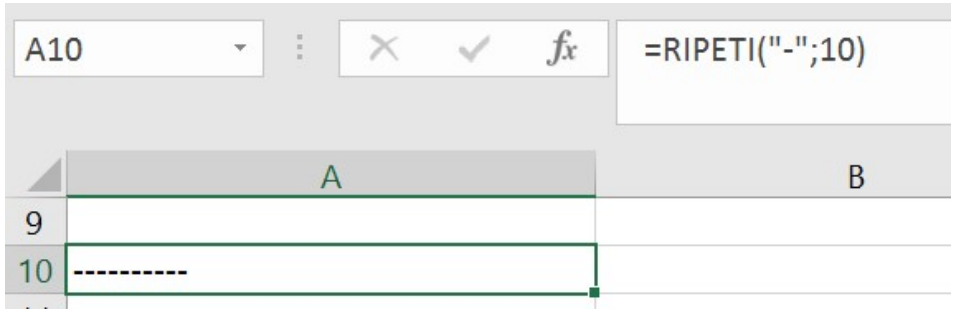
Lavorare con le stringhe (12)

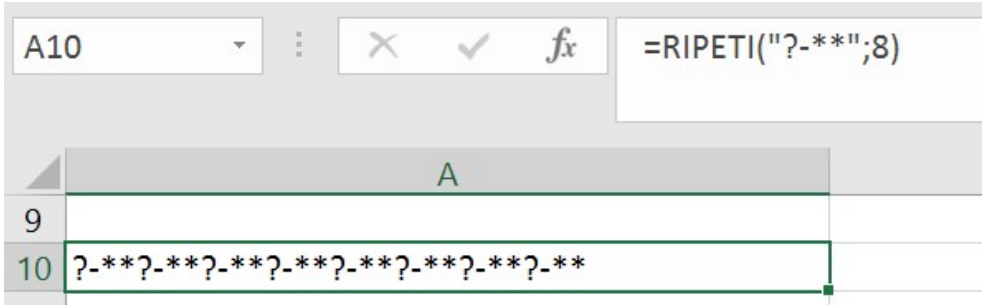
- SOSTITUISCI(testo; testo_prec; nuovo_testo; [occorrenza])
- Sostituisce nuovo_testo a testo_prec in una stringa di testo



Lavorare con le stringhe (13)

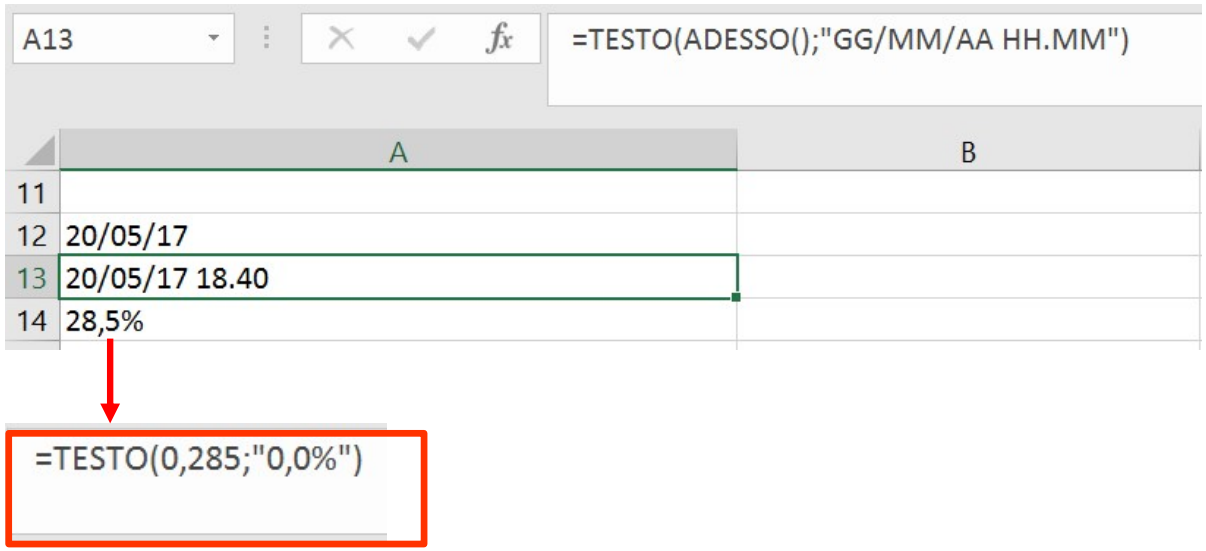
- RIPETI(testo; volte)
- Ripete un testo per il numero di volte specificato. Utilizzare la funzione RIPETI per riempire una cella con una stringa di testo ripetuta più volte.





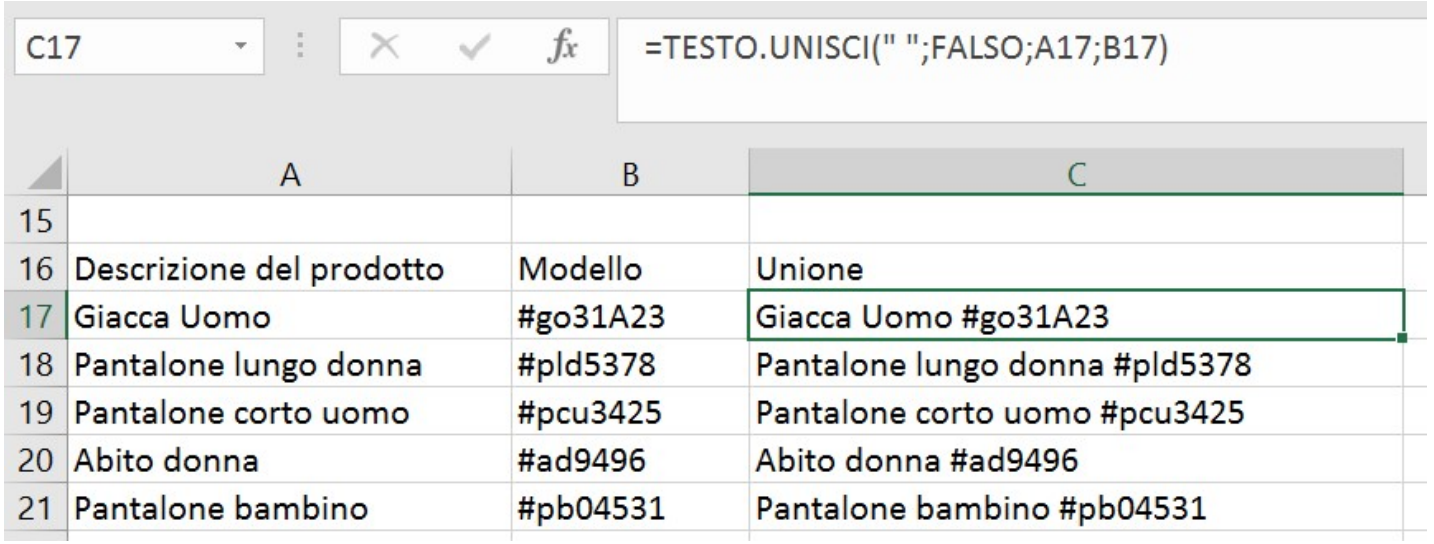
Lavorare con le stringhe (14)

- TESTO(Valore da formattare;"Codice formato da applicare")



Lavorare con le stringhe (15)

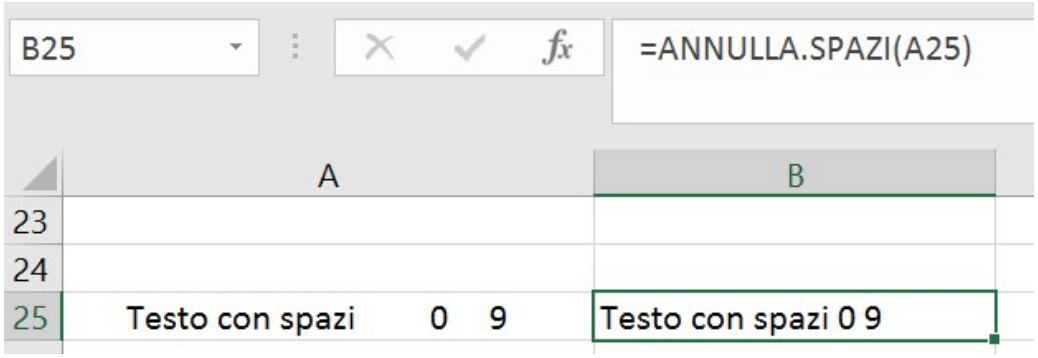
- TESTO.UNISCI(delimitatore;ignora_vuote; testo1;[testo2]; …)
- La funzione TESTO.UNISCI combina il testo di più intervalli e/o stringhe e include un delimitatore specificato dall'utente tra ogni valore di testo da unire. Se il delimitatore è una stringa di testo vuota, la funzione concatena correttamente gli intervalli



	A	B	C
15			
16	Descrizione del prodotto	Modello	Unione
17	Giacca Uomo	#go31A23	Giacca Uomo #go31A23
18	Pantalone lungo donna	#pld5378	Pantalone lungo donna #pld5378
19	Pantalone corto uomo	#pcu3425	Pantalone corto uomo #pcu3425
20	Abito donna	#ad9496	Abito donna #ad9496
21	Pantalone bambino	#pb04531	Pantalone bambino #pb04531

Lavorare con le stringhe (16)

- ANNULLA.SPAZI(testo)
- Rimuove tutti gli spazi dal testo ad eccezione dei singoli spazi tra le parole. Utilizzare la funzione ANNULLA.SPAZI sul testo creato con altre applicazioni che può presentare una distribuzione irregolare degli spazi



Funzioni Logiche (1)

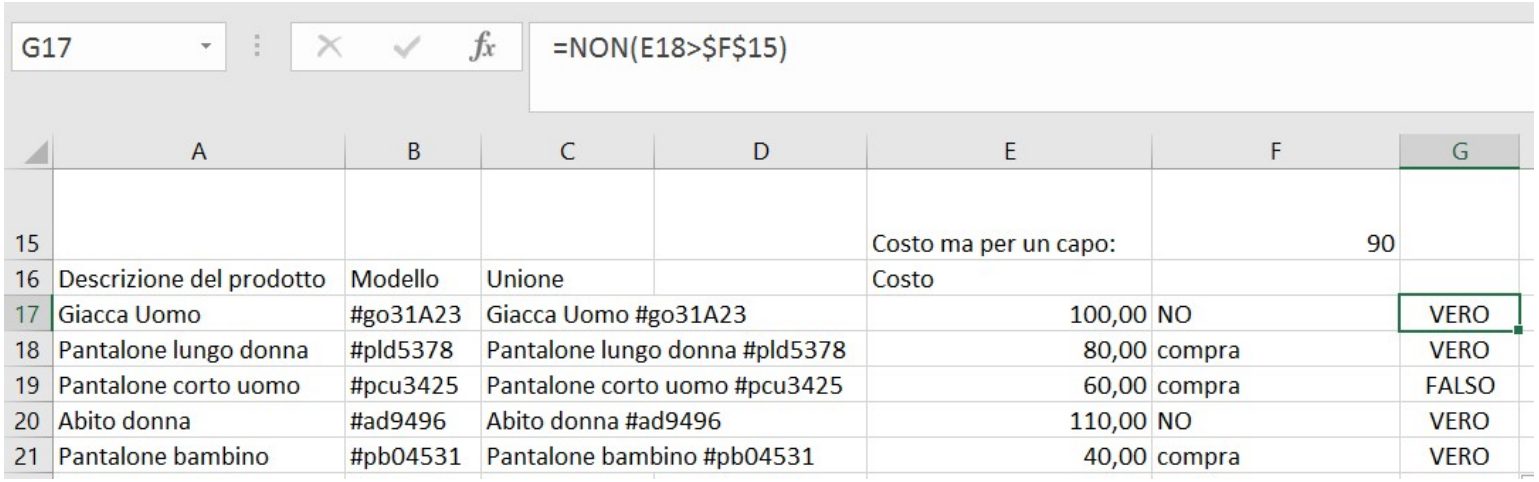
- TRUE
- FALSE
- SE (IF)
  - SE(qualcosa è Vero; fai qualcosa; altrimenti fai qualcos'altro)

F17 | =SE(E17<=$F$15;"compra";"NO")

	A	B	C	D	E	F
15					Costo ma per un capo:	90
16	Descrizione del prodotto	Modello	Unione		Costo	
17	Giacca Uomo	#go31A23	Giacca Uomo #go31A23		100,00	NO
18	Pantalone lungo donna	#pld5378	Pantalone lungo donna #pld5378		80,00	compra
19	Pantalone corto uomo	#pcu3425	Pantalone corto uomo #pcu3425		60,00	compra
20	Abito donna	#ad9496	Abito donna #ad9496		110,00	NO
21	Pantalone bambino	#pb04531	Pantalone bambino #pb04531		40,00	compra

Funzioni Logiche (2)

- NON(Argomento)
- Inverte la logica dell'argomento (Restituisce FALSO per un argomento VERO e VERO per un argomento FALSO)

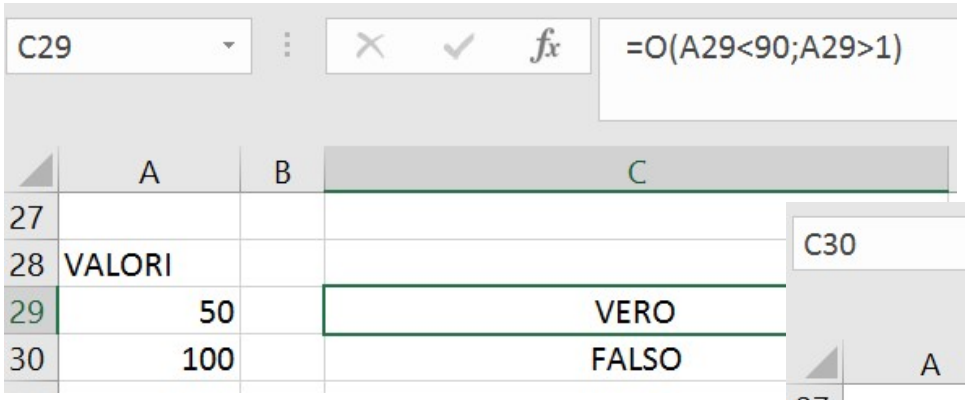


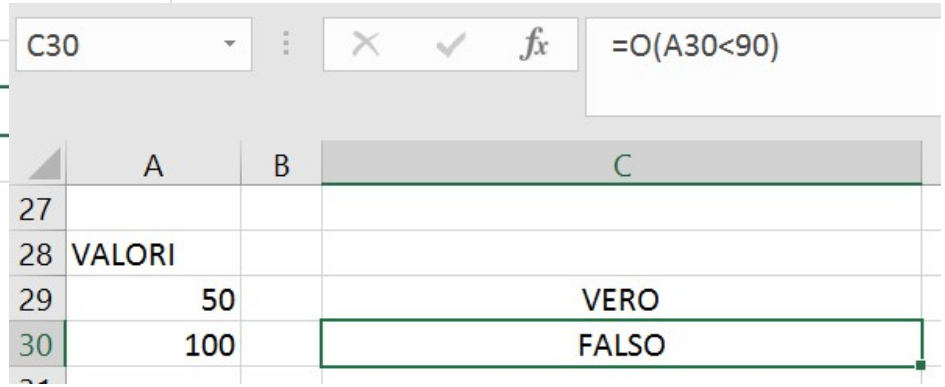
G17 | =NON(E18>$F$15)

	A	B	C	D	E	F	G
15					Costo ma per un capo:	90	
16	Descrizione del prodotto	Modello	Unione		Costo		
17	Giacca Uomo	#go31A23	Giacca Uomo #go31A23		100,00	NO	VERO
18	Pantalone lungo donna	#pld5378	Pantalone lungo donna #pld5378		80,00	compra	VERO
19	Pantalone corto uomo	#pcu3425	Pantalone corto uomo #pcu3425		60,00	compra	FALSO
20	Abito donna	#ad9496	Abito donna #ad9496		110,00	NO	VERO
21	Pantalone bambino	#pb04531	Pantalone bambino #pb04531		40,00	compra	VERO

Funzioni Logiche (3)

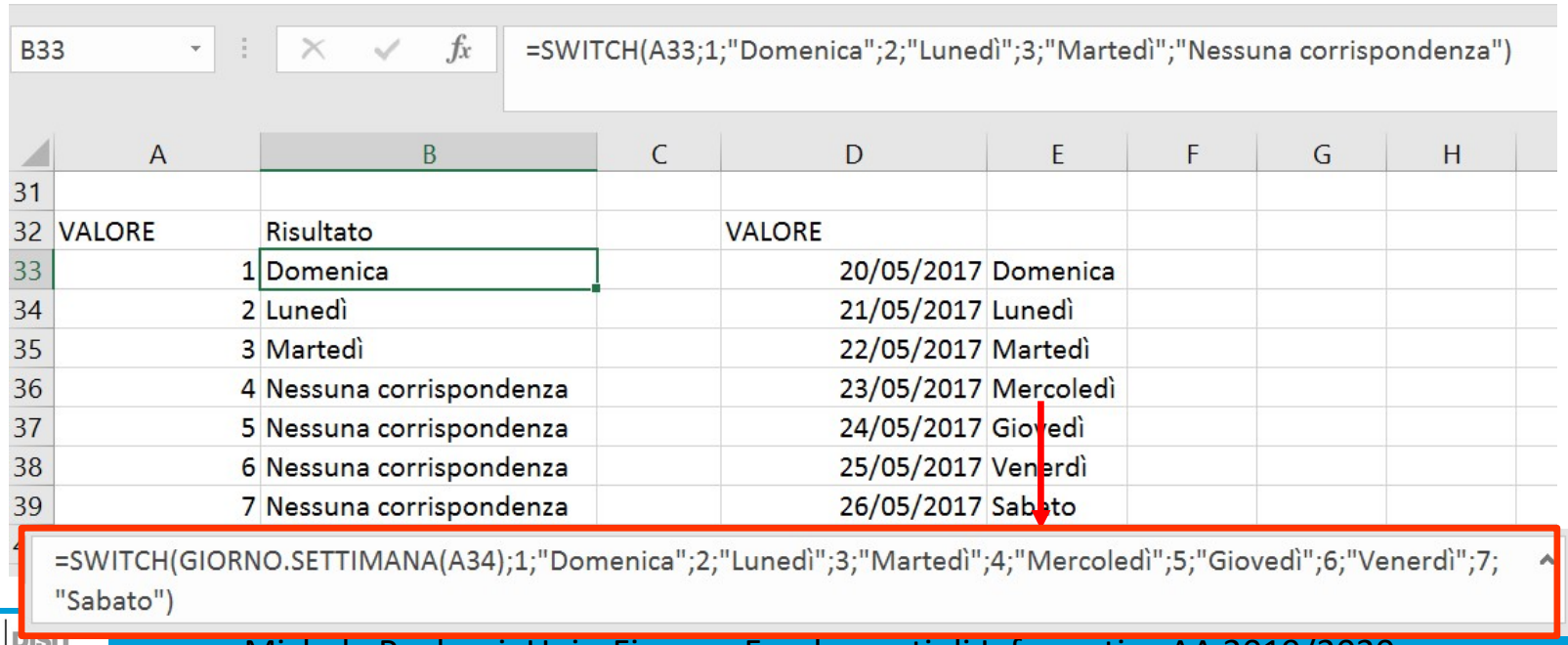
- O(arg1;arg1;…)
- Restituisce VERO se un argomento qualsiasi è VERO





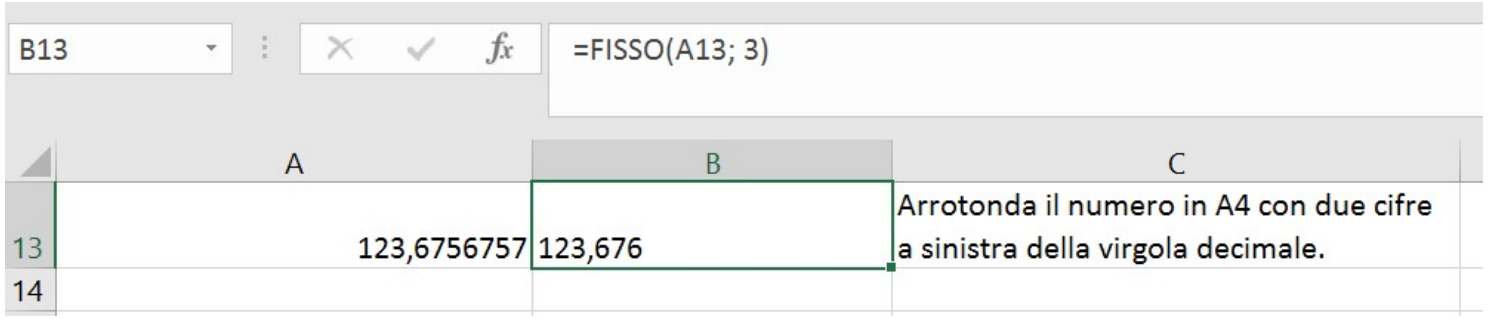
Funzioni Logiche (4)

- =SWITCH(Valore da cambiare, Valore per corrispondenza1...[2-126], Valore da restituire se esiste una corrispondenza1...[2-126], Valore da restituire se non esistono corrispondenze)
- La funzione SWITCH valuta un valore, chiamato **espressione**, rispetto a un elenco di valori e restituisce il risultato che equivale al primo valore corrispondente. Se non ci sono valori corrispondenti, verrà restituito un valore predefinito facoltativo



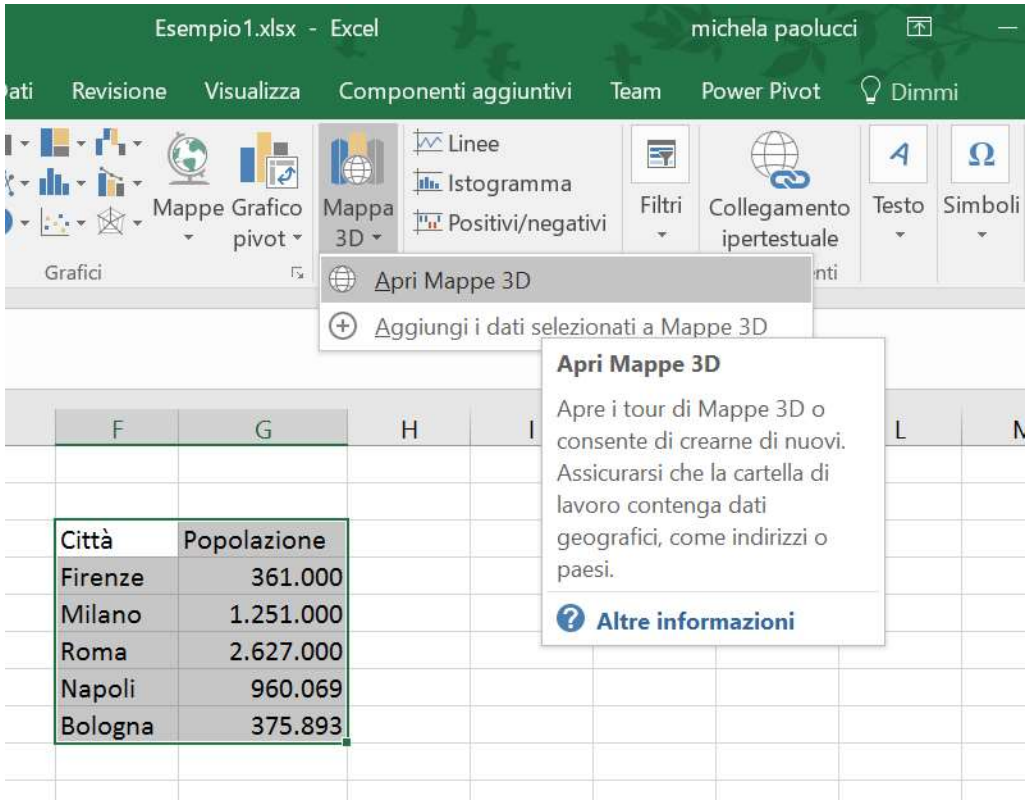
Lavorare con i numeri

- FISSO(num; [decimali]; [nessun_separatore])
  - Arrotonda un numero al numero specificato di decimali, formattandolo con i separatori delle migliaia e la virgola decimale, e restituisce il risultato in forma di testo.

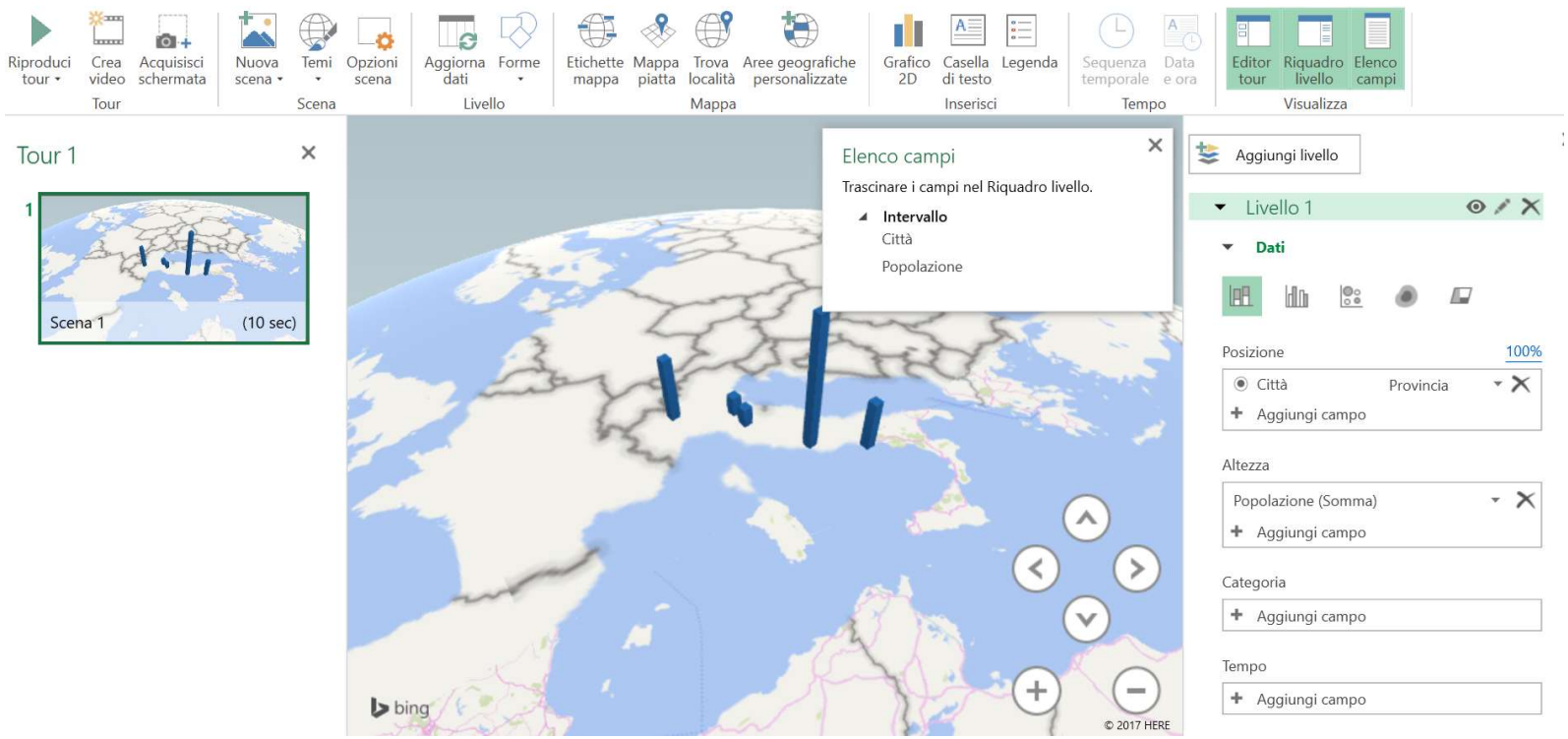


Mappe 3D (1)

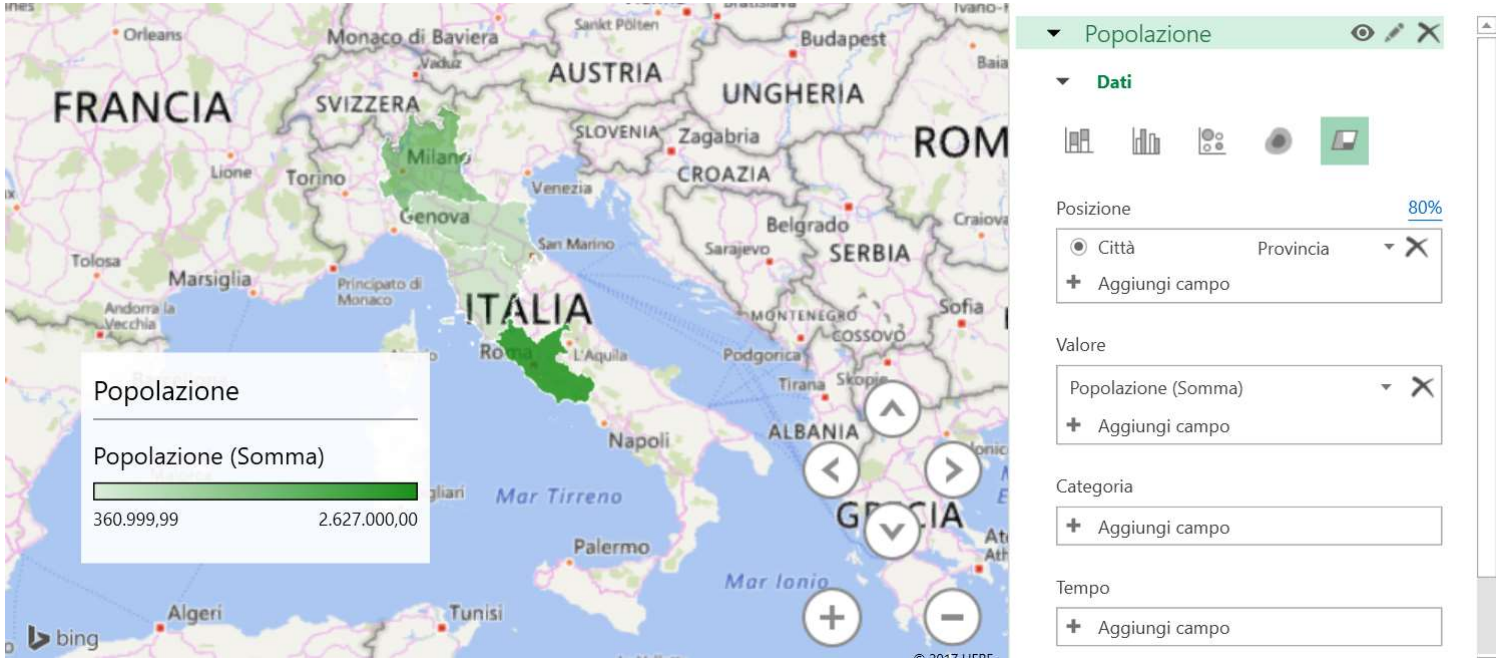
- Selezionare dati > Inserisci >Mappa 3D (componenti aggiuntivi)



Mappe 3D (2)


Riproduci tour
Crea video
Acquisisci schermata
Tour
Nuova scena
Temi
Opzioni scena
Scena
Aggiorna dati
Forme
Livello
Etichette mappa
Mappa piatta
Trova località
Aree geografiche personalizzate
Mappa
Grafico 2D
Casella di testo
Legenda
Inserisci
Sequenza temporale
Data e ora
Tempo
Editor tour
Riquadro livello
Elenco campi
Visualizza
Tour 1
1
Scena 1
(10 sec)
Elenco campi
Trascinare i campi nel Riquadro livello.
Intervallo
Città
Popolazione
bing
© 2017 HERE
Aggiungi livello
Livello 1
Dati
Posizione
100%
Città
Provincia
Aggiungi campo
Altezza
Popolazione (Somma)
Aggiungi campo
Categoria
Aggiungi campo
Tempo
Aggiungi campo

Mappe 3D (3)


Popolazione
Popolazione (Somma)
360.999,99
2.627.000,00
FRANCIA
SVIZZERA
AUSTRIA
UNGHERIA
SLOVENIA
CROAZIA
SERBIA
ITALIA
ALBANIA
Mar Tirreno
Mar Ionio
Monaco di Baviera
Budapest
Zagabria
Belgrado
Sarajevo
Venezia
Genova
Torino
Milano
Napoli
Palermo
Algeri
Tunisi
Marsiglia
Orleans
Tolosa
Podgorica
Tirana
Skopje
Sofia
bing
Popolazione
Dati
Posizione
80%
Città
Provincia
Aggiungi campo
Valore
Popolazione (Somma)
Aggiungi campo
Categoria
Aggiungi campo
Tempo
Aggiungi campo

Macro (1)

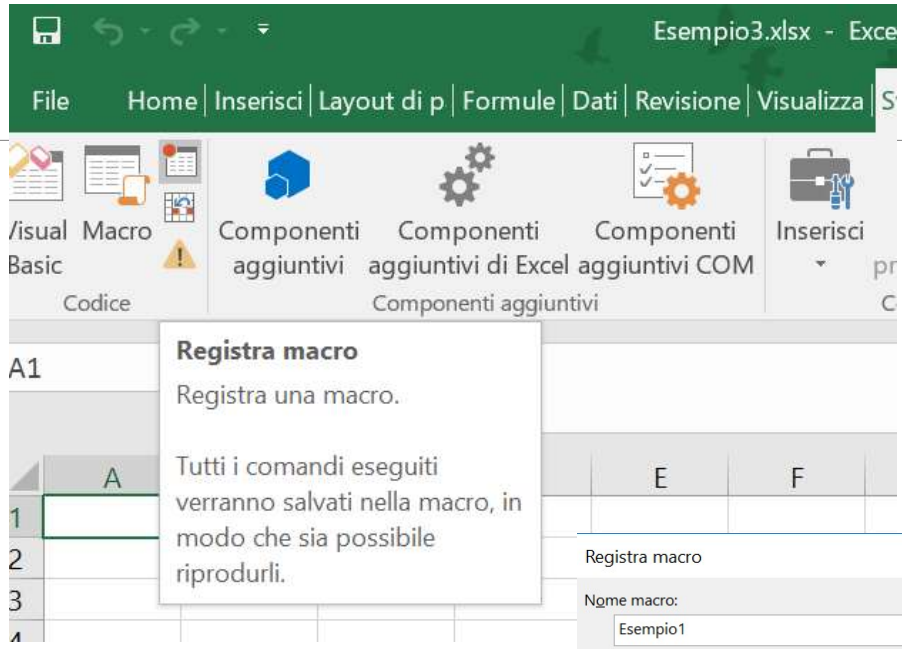
- Se si eseguono ripetutamente le stesse attività in Microsoft Excel, è possibile automatizzarle registrando una macro.
- Una macro è una azione o un insieme di azioni che è possibile eseguire per un numero illimitato di volte
- Quando si crea una macro, vengono registrati i clic del mouse e le sequenze di tasti
- Dopo aver creato una macro, è possibile modificarla per cambiarne lievemente il funzionamento.
- Se si deve creare ogni mese un rapporto per il responsabile della contabilità in modo da:
  - formattare in rosso i nomi dei clienti con i conti scaduti e applicare inoltre il grassetto
  - Allora è possibile creare ed eseguire una macro per applicare rapidamente queste modifiche di formattazione alle celle selezionate

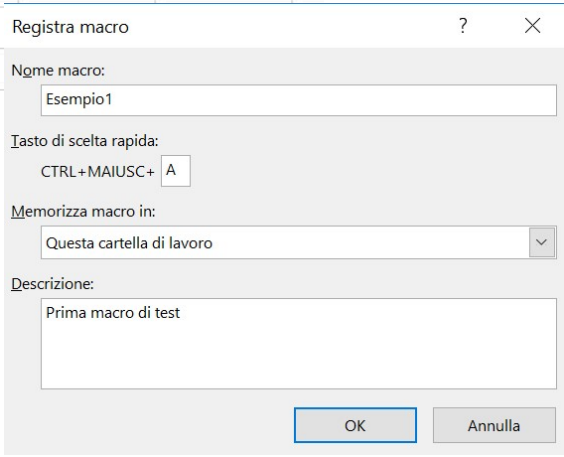
Macro (2)

- Attivare la scheda per lo sviluppo delle macro
- Verificare che nella barra multifunzione sia visualizzata la scheda **Sviluppo**.
- Se la barra **Sviluppo** non è visibile, eseguire le operazioni seguenti:
  - **File** > **Opzioni** > **Personalizzazione barra multifunzione e** Nell'elenco **Schede principali** della categoria **Personalizza barra multifunzione** fare clic su **Sviluppo** e quindi su **OK**

Macro (3)

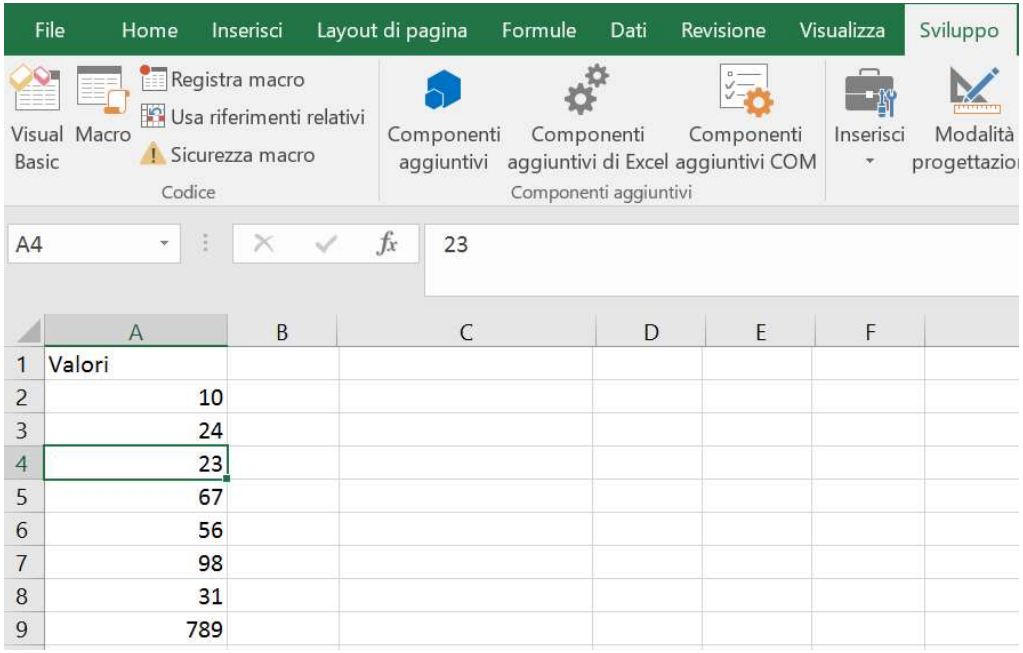
- Registrare una macro
- Nel gruppo **Codice** della scheda **Sviluppo** fare clic su **Registra macro**
- Immettere:
  - **Nome macro**
  - **Tasto di scelta rapida**
  - **Descrizione**
- Fare clic su **OK** per avviare la registrazione
- Eseguire le operazioni
- Cliccare su interrompi registrazione
- Riusare la macro quando necessario





Macro – Esempio soglia (1)

- Partenza:
  - Presenza di lista di valori interi in un foglio excel (una colonna, esempio: A2:A20)
- Scopo:
  - Creare una macro capace di determinare quali di questi valori superano una determinata soglia

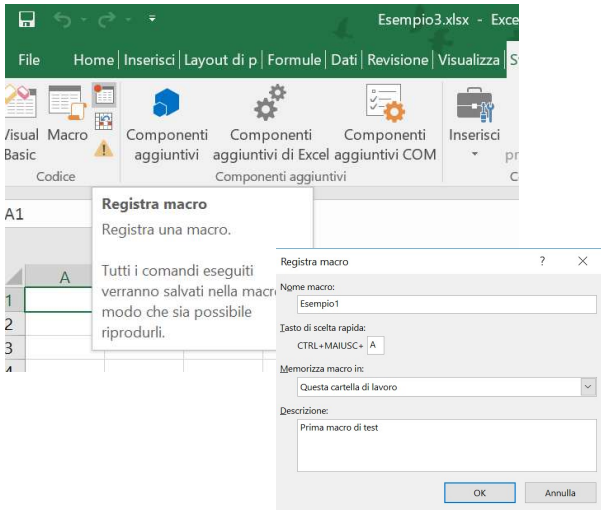


Macro – Esempio soglia (2)

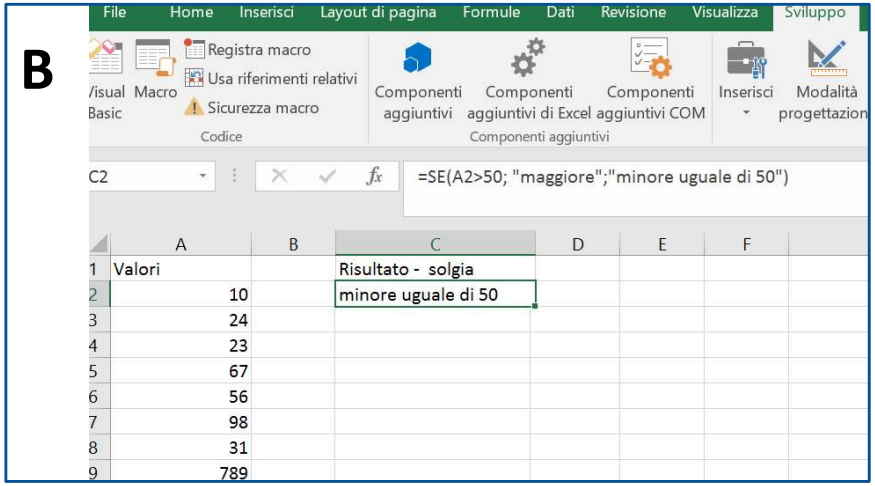
- Svolgimento:
  - A. Registrare una macro (nome, etc.)
  - B. Scrivere nella cella B2 la formula:

    =SE(A2>50; "maggiore";"minore uguale di 50")
  - C. Trascinare la formula in tutta la colonna B (B2:B20)
  - D. Interrompere la registrazione

A



B

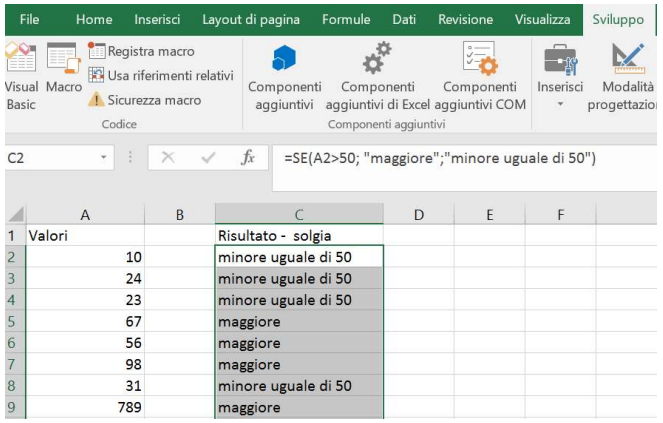


Macro – Esempio soglia (3)

- Svolgimento:
  - A. Registrare una macro (nome, etc.)
  - B. Scrivere nella cella B2 la formula:

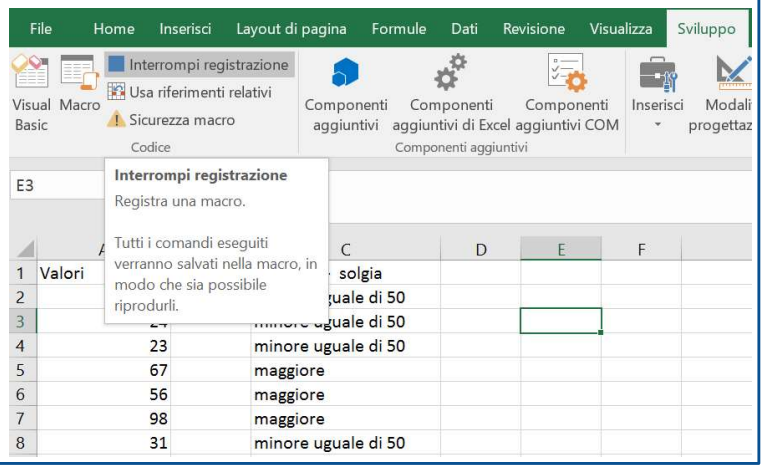
    =SE(A2>50; "maggiore";"minore uguale di 50")
  - C. Trascinare la formula in tutta la colonna B (B2:B20)
  - D. Interrompere la registrazione

C



	A	B	C	D	E	F
1	Valori		Risultato - solgia			
2	10		minore uguale di 50			
3	24		minore uguale di 50			
4	23		minore uguale di 50			
5	67		maggiore			
6	56		maggiore			
7	98		maggiore			
8	31		minore uguale di 50			
9	789		maggiore			

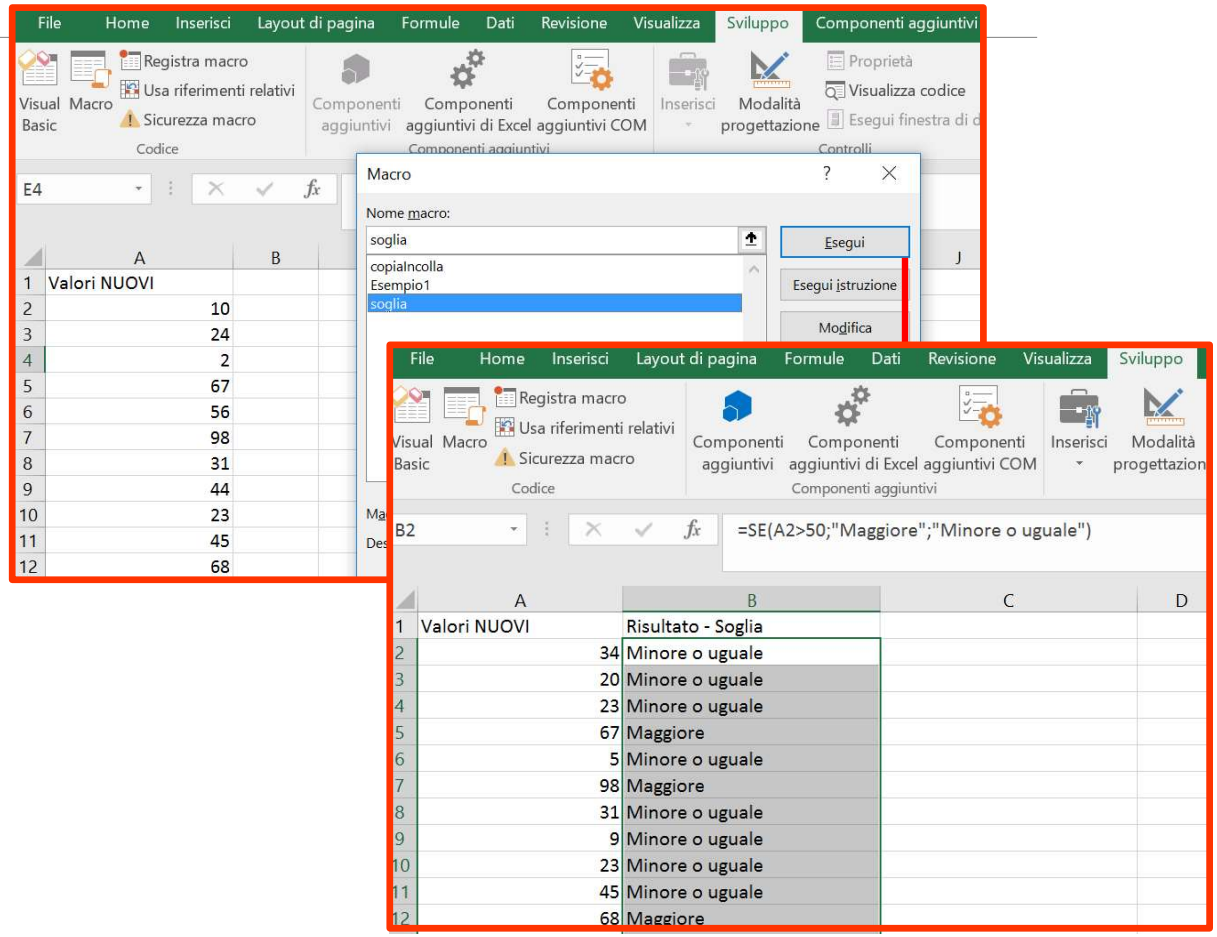
D



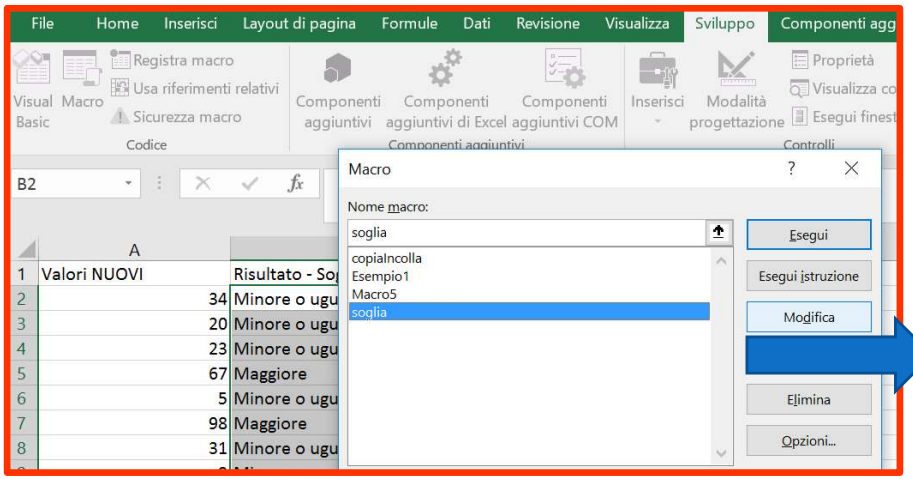
	A	B	C	D	E	F
1	Valori		solgia			
2			uale di 50			
3			uale di 50			
4	23		minore uguale di 50			
5	67		maggiore			
6	56		maggiore			
7	98		maggiore			
8	31		minore uguale di 50			

Macro – Esempio soglia (4)

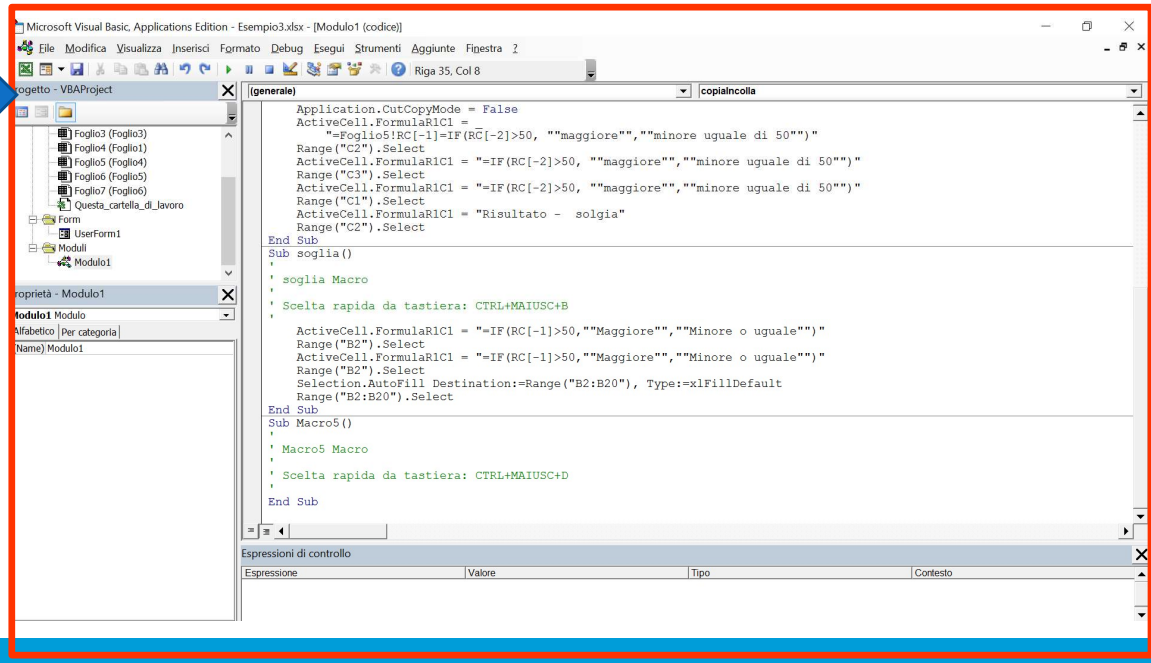
- Uso:
  - Andare in un nuovo foglio excel
  - Partire da una nuova lista di valori
  - Eseguire la macro:
    - Sviluppo > Macro > scegliere la macro



Macro – Esempio soglia (5)

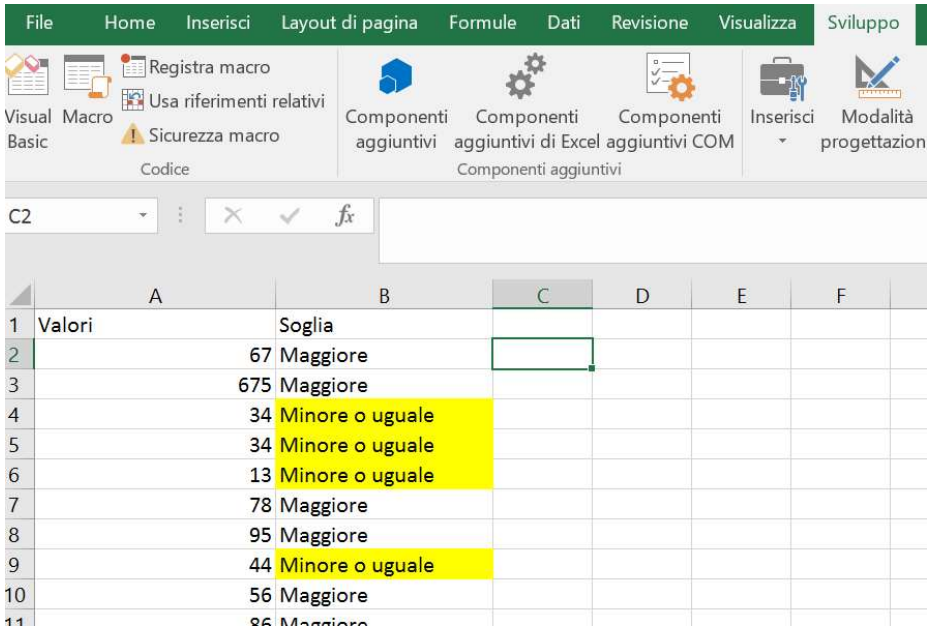


- Modifica di una macro tramite editor Visual Basic



Macro – Esempio soglia (6)

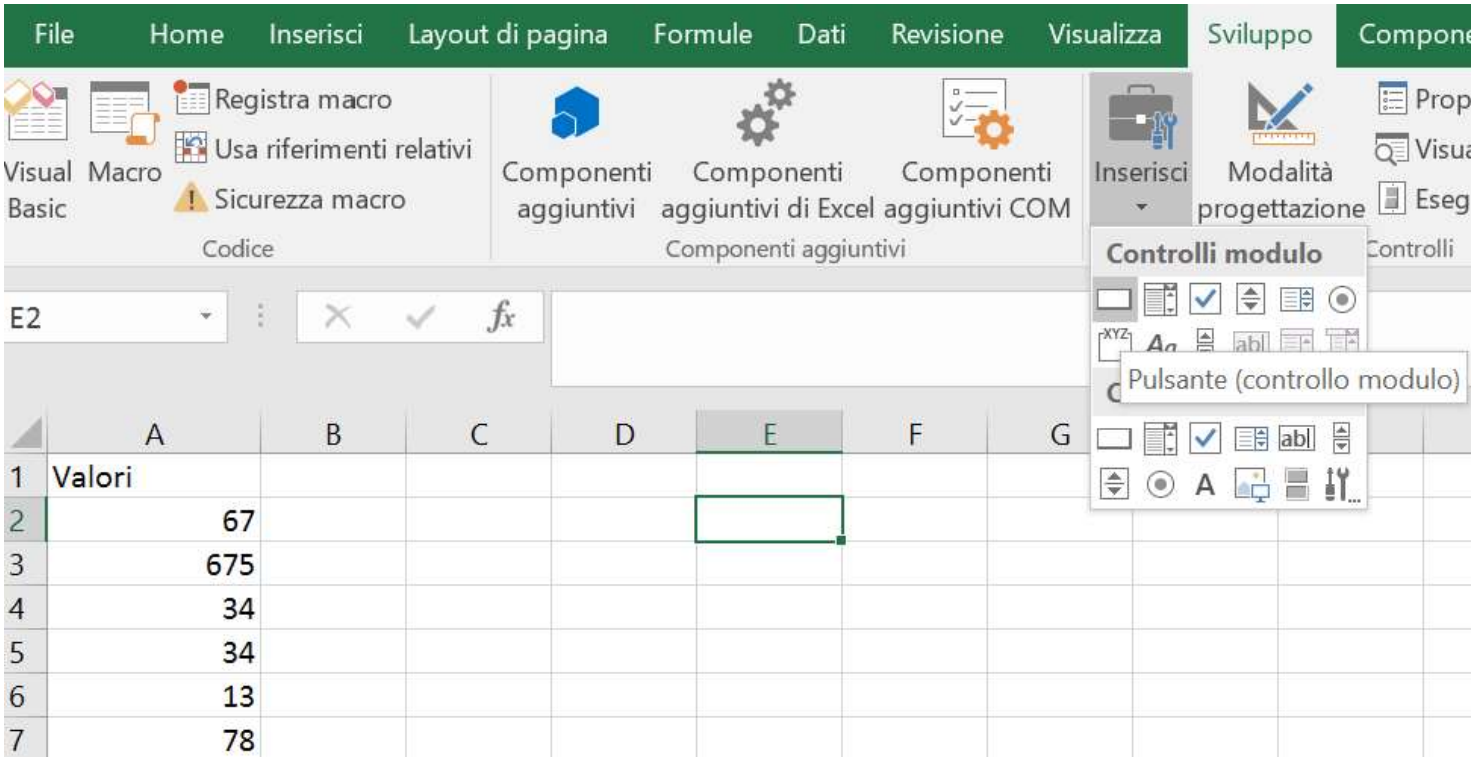
- Variante:
  - E’ possibile anche registrare le operazioni di formattazione alle celle.
  - Ad esempio ‘mettere sfondo giallo alle caselle che fanno riferimento a valori minori o uguali a 50’



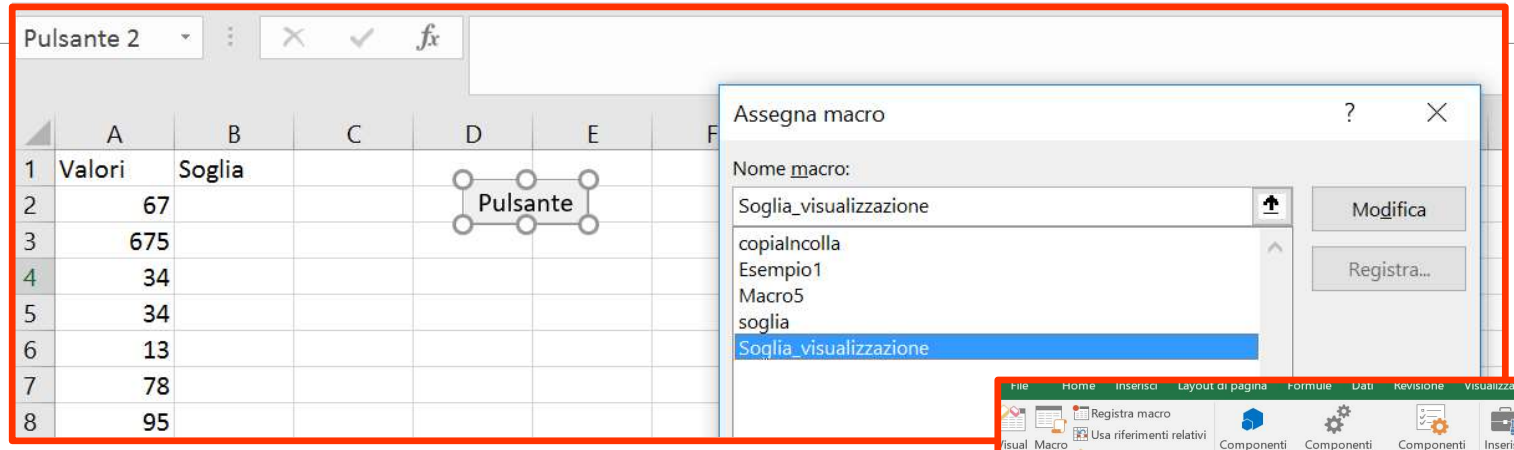
	A	B	C	D	E	F
1	Valori	Soglia				
2	67	Maggiore				
3	675	Maggiore				
4	34	Minore o uguale				
5	34	Minore o uguale				
6	13	Minore o uguale				
7	78	Maggiore				
8	95	Maggiore				
9	44	Minore o uguale				
10	56	Maggiore				

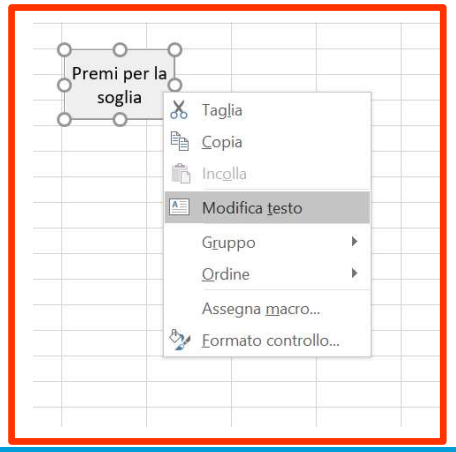
Macro – Esempio soglia (7)

- Richiamare la Macro agganciandola ad un bottone

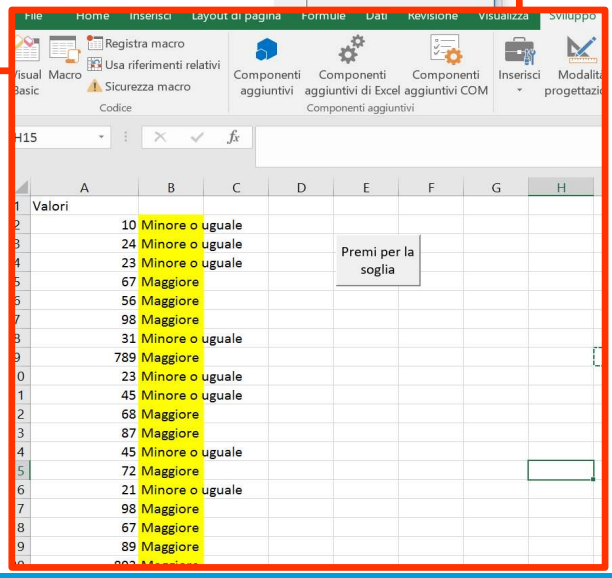


Macro – Esempio soglia (8)





- Associazione bottone-Macro
- Modifica vista bottone
- Click sul bottone per eseguire macro



Tipi di pulsante

Tipo di controllo	Descrizione
Etichetta	Testo aggiunto a un foglio di lavoro o a un modulo per fornire informazioni su un controllo, sul foglio di lavoro o sul modulo.
Casella di gruppo	Bordo ed etichetta che raggruppano controlli correlati, quali pulsanti di opzione o caselle di controllo.
Pulsante	Pulsante che esegue una macro quando viene scelto.
Casella di controllo	Casella che attiva o disattiva un'opzione. In un foglio o in un gruppo è possibile selezionare contemporaneamente più caselle di controllo.
Pulsante di opzione	Pulsante che seleziona un'opzione in un gruppo contenuto in una casella di gruppo. Poiché in un gruppo è possibile selezionare un solo pulsante di opzione, questo viene utilizzato quando è consentita una sola possibilità di scelta.
Casella di riepilogo	Casella contenente un elenco di voci.
Casella combinata	Casella di riepilogo a discesa. La voce selezionata nella casella di riepilogo è visualizzata nella casella di testo.
Barra di scorrimento	Controllo che scorre un intervallo di valori quando si fa clic sulle frecce di scorrimento o quando si trascina la casella di scorrimento. È possibile muoversi in una successiva pagina di valori facendo clic tra la casella di scorrimento e la freccia di scorrimento.
Casella di selezione	Pulsante con una freccia Su e una freccia Giù che è possibile associare a una cella e che consente di aumentare o diminuire un valore facendo clic sulla freccia corrispondente.

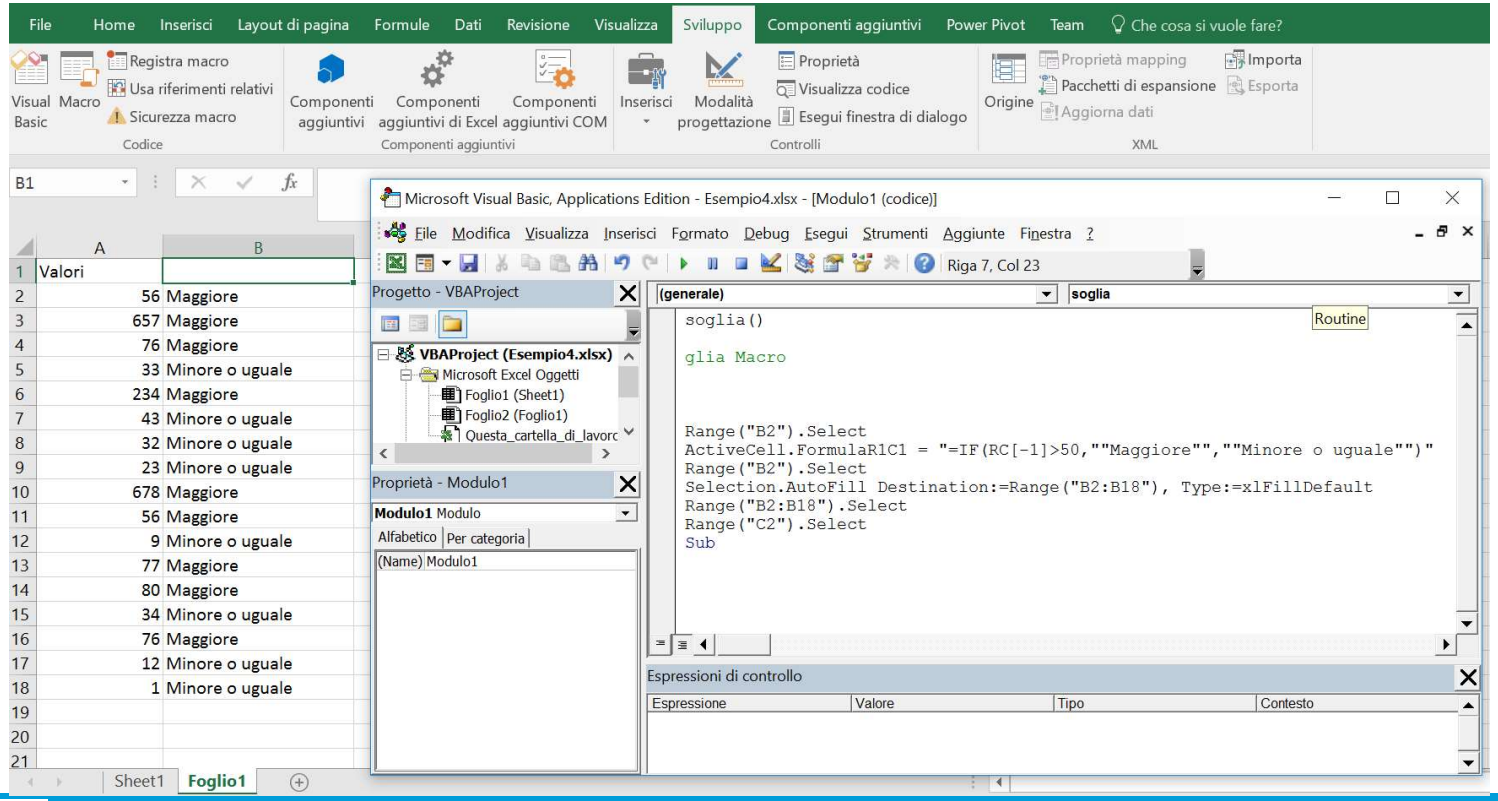
Gestire le macro

Gestire le macro

- Con l'Editor di Visual Basic è possibile:
  - modificare le macro
  - copiare macro da un modulo all'altro
  - copiare macro tra diverse cartelle di lavoro
  - rinominare i moduli che memorizzano le macro
  - rinominare le macro stesse
- Se ad esempio si desidera che la macro per la sottolineatura di alcune celle, sarà possibile registrare un'altra macro per formattare la cella, quindi copiare le istruzioni da quest'ultima nella prima macro

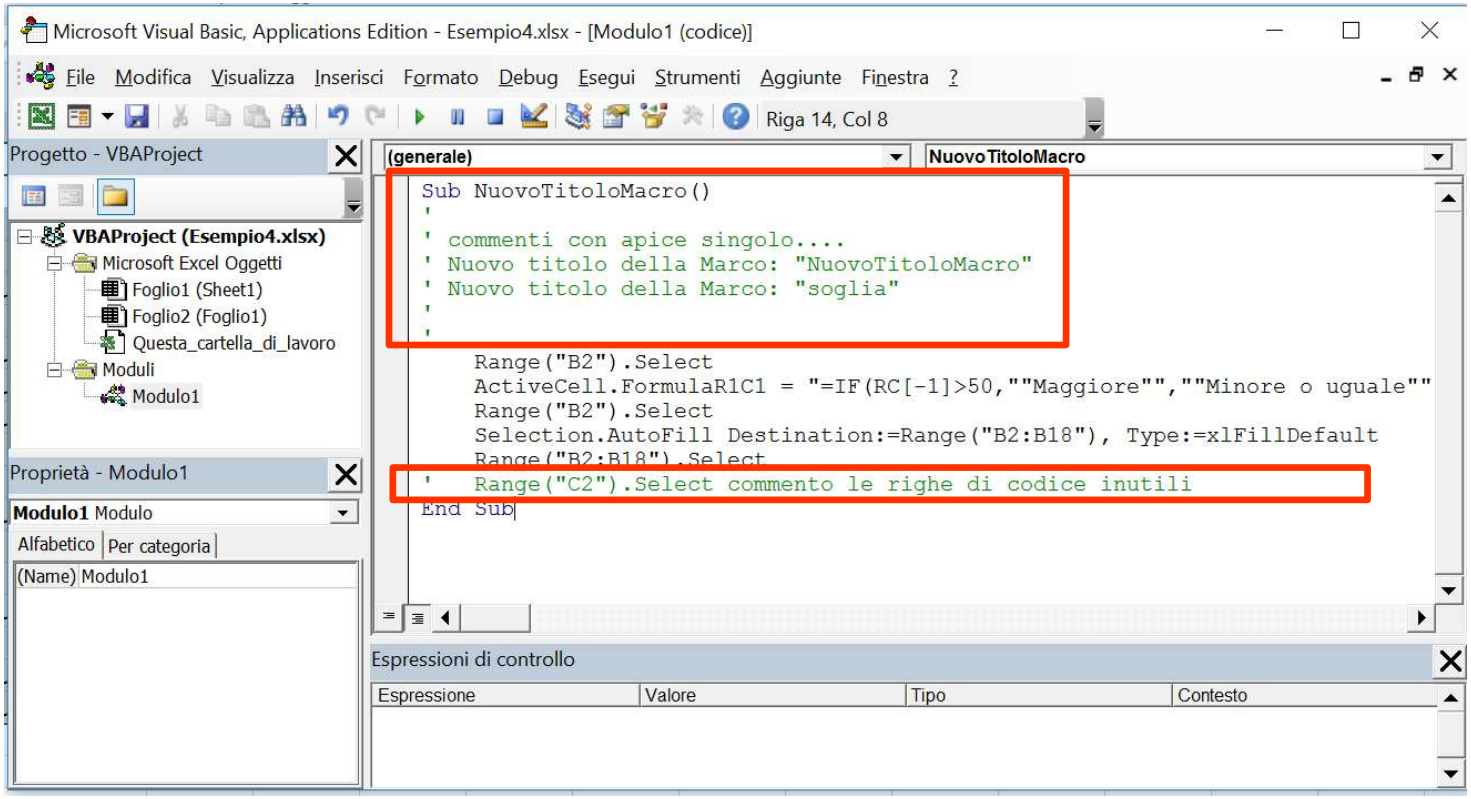
Modificare una macro (1)

- Supponiamo di partire dal codice per determinare quali valori superano una data soglia…



Modificare una macro (2)

- Modifica del titolo di una macro
- Commenti (con apice singolo)



Modificare una macro (3)

- Variazione del Range (celle su cui eseguire le operazioni)

```
(generale)                                   NuovoTitoloMacro
Sub NuovoTitoloMacro()
'
' commenti con apice singolo....
' Nuovo titolo della Marco: "NuovoTitoloMacro"
' Nuovo titolo della Marco: "soglia"
'
'
    Range("B2").Select
    ActiveCell.FormulaR1C1 = "=IF(RC[-1]>50,""Maggiore"",""Minore o uguale"")"
' --variazione range da B2:B19 a B2:B18
    Selection.AutoFill Destination:=Range("B2:B16"), Type:=xlFillDefault
    Range("B2:B16").Select
'   Range("C2").Select commento le righe di codice inutili
End Sub
```

ESECUZIONE

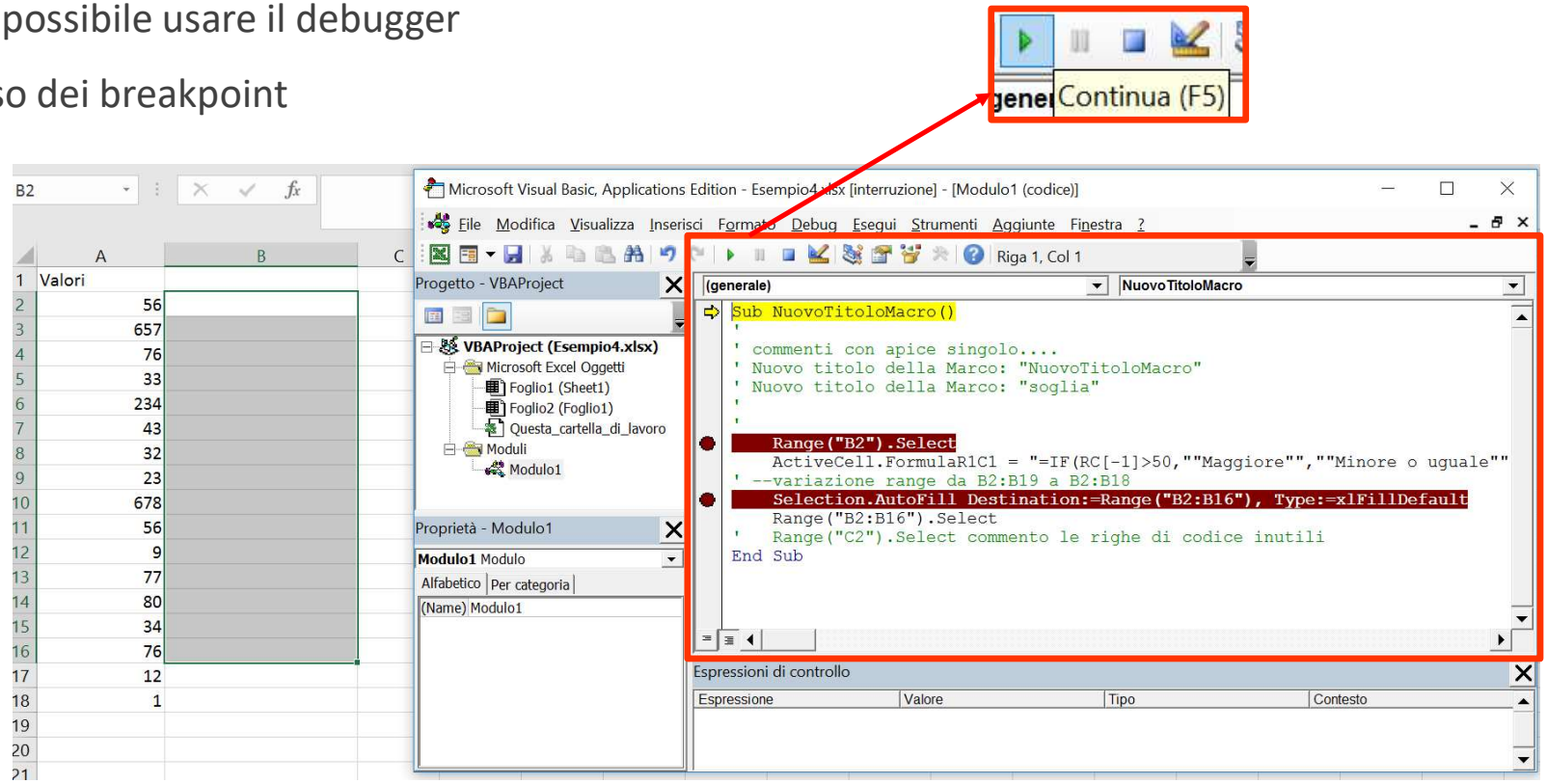
B2 =SE(A2>50;

	A	B	C
1	Valori		
2	56	Maggiore	
3	657	Maggiore	
4	76	Maggiore	
5	33	Minore o uguale	
6	234	Maggiore	
7	43	Minore o uguale	
8	32	Minore o uguale	
9	23	Minore o uguale	
10	678	Maggiore	
11	56	Maggiore	
12	9	Minore o uguale	
13	77	Maggiore	
14	80	Maggiore	
15	34	Minore o uguale	
16	76	Maggiore	
17	12		
18	1		

- Sia il calcolo della 'soglia' che la selezione finale delle celle sono NON più nel range B2:B18 ma coinvolgono le celle B2:B16

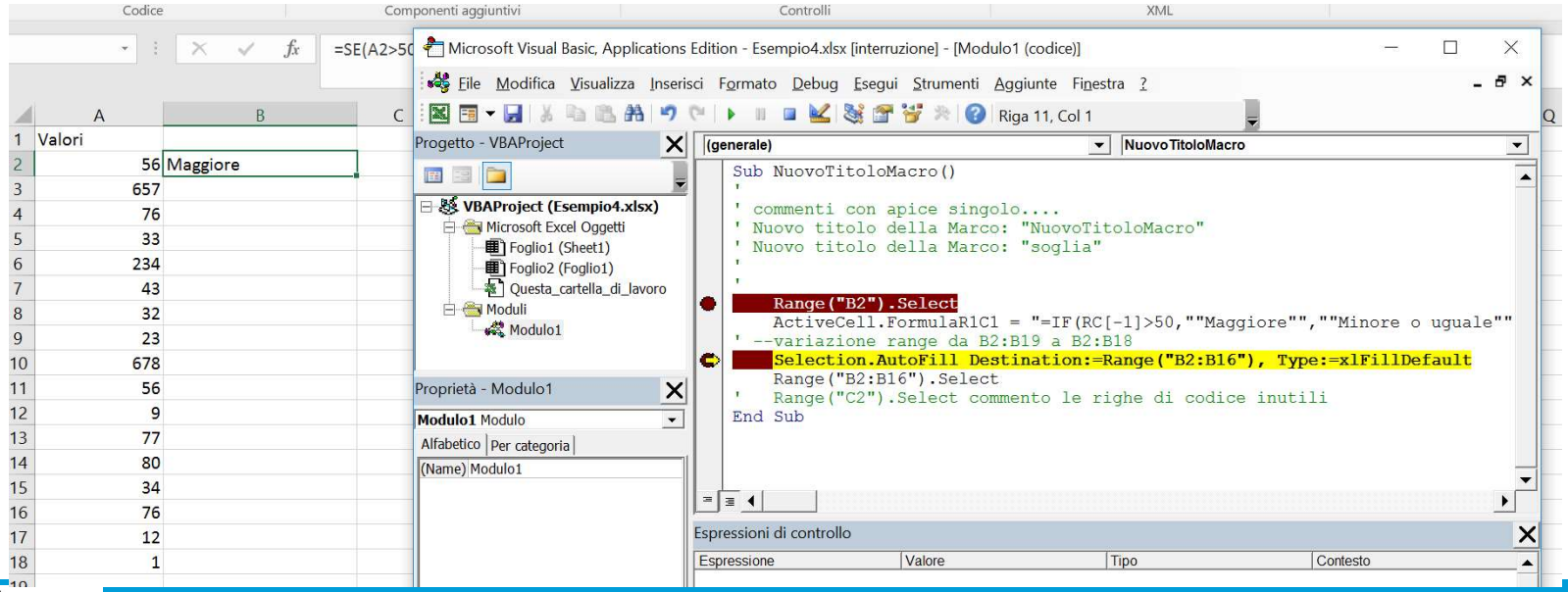
Uso del debugger (1)

- E' possibile usare il debugger
- Uso dei breakpoint



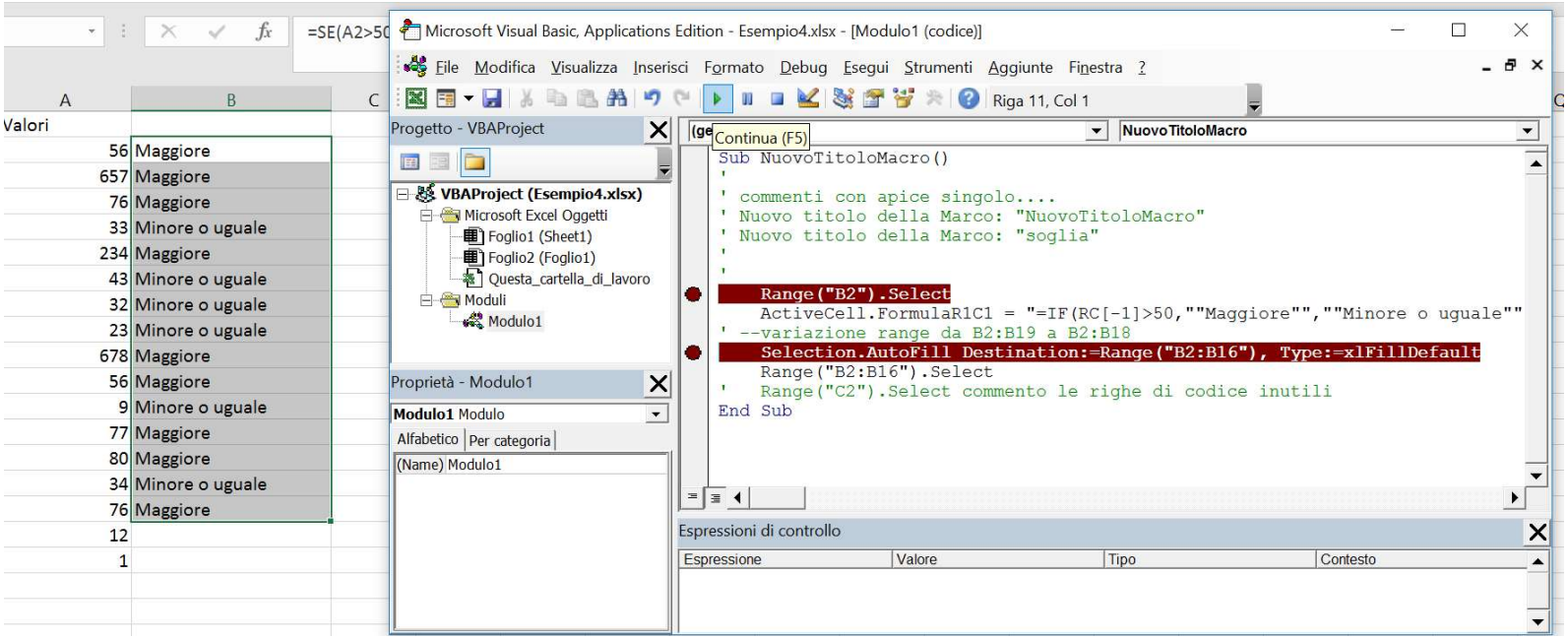
Uso del debugger (2)

- Si preme due volte su F5:
  - il debugger si ferma al secondo breakpoint
  - Il debugger ha eseguito la prima riga di codice sul foglio excel (ovvero ha calcolato la soglia nella cella B2)
  - Il debugger deve ancora eseguire le altre istruzioni



Uso del debugger (3)

- Fine della esecuzione:
  - Tutte le operazioni VB sono state eseguite

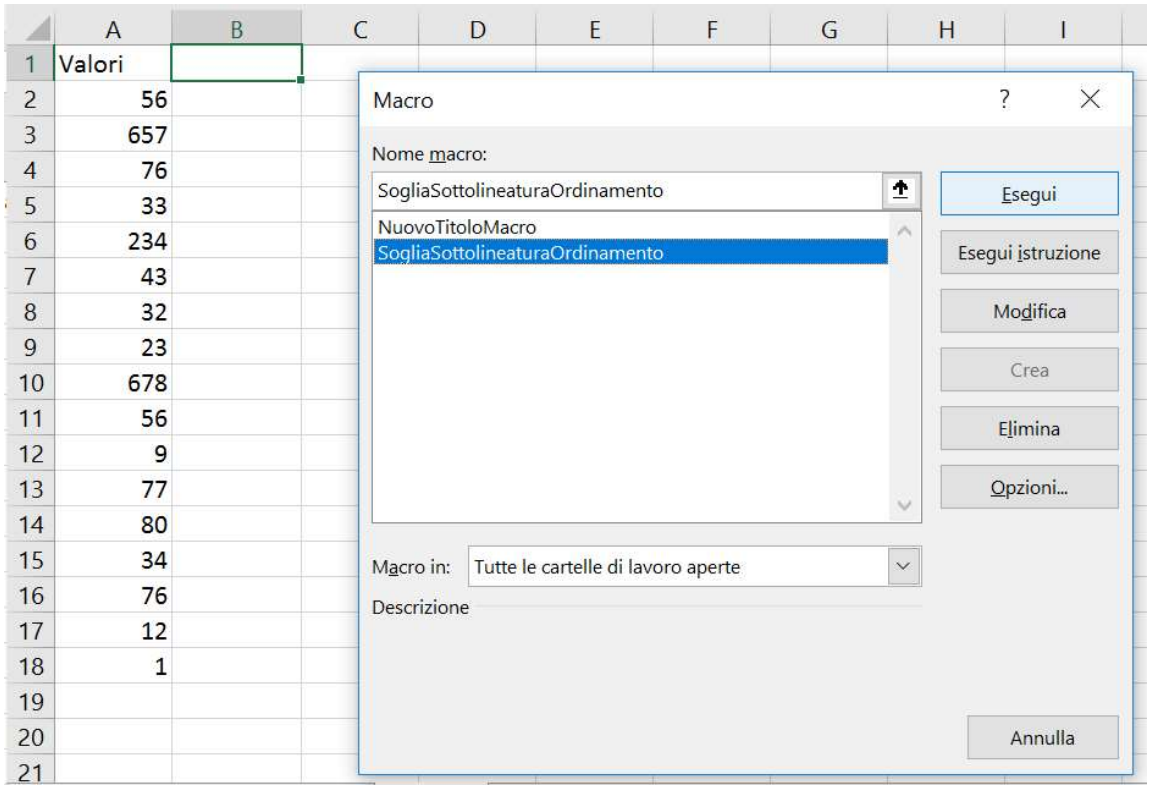


Modificare una macro (4)

- Partendo da un esempio simile e quindi sempre da una lista di valori nelle celle A2:A18
- Si inizia la registrazione della macro
- Si scrive il titolo della colonna B (cella B1)
- Si cambia lo sfondo della cella B1
- Si selezionano le celle dei valori (A2:A18) e si ordinano per valore (dal più piccolo al più grande da menu' > dati>ordina)
- Si scrive la formula per la soglia nella cella B2
- Si trascina la formula nelle altre celle (B3:B18)
- Si finisce la registrazione della macro

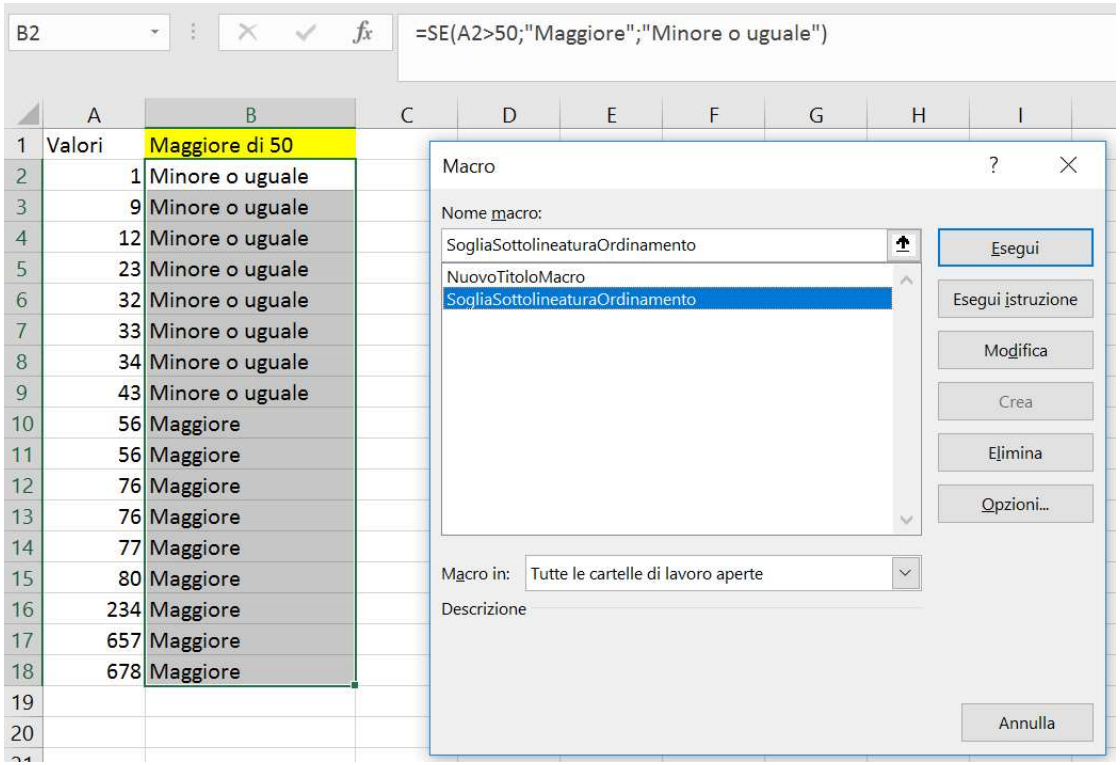
Modificare una macro (5)

- Applicazione della macro: stato iniziale



Modificare una macro (6)

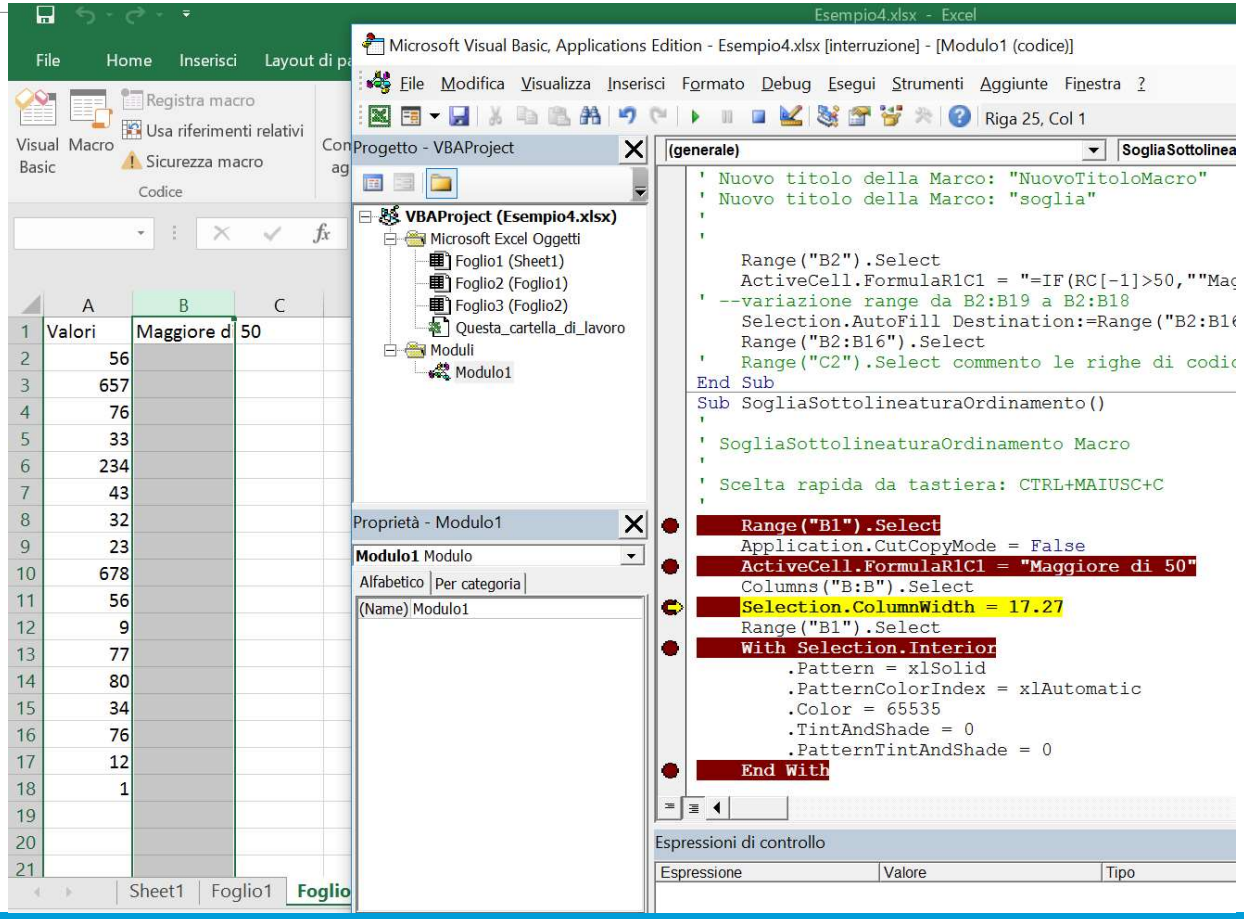
- Applicazione della macro: stato finale



	A	B
1	Valori	Maggiore di 50
2	1	Minore o uguale
3	9	Minore o uguale
4	12	Minore o uguale
5	23	Minore o uguale
6	32	Minore o uguale
7	33	Minore o uguale
8	34	Minore o uguale
9	43	Minore o uguale
10	56	Maggiore
11	56	Maggiore
12	76	Maggiore
13	76	Maggiore
14	77	Maggiore
15	80	Maggiore
16	234	Maggiore
17	657	Maggiore
18	678	Maggiore
19		
20		

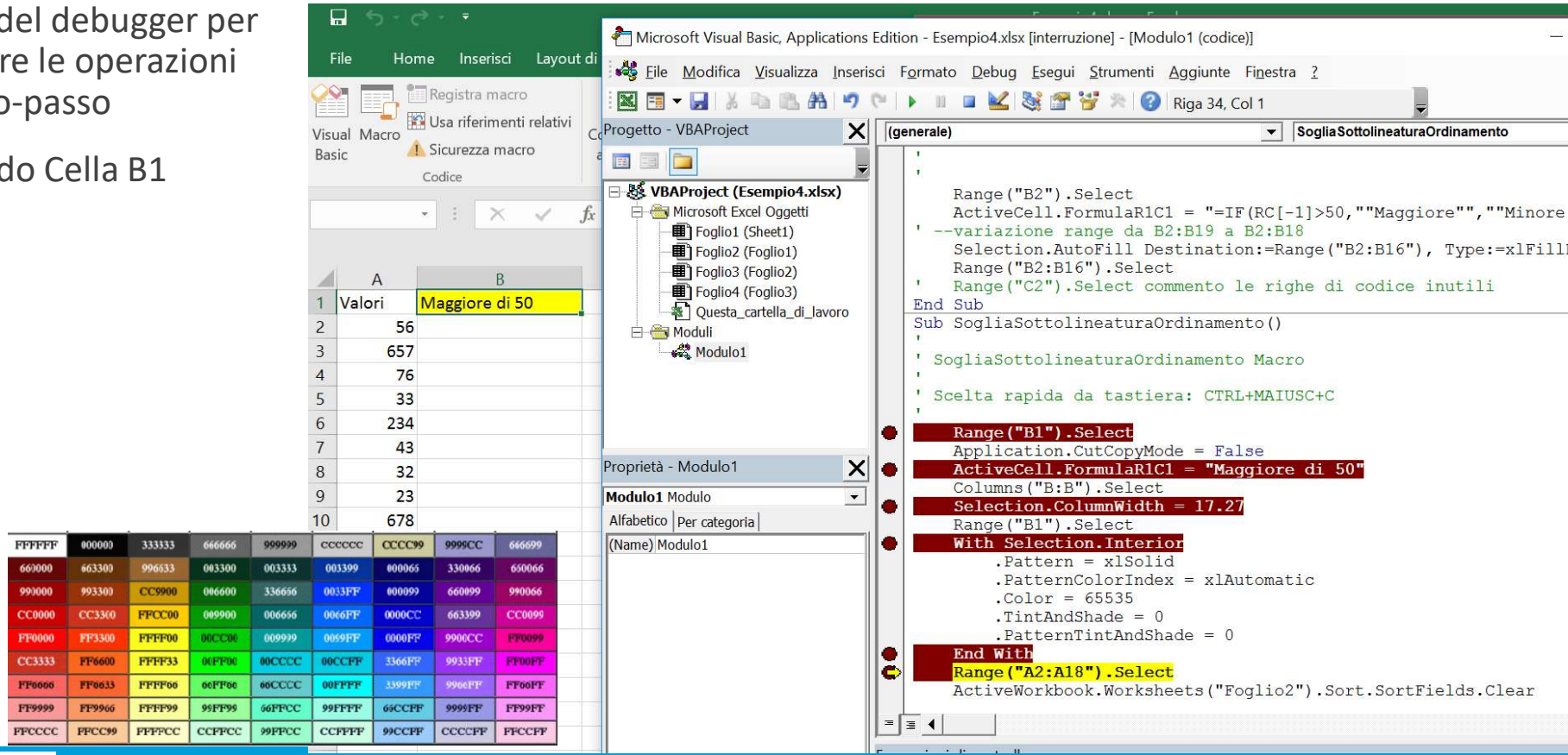
Modificare una macro (7)

- Si scrive il titolo in B1
- Si Allarga la colonna B (Section.ColumnWidth)



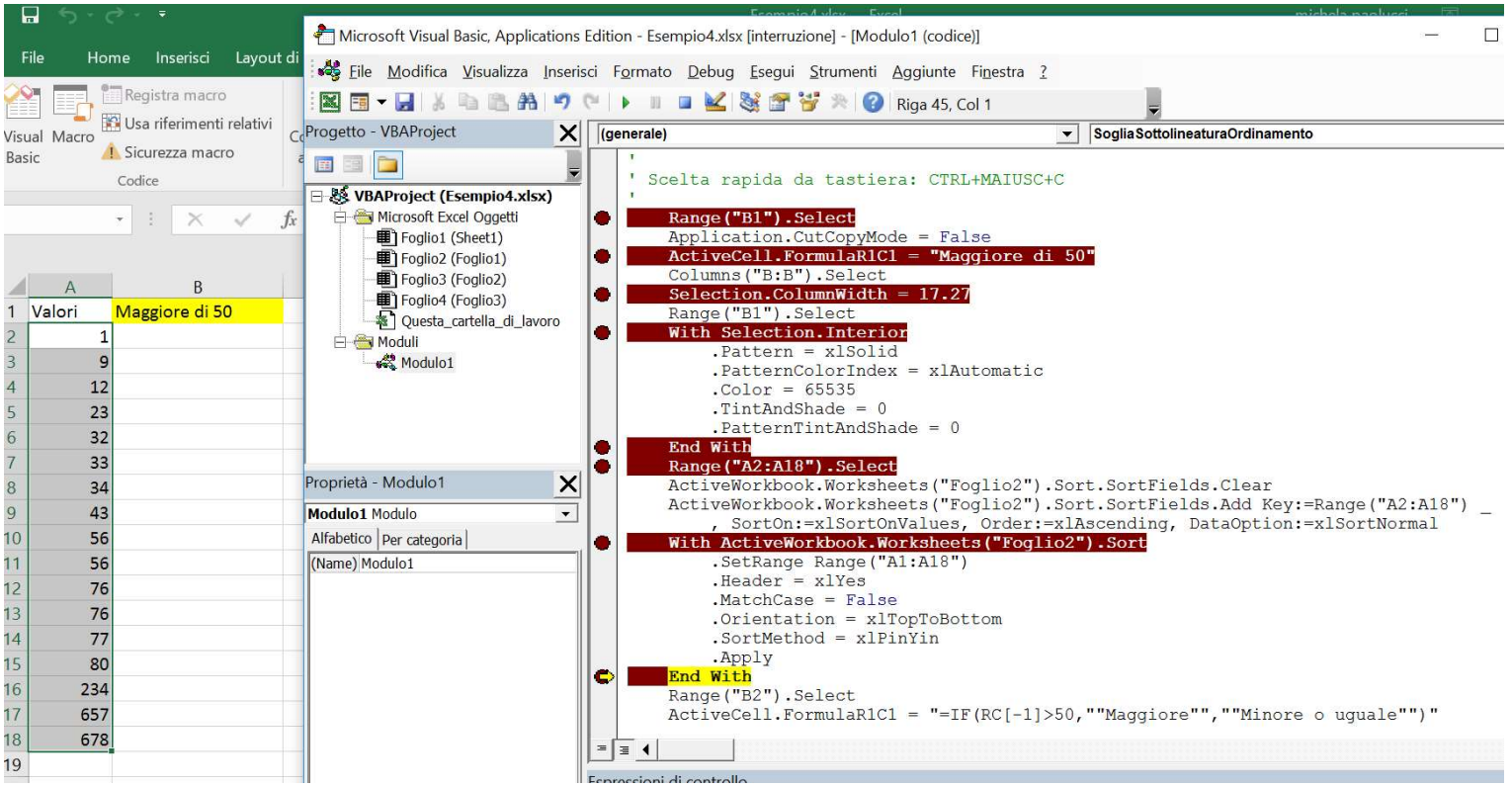
Modificare una macro (8)

- Uso del debugger per vedere le operazioni passo-passo
- Sfondo Cella B1



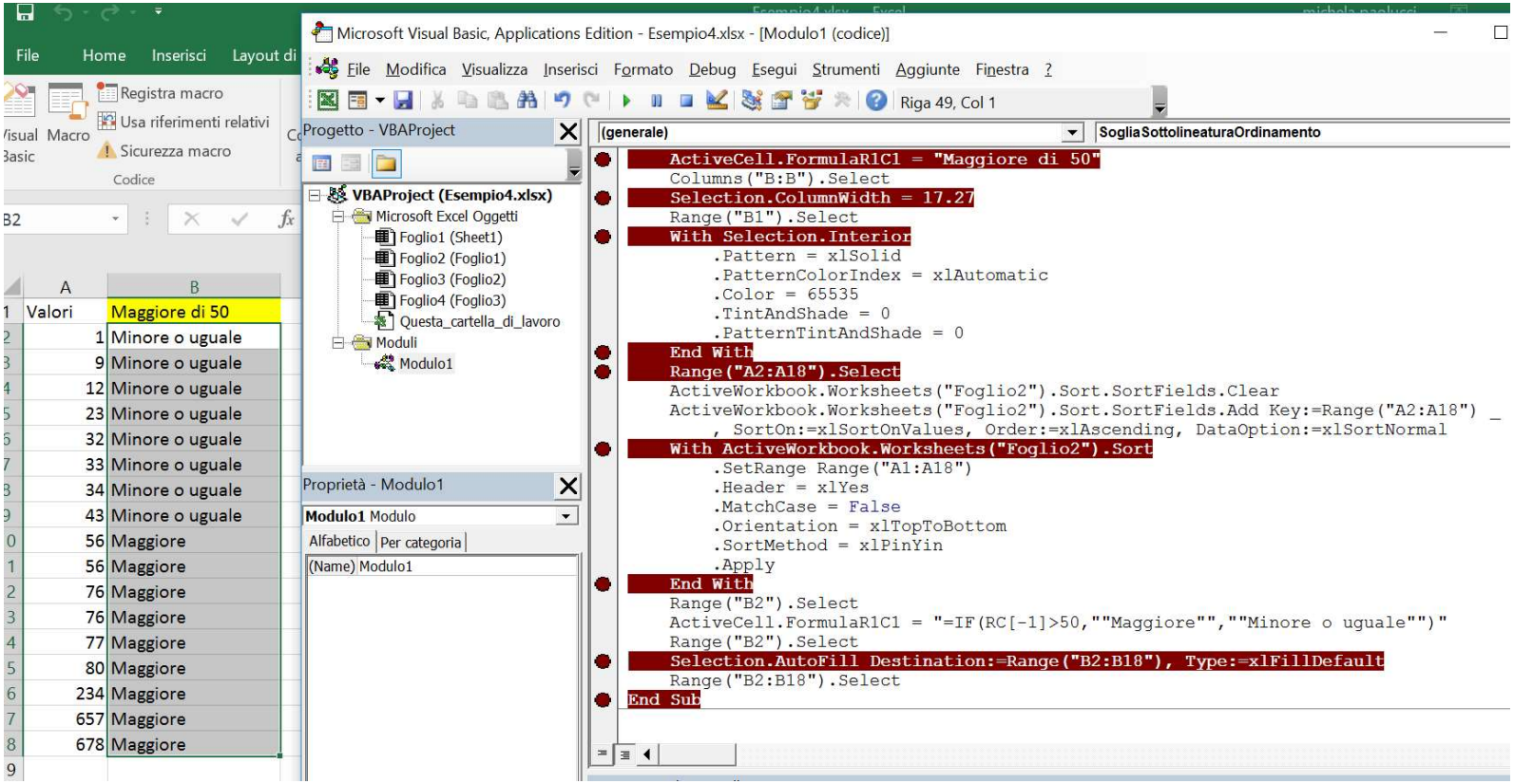
Modificare una macro (9)

- Uso del debugger per vedere le operazioni passo-passo
- Ordinamento valori celle A2:A18



Modificare una macro (10)

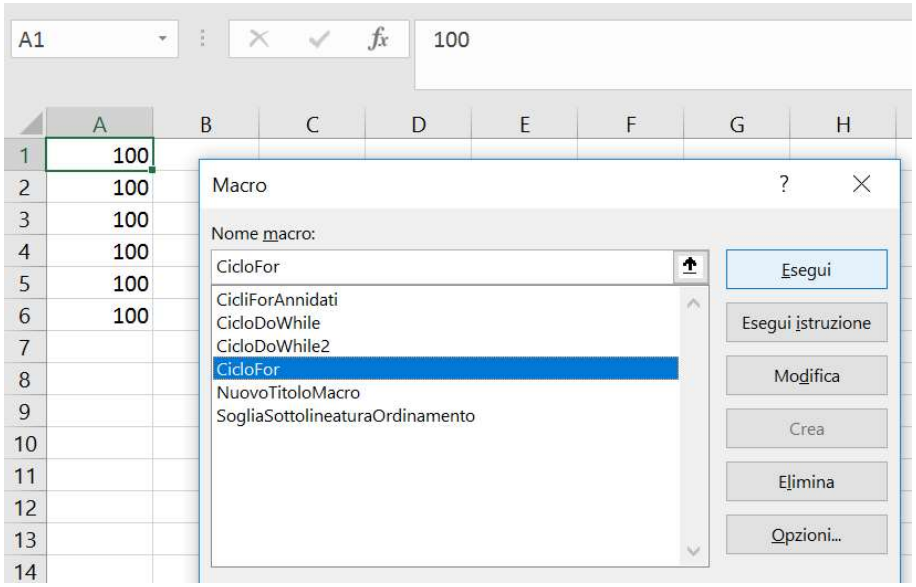
- Uso del debugger per vedere le operazioni passo-passo
- Fase finale: si scrive la formula nella B2 e si trascina B3:B18



Scrivere una macro (1)

- Esempio Ciclo for

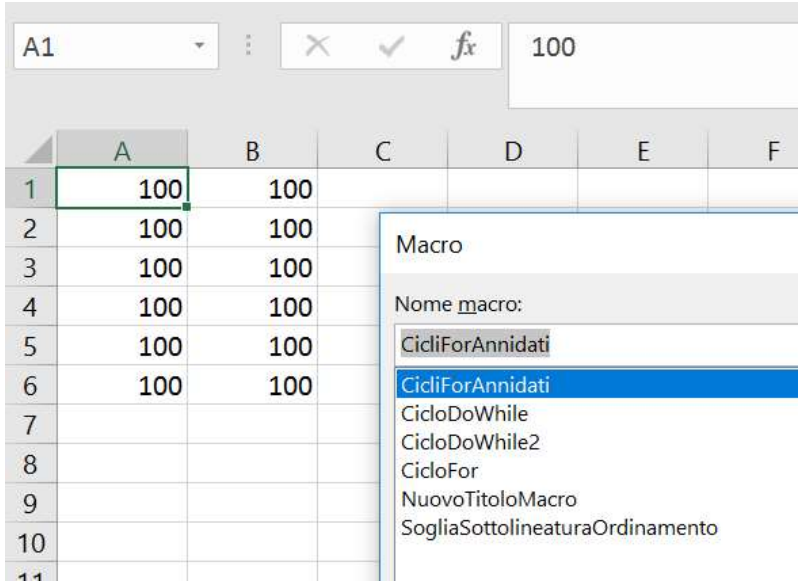
```
Sub CicloFor()
' Esempio di ciclo for
' Cells(x,y) --> Excel VBA considera la
' riga 1 e colonna 1
' esempio: Cells(1,3) è la cella C1
Dim i As Integer

For i = 1 To 6
    Cells(i, 1).Value = 100
Next i
End Sub
```



Scrivere una macro (2)

- Esempio Cicli for annidati:

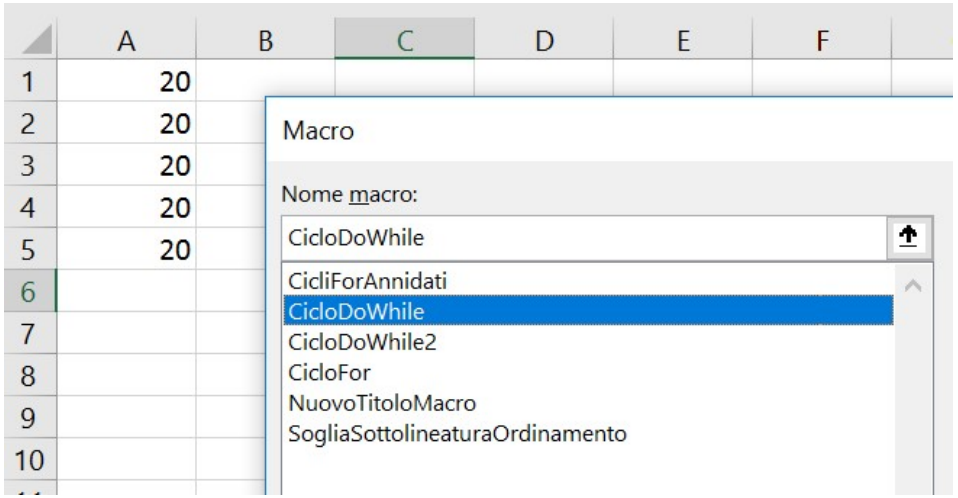
```
Sub CicliForAnnidati()
' Esempio di cicli for annidati
' Cells(x,y) --> Excel VBA considera la
' riga 1 e colonna 1
' esempio: Cells(1,3) è la cella C1

Dim i As Integer, j As Integer

For i = 1 To 6
    For j = 1 To 2
        Cells(i, j).Value = 100
    Next j
Next i
End Sub
```



Scrivere una macro (3)

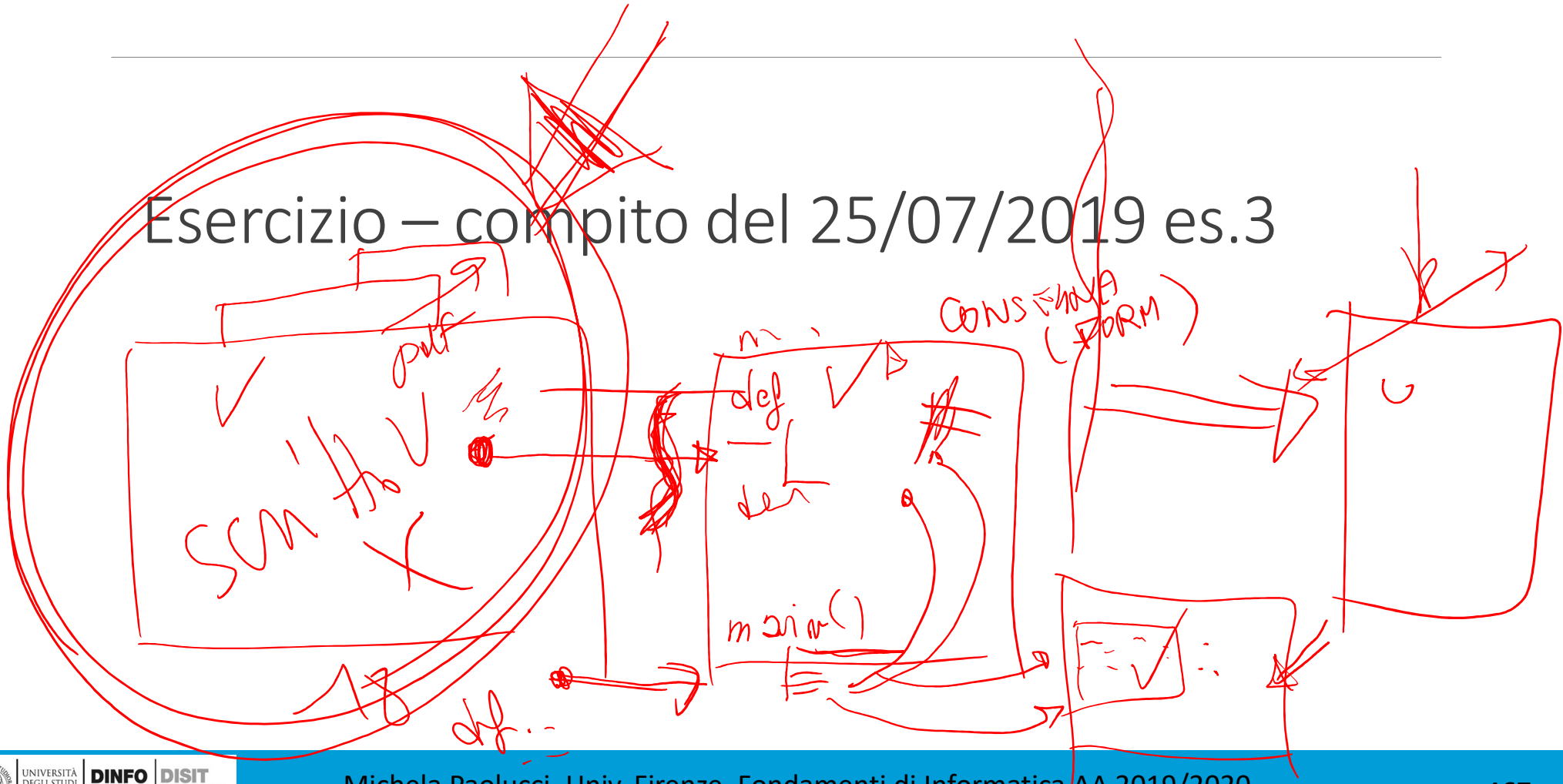
- Esempio Ciclo Do While:

```
Sub CicloDoWhile()
' Esempio di cicli for annidati
' Cells(x,y) --> Excel VBA considera la
' riga 1 e colonna 1
' esempio: Cells(1,3) è la cella C1
Dim i As Integer
i = 1

Do While i < 6
    Cells(i, 1).Value = 20
    i = i + 1
Loop
End Sub
```



Esercizi – Compiti

Esercizio – compito del 25/07/2019 es.3



Esercizio – compito del 25/07/2019 es.3 (1)

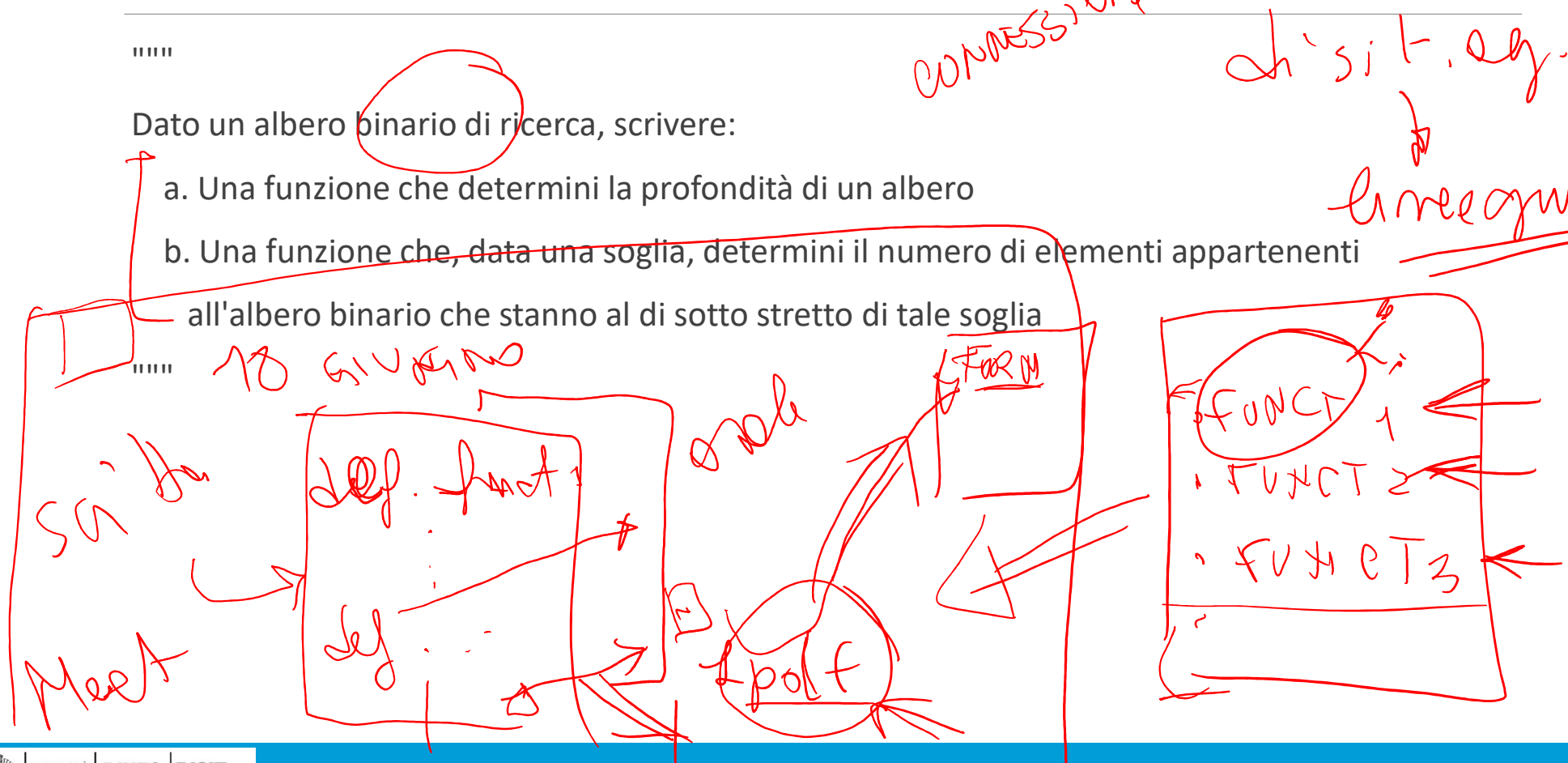
"""

Dato un albero binario di ricerca, scrivere:

a. Una funzione che determini la profondità di un albero

b. Una funzione che, data una soglia, determini il numero di elementi appartenenti all'albero binario che stanno al di sotto stretto di tale soglia

"""

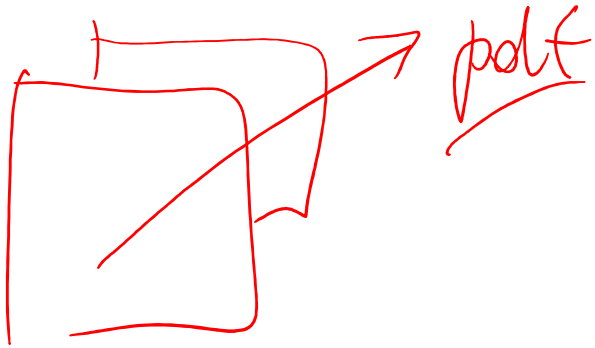


Esercizio – compito del 25/07/2019 es.3 (2)

```
def calcola_prodondita(root):
    if root is None:
        return 0
    if root.left is None and root.right is None:
        return 0
    depth_sx = calcola_prodondita(root.left)
    depth_dx = calcola_prodondita(root.right)

    if depth_sx > depth_dx:
        return 1 + depth_sx
    else:
        return 1 + depth_dx
```

Esercizio – compito del 25/07/2019 es.3 (3)

```
def conta_sotto_valore(root, valore):
    conto = 0
    if root is not None:
        conto_sx = conta_sotto_valore(root.left, valore)
        conto_dx = conta_sotto_valore(root.right, valore)
        conto = conto_sx + conto_dx
        if root.data < valore:
            conto += 1
    return conto
```



Esercizio – compito del 25/07/2019 es.3 (4)

```
import tree
import random
[...]
def main():
    valori = list()
    for _ in range(100):
        valori.append(random.randint(1, 100))
    valore = valori[random.randint(0, len(valori) - 1)]
    root = tree.populate_tree_from(valori)
    profondita = calcola_prodondita(root)
    numero_valori_sotto = conta_sotto_valore(root, valore)
    print(f"La profondità dell'albero è: {profondita}")
    print(f"Il numero di valori strettamente sotto {valore} è: {numero_valori_sotto}")


if __name__ == "__main__":
    main()
```

Esercizio – compito del 23/01/2020 es.2

Esercizio – Compito del 23/01/2020 es.2 (1)

"""

Si consideri una matrice M che contenga l'entità delle spese giornaliere in euro previste per il mese di febbraio

(righe = giorno del mese, colonne = tipo di spesa). Sapendo che il budget per tale mese è di 500 euro, scrivere

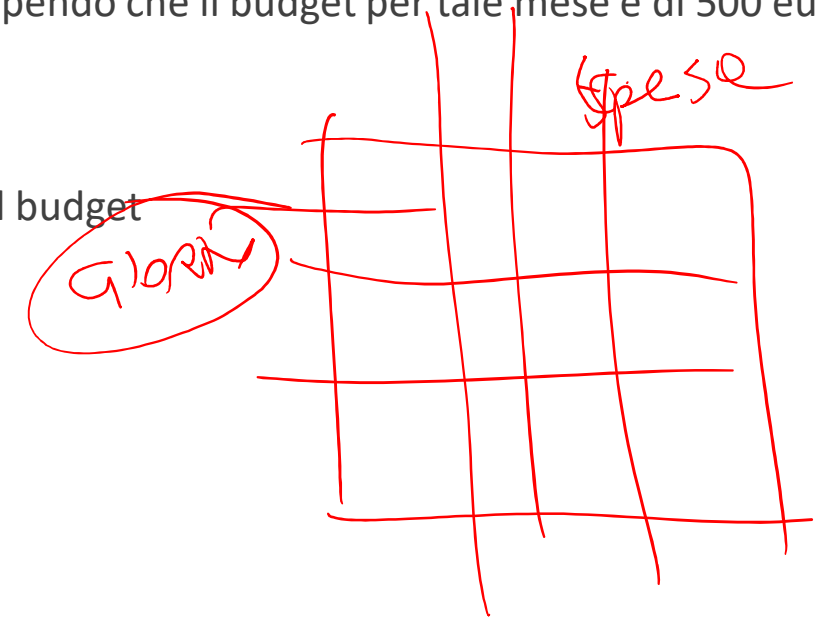
una funzione capace di:

a. Determinare se la previsione di spesa rientra nel budget

b. Stampare la spesa media giornaliera

c. Restituire il risparmio mensile

"""



Esercizio – Compito del 23/01/2020 es.2 (2)

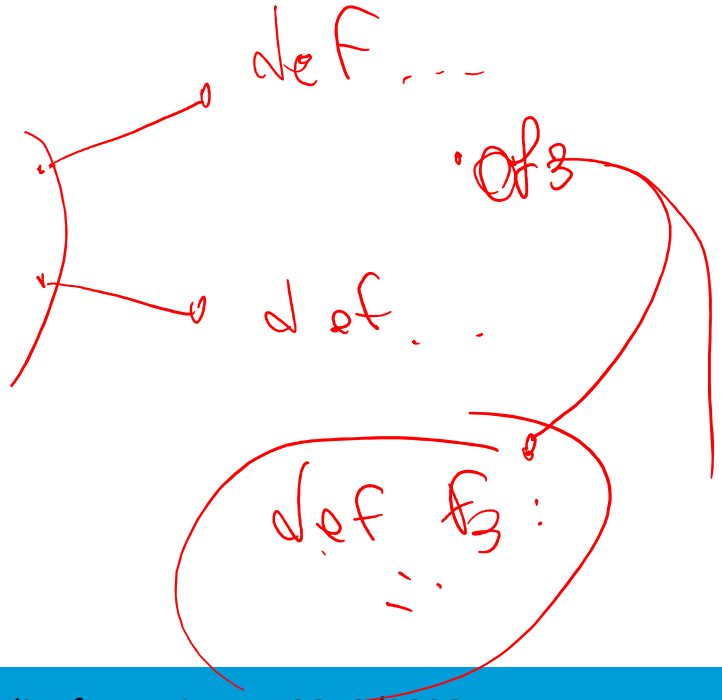
```
def calcolo_spesa_mese(spese_del_mese):
    """
    Calcola la spesa media giornaliera e verifica che la spesa totale rientri nel
    budget
    """
    totale = 0
    media_giornaliera = 0
    for spese_del_giorno in spese_del_mese:
        for spesa in spese_del_giorno:
            totale += spesa
    media_giornaliera = totale / GIORNI
    print(f"Media giornaliera: {media_giornaliera:.2f} Euro")
    risparmio = BUDGET - totale
    if totale <= BUDGET:
        return True, totale, risparmio
    else:
        return False, totale, risparmio
```

Esercizio – Compito del 23/01/2020 es.2 (3)

```
def crea_singola_spesa():
    """
    Crea una spesa con valore casuale tra 0,50 Euro e 3,99 euro.
    """
    parte_intera = random.randint(0, 3)
    parte_decimale = random.randint(0, 99)
    spesa = parte_intera + parte_decimale / 100
    if spesa < 0.50:
        return 0.50
    else:
        return spesa
```

Esercizio – Compito del 23/01/2020 es.2 (4)

```
def scegli_numero_di_spese():
    """
    Sceglie il numero di spese giornaliere in modo casuale tra 2 e 15
    """
    return random.randint(2, 15)
```



Esercitazione 23/01/2020 es.2 (5)

```
def stampa_spese(spese_del_mese):
    """
    Stampa le spese del mese scrivendo per ogni riga le spese relative al rispettivo giorno.
    Il formato di ogni riga è: 'Giorno {giorno}: ({numero_di_spese}) [ {lista_delle_spese} ]Euro'
    La {lista_delle_spese} viene formattata con 2 soli valori decimali.
    """
    print("Spese del mese:")
    giorno = 0
    for spese_del_giorno in spese_del_mese:
        giorno += 1
        numero_spese = len(spese_del_giorno)
        spese_del_giorno_str = f"({numero_spese:2d}) [ "
        for spesa in spese_del_giorno:
            spese_del_giorno_str += f"{spesa:.2f} "
        spese_del_giorno_str += "] Euro"
        print(f" Giorno {giorno:2d}: {spese_del_giorno_str}")
```

Esercitazione 23/01/2020 es.2 (6)

```
def main():
    M = list()
    for giorno in range(GIORNI):
        spese_del_giorno = list()
        numero_spese = scegli_numero_di_spese()
        for spesa in range(numero_spese):
            valore_spesa = crea_singola_spesa()
            spese_del_giorno.append(valore_spesa)
        M.append(spese_del_giorno)
    stampa_spese(M)
    print()
    budget_ok, totale, risparmio = calcolo_spesa_mese(M)
    print(f"La spesa totale del mese di febbraio è stata di {totale:.2f} Euro")
    print()
    if budget_ok:
        print(f"Il budget è stato rispettato con un risparmio di {risparmio:.2f} Euro")
    else:
        print(f"Il budget non è stato rispettato e si sono spesi {-risparmio:.2f} Euro in più")

if __name__ == "__main__":
        main()
```

CON:
import random

GIORNI = 29
BUDGET = 500

Esercizio – compito del 23/01/2020 es.3

Esercizio – Compito del 23/01/2020 es.3 (1)

"""

Dati due alberi binari di ricerca, creare una funzione per determinare:

a. Quale dei due alberi ha il valor medio maggiore

b. Quale dei due alberi ha il maggior numero di elementi

c. Inserire nel primo albero anche gli elementi del secondo albero

"""

```
def conta_elementi(root):
    conto = 0
    if root is not None:
        elementi_sx = conta_elementi(root.left)
        elementi_dx =
        conta_elementi(root.right)
        conto = 1 + elementi_sx + elementi_dx
    return conto
```

Esercizio – Compito del 23/01/2020 es.3 (2)

```
def calcola_totale(root):
   totale = 0
   if root is not None:
     totale_sx = calcola_totale(root.left)
     totale_dx = calcola_totale(root.right)
     totale = root.data + totale_sx + totale_dx
   return totale
```

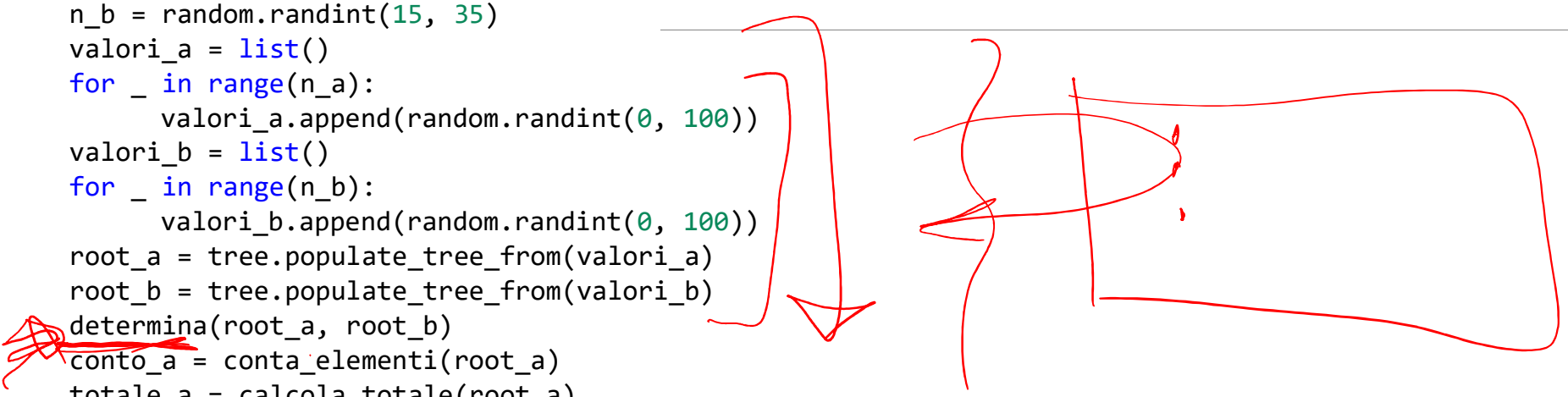
```
  def unisci_alberi(root_a, root_b):
      if root_b is not None:
          unisci_alberi(root_a, root_b.left)
          tree.insert(root_a, root_b.data)
          unisci_alberi(root_a, root_b.right)
```

Esercizio - 23/01/2020 es.3 (3)

```
def determina(root_a, root_b):
    conto_a = conta_elementi(root_a)
    conto_b = conta_elementi(root_b)
    totale_a = calcola_totale(root_a)
    totale_b = calcola_totale(root_b)
    media_a = totale_a / conto_a
    media_b = totale_b / conto_b
    if media_a > media_b:
        print(f"La media dei valori del primo albero ({media_a:.2f}) è maggiore di quella del secondo
              ({media_b:.2f})")
    elif media_a < media_b:
        print(f"La media dei valori del secondo albero ({media_b:.2f}) è maggiore di quella del primo
              ({media_a:.2f})")
    else:
        print("I due alberi hanno lo stesso valor medio ({media_a:.2f})")
    if conto_a > conto_b:
        print(f"Il numero di elementi del primo albero ({conto_a}) è maggiore di quello del secondo
              ({conto_b})")
    elif conto_a < conto_b:
        print(f"Il numero di elementi del secondo albero ({conto_b}) è maggiore di quello del primo
              ({conto_a})")
    else:
        print("I due alberi hanno lo stesso numero di elementi ({conto_a})")
    unisci_alberi(root_a, root_b)
```

Esercizio 23/01/2020 es.3 (4)

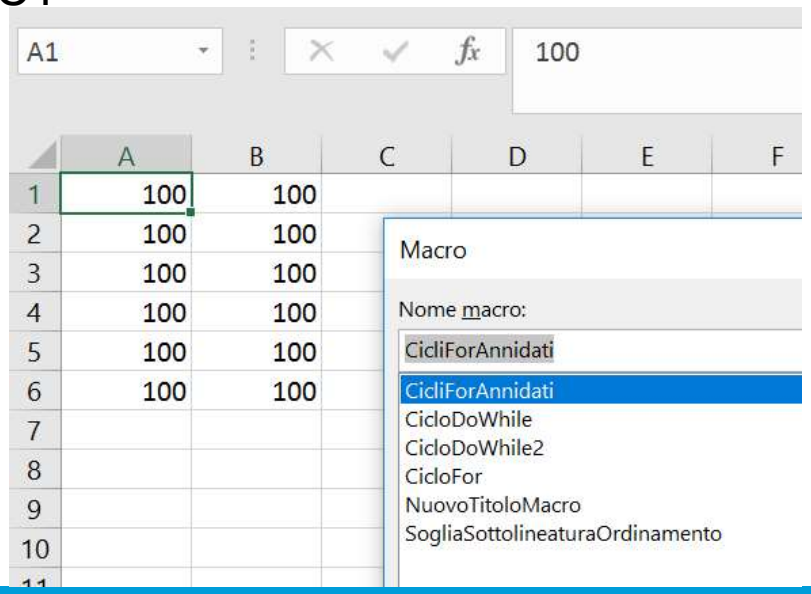
```
[…]
def main():
    n_a = random.randint(15, 35)
    n_b = random.randint(15, 35)
    valori_a = list()
    for _ in range(n_a):
        valori_a.append(random.randint(0, 100))
    valori_b = list()
    for _ in range(n_b):
        valori_b.append(random.randint(0, 100))
    root_a = tree.populate_tree_from(valori_a)
    root_b = tree.populate_tree_from(valori_b)
    determina(root_a, root_b)
    conto_a = conta_elementi(root_a)
    totale_a = calcola_totale(root_a)
    media_a = totale_a / conto_a
    print()
    print("Dopo l'inserimento dei valori del secondo albero nel primo adesso il primo albero ha le seguenti proprietà:")
    print(f" * Numero di elementi: {conto_a}")
    print(f" * Somma totale dei valori: {totale_a}")
    print(f" * Media dei valori: {media_a:.2f}")

if __name__ == "__main__":
        main()
```



Scrivere una macro (2)

- Esempio Cicli for annidati:

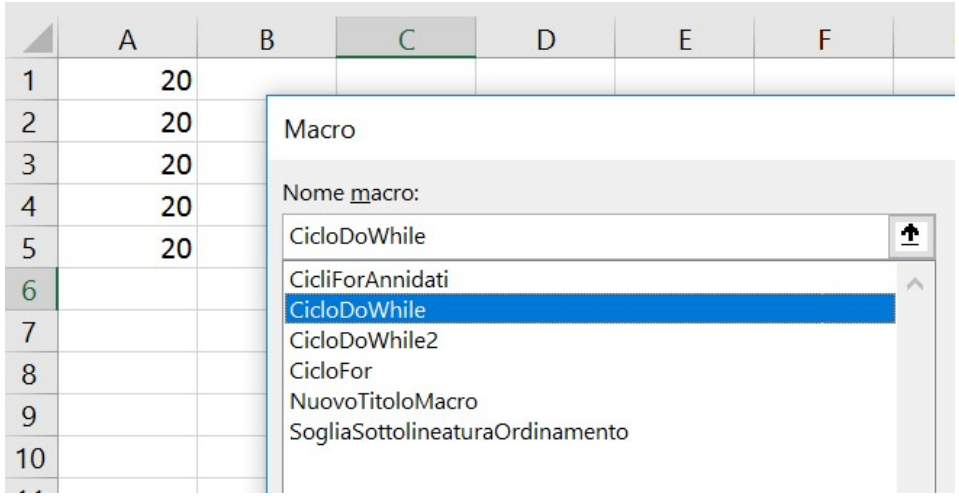
```
Sub CicliForAnnidati()
' Esempio di cicli for annidati
' Cells(x,y) --> Excel VBA considera la
' riga 1 e colonna 1
' esempio: Cells(1,3) è la cella C1

Dim i As Integer, j As Integer

For i = 1 To 6
    For j = 1 To 2
        Cells(i, j).Value = 100
    Next j
Next i
End Sub
```



Scrivere una macro (3)

- Esempio Ciclo Do While:

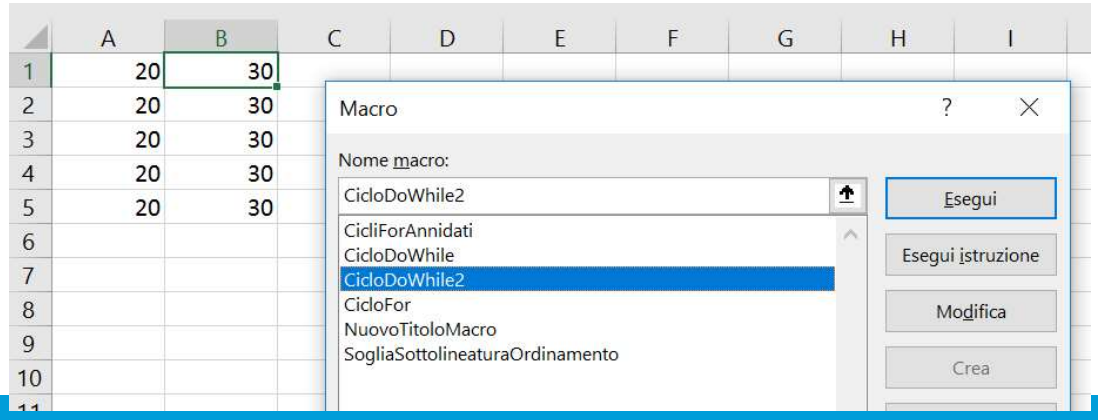
```
Sub CicloDoWhile()
' Esempio di cicli for annidati
' Cells(x,y) --> Excel VBA considera la
' riga 1 e colonna 1
' esempio: Cells(1,3) è la cella C1
Dim i As Integer
i = 1

Do While i < 6
    Cells(i, 1).Value = 20
    i = i + 1
Loop
End Sub
```



Scrivere una macro (4)

- Esempio Ciclo Do While2 (applicato dopo il precedente ciclo):

```
Sub CicloDoWhile2()
' Esempio di cicli for annidati
Dim i As Integer
i = 1

Do While Cells(i, 1).Value <> ""
    Cells(i, 2).Value = Cells(i, 1).Value + 10
    i = i + 1
Loop
End Sub
```



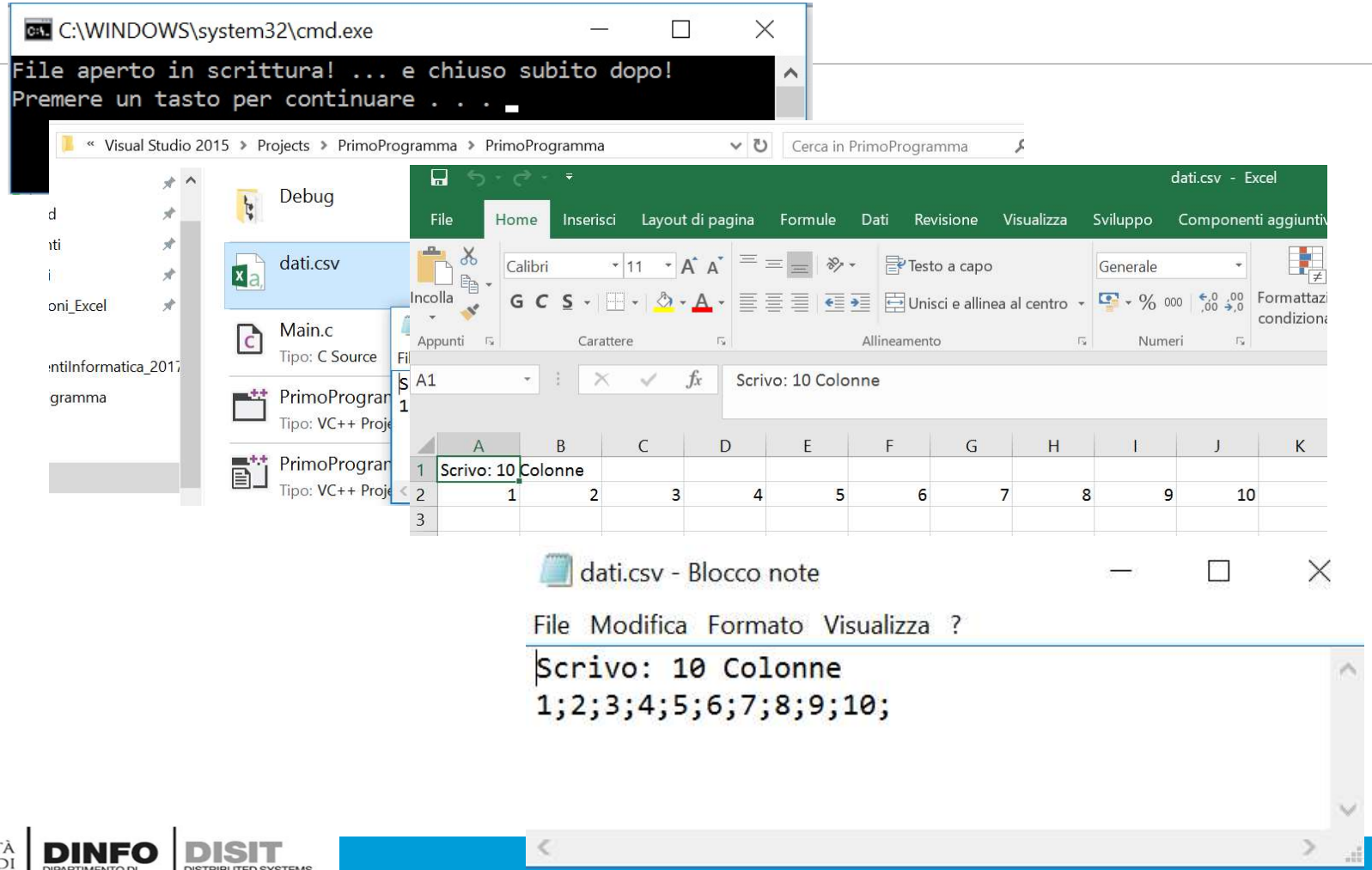
Dal C a excel: scrivere un file da C e aprirlo da excel

- Si usano le funzioni viste in precedenza per la lettura e scrittura file da C
- Si scrivono i dati di output del programmo in file csv
- I file csv contengono dati:
  - **In una riga separati da “;” o “,”**
  - **In una colonna separati dal carattere di “a capo”, ovvero in C nelle stringhe si inserisce il “\n”**

Esempio 1

```
#include <stdio.h>
main() {
    int i, num = 10;
    FILE *hand;
    float x, y;
    hand = fopen("dati.csv", "w+");//sovrascrivo il file se esiste
    if (!hand)
    printf("Impossibile aprire il file");
    else {
      fprintf(hand, "Scrivo: %d Colonne\n", num);
      for (i = 1; i <= num; i++) {
          fprintf(hand, "%d;", i); //1 riga con num (10) colonne
      }
    fclose(hand);
    printf("File aperto in scrittura! ... e chiuso subito dopo!\n");
    }
}
```

Esempio 1 – File csv

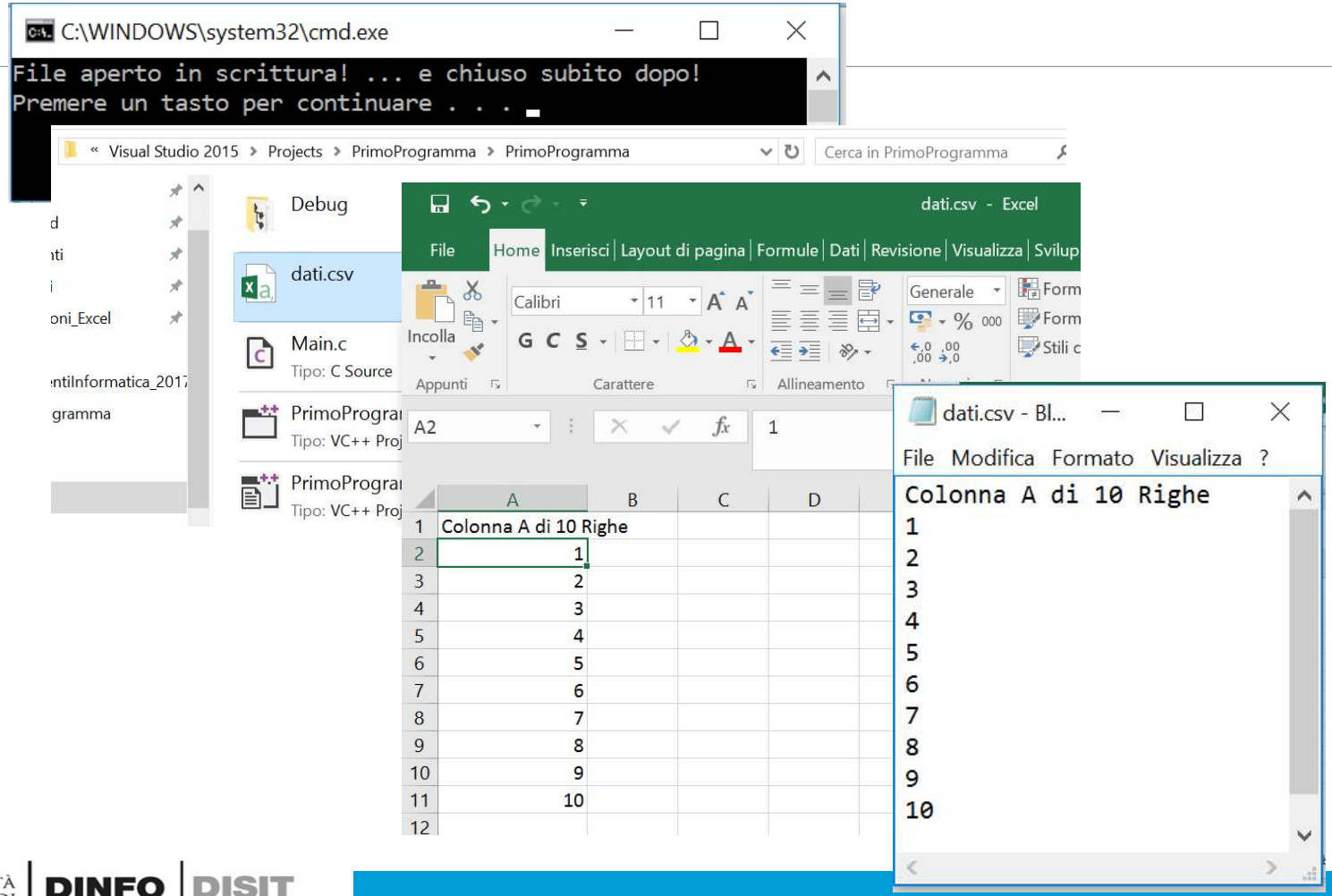


489

Esempio 2

```
#include <stdio.h>
main() {
    int i, num = 10;
    FILE *hand;
    float x, y;
    hand = fopen("dati.csv", "w+");//sovrascrivo il file se esiste
    if (!hand)
    printf("Impossibile aprire il file");
    else {
      fprintf(hand, "Colonna A\n di %d Righe\n", num); //header
      for (i = 1; i <= num; i++) {
          fprintf(hand, "%d\n", i); //1 Colonna e num (10) righe
      }
      fclose(hand);
      printf("File aperto in scrittura! ... e c
    hiuso subito dopo!\n");
    }
}
```

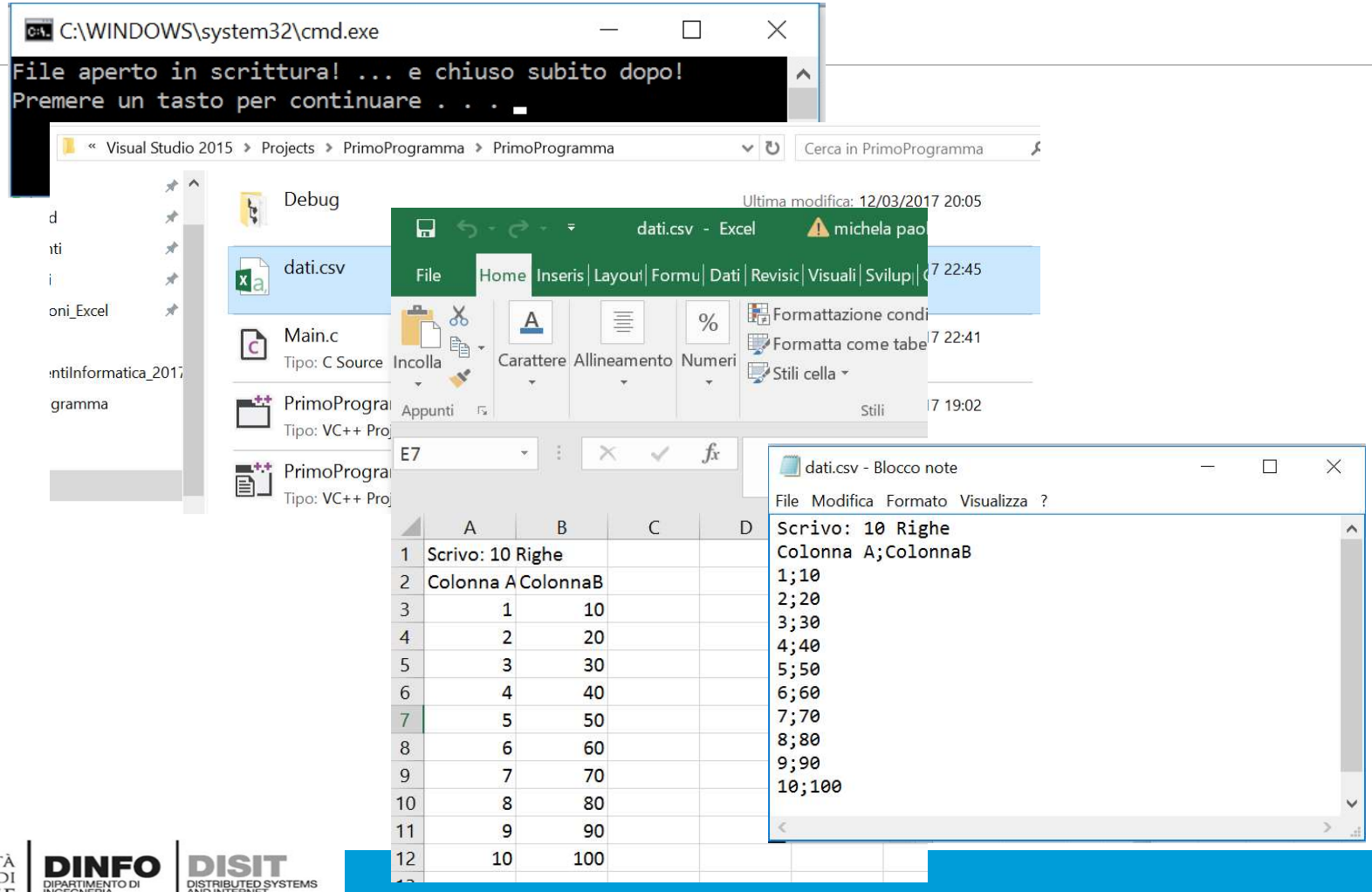
Esempio 2 – file csv



Esempio 3

```
main() {
int i, num = 10;
FILE *hand;
float x, y;
hand = fopen("dati.csv", "w+");//sovrascrivo il file se esiste
if (!hand)
printf("Impossibile aprire il file");
else {
    fprintf(hand, "Scrivo: %d Righe\n", num);
    fprintf(hand, "Colonna A;ColonnaB\n"); //header
    int sum = 0;
    for (i = 1; i <= num; i++) {
        sum = sum + 10;
        fprintf(hand, "%d;%d\n", i, sum);//2 colonne con num (10) righe
    }
fclose(hand);
printf("File aperto in scrittura! ... e chiuso subito dopo!\n");
}
}
```

Esempio 3 – file csv



	A	B	C	D
1	Scrivo: 10 Righe			
2	Colonna A	ColonnaB		
3	1	10		
4	2	20		
5	3	30		
6	4	40		
7	5	50		
8	6	60		
9	7	70		
10	8	80		
11	9	90		
12	10	100		

493

Correzione Compiti

Compito del 18/06/2020

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 18/06/2020

Esercizi:

a) (8 punti) Data una matrice A di float NxM, con M>N, scrivere una funzione che ritagli e restituisca una matrice quadrata Q NxN contenente l'elemento A_{00} e stampi il numero di colonne escluse e gli elementi esclusi colonna per colonna.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad Q = \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}$$

Stampa a schermo i seguenti dati

N_col = 1, 3, 6

b) (6 punti) Scrivere una funzione che prenda in ingresso una matrice A e renda una lista(*) L contenente la somma dei quadrati degli elementi di ogni riga della matrice

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad L = [14, 77]$$

$$L[0] = \sum_{i=0}^{M} A_{0i}^2$$

c) (6 punti) Scrivere una funzione che prenda in ingresso una lista L e la modifichi aggiungendo degli elementi con numeri random fino ad arrivare ad una dimensione di N (passata in input alla funzione).

L = [3,5], N=6, L' = [3,5,1,-2.1,3,0.3]

d) (10 punti) Scrivere una funzione che dati due alberi binari di ricerca b1 e b2 restituisca una lista (*) con gli elementi comuni a b1 e b2

(*) in C, si intende lista in forma collegata con puntatori
In Python si intendono Liste standard Python

Compito del 18/06/2020: Python

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 18/06/2020

Esercizi:

a) (8 punti) Data una matrice A di float NxM, con M>N, scrivere una funzione che ritagli e restituisca una matrice quadrata Q NxN contenente l'elemento A_{00} e stampi il numero di colonne escluse e gli elementi esclusi colonna per colonna.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad Q = \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}$$

Stampa a schermo i seguenti dati
N_col = 1, 3, 6

b) (6 punti) Scrivere una funzione che prenda in ingresso una matrice A e renda una lista(*) L contenente la somma dei quadrati degli elementi di ogni riga della matrice

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad L = [14, 77]$$

$$L[0] = \sum_{i=0}^{M} A_{0i}^2$$

c) (6 punti) Scrivere una funzione che prenda in ingresso una lista L e la modifichi aggiungendo degli elementi con numeri random fino ad arrivare ad una dimensione di N (passata in input alla funzione).

L = [3,5], N=6, L' = [3,5,1,-2.1,3,0.3]

d) (10 punti) Scrivere una funzione che dati due alberi binari di ricerca b1 e b2 restituisca una lista (*) con gli elementi comuni a b1 e b2

(*) in C, si intende lista in forma collegata con puntatori
In Python si intendono Liste standard Python

Esercizio a)

- La funzione crop_square prende in ingresso la Matrice NxM
- Calcola `removed_cols`, il numero di colonne da rimuovere, come differenza tra il numero di colonne e il numero di righe
- Effettua un ciclo sulle righe:
  - Finchè l'indice della Colonna è minore uguale al numero delle righe della matrice, metto gli elementi in Q
  - Altrimenti li stampo
- Stampa il numero di colonne rimosse
- Restituisce la matrice Q

```
def crop_square(m):
    Q=[]
    removed_cols = len(m[0]) - len(m)
    print('Colonne rimosse:', removed_cols)
    print('Elementi rimossi:')
    for row in m:
        Q.append(row[:len(m)])
        print(row[len(m):])
    return Q
```

Compito del 18/06/2020: Python

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 18/06/2020

Esercizi:

a) (8 punti) Data una matrice A di float NxM, con M>N, scrivere una funzione che ritagli e restituisca una matrice quadrata Q NxN contenente l'elemento A_{00} e stampi il numero di colonne escluse e gli elementi esclusi colonna per colonna.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad Q = \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}$$

Stampa a schermo i seguenti dati
N_col = 1, 3, 6

b) (6 punti) Scrivere una funzione che prenda in ingresso una matrice A e renda una lista(*) L contenente la somma dei quadrati degli elementi di ogni riga della matrice

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad L = [14, 77]$$

$$L[0] = \sum_{i=0}^{M} A_{0i}^2$$

c) (6 punti) Scrivere una funzione che prenda in ingresso una lista L e la modifichi aggiungendo degli elementi con numeri random fino ad arrivare ad una dimensione di N (passata in input alla funzione).

L = [3,5], N=6, L' = [3,5,1,-2.1,3,0.3]

d) (10 punti) Scrivere una funzione che dati due alberi binari di ricerca b1 e b2 restituisca una lista (*) con gli elementi comuni a b1 e b2

(*) in C, si intende lista in forma collegata con puntatori
In Python si intendono Liste standard Python

Esercizio b)

- La funzione `sum_square_rows`, prende in ingresso la matrice m
- Inizializza una lista sq e un valore sum
- Visita la matrice con due cicli:
  - Ciclo esterno sulle righe
  - Ciclo interno sulle colonne in cui effettua la somma dei quadrati aggironando la variabile sum
  - Prima di rientrare in una nuova riga, inserisce sum nella lista sq
- Rende in output la lista dei quadrati sq

```
def sum_square_rows(m):
    sq = []
    sum =0
    for i in range(len(m)):
        for el in m[i]:
            sum += el*el
        sq.append(sum)
    return sq
```

Compito del 18/06/2020: Python

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 18/06/2020

Esercizi:

a) (8 punti) Data una matrice A di float NxM, con M>N, scrivere una funzione che ritagli e restituisca una matrice quadrata Q NxN contenente l'elemento A_{00} e stampi il numero di colonne escluse e gli elementi esclusi colonna per colonna.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad Q = \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}$$

Stampa a schermo i seguenti dati

N_col = 1, 3, 6

b) (6 punti) Scrivere una funzione che prenda in ingresso una matrice A e renda una lista(*) L contenente la somma dei quadrati degli elementi di ogni riga della matrice

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad L = [14, 77]$$

$$L[0] = \sum_{i=0}^{M} A_{0i}^2$$

c) (6 punti) Scrivere una funzione che prenda in ingresso una lista L e la modifichi aggiungendo degli elementi con numeri random fino ad arrivare ad una dimensione di N (passata in input alla funzione).

L = [3,5], N=6, L' = [3,5,1,-2.1,3,0.3]

d) (10 punti) Scrivere una funzione che dati due alberi binari di ricerca b1 e b2 restituisca una lista (*) con gli elementi comuni a b1 e b2

(*) in C, si intende lista in forma collegata con puntatori
In Python si intendono Liste standard Python

Esercizio c)

- Si importa la libreria random
- La funzione `enlarge`, prende in ingresso una lista e il numero di colonne da aggiungere
- Effettua un ciclo per aggiungere le colonne mancanti usando la funzione random

```
import random
def enlarge(l,N):
    for k in range(N-len(l)):
        l.append(random.random())
```

Compito del 18/06/2020: Python

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 18/06/2020

Esercizi:

a) (8 punti) Data una matrice A di float NxM, con M>N, scrivere una funzione che ritagli e restituisca una matrice quadrata Q NxN contenente l'elemento A_{00} e stampi il numero di colonne escluse e gli elementi esclusi colonna per colonna.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad Q = \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}$$

Stampa a schermo i seguenti dati

N_col = 1, 3, 6

b) (6 punti) Scrivere una funzione che prenda in ingresso una matrice A e renda una lista(*) L contenente la somma dei quadrati degli elementi di ogni riga della matrice

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad L = [14, 77]$$

$$L[0] = \sum_{i=0}^{M} A_{0i}^2$$

c) (6 punti) Scrivere una funzione che prenda in ingresso una lista L e la modifichi aggiungendo degli elementi con numeri random fino ad arrivare ad una dimensione di N (passata in input alla funzione).

L = [3,5], N=6, L' = [3,5,1,-2.1,3,0.3]

d) (10 punti) Scrivere una funzione che dati due alberi binari di ricerca b1 e b2 restituisca una lista (*) con gli elementi comuni a b1 e b2

(*) in C, si intende lista in forma collegata con puntatori
In Python si intendono Liste standard Python

Esercizio c)

- Si importa la libreria random
- La funzione `enlarge`, prende in ingresso una lista e il numero di colonne da aggiungere
- Effettua un ciclo per aggiungere le colonne mancanti usando la funzione random

```
import random
def enlarge(l,N):
    for k in range(N-len(l)):
        l.append(random.random())
```

Compito del 18/06/2020: Python

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 18/06/2020

Esercizi:

a) (8 punti) Data una matrice A di float NxM, con M>N, scrivere una funzione che ritagli e restituisca una matrice quadrata Q NxN contenente l'elemento A_{00} e stampi il numero di colonne escluse e gli elementi esclusi colonna per colonna.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad Q = \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}$$

Stampa a schermo i seguenti dati
N_col = 1, 3, 6

b) (6 punti) Scrivere una funzione che prenda in ingresso una matrice A e renda una lista(*) L contenente la somma dei quadrati degli elementi di ogni riga della matrice

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad L = [14, 77]$$

$$L[0] = \sum_{i=0}^{M} A_{0i}^2$$

c) (6 punti) Scrivere una funzione che prenda in ingresso una lista L e la modifichi aggiungendo degli elementi con numeri random fino ad arrivare ad una dimensione di N (passata in input alla funzione).

L = [3,5], N=6, L' = [3,5,1,-2.1,3,0.3]

d) (10 punti) Scrivere una funzione che dati due alberi binari di ricerca b1 e b2 restituisca una lista (*) con gli elementi comuni a b1 e b2

(*) in C, si intende lista in forma collegata con puntatori
In Python si intendono Liste standard Python

Esercizio d)

- Import del modulo tree con classe e funzioni viste a lezione
- La funzione `tree_intersection`, prende in ingresso due alberi binari
- Inizializza la lista l
- Se un elemento del primo albero è presente anche nel secondo albero, lo aggiunge alla lista
- Restituisce la lista l

```
import tree

def tree_intersection(n, t2, l):
    if n:
        result, _ = tree.search(t2, n.data)
        if result:
            l.append(n.data)
        tree_intersection(n.left, t2,l)
        tree_intersection(n.right, t2,l)
    return l
```

Compito del 18/06/2020: C

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 18/06/2020

Esercizi:

a) (8 punti) Data una matrice A di float NxM, con M>N, scrivere una funzione che ritagli e restituisca una matrice quadrata Q NxN contenente l'elemento A_{00} e stampi il numero di colonne escluse e gli elementi esclusi colonna per colonna.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad Q = \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}$$

Stampa a schermo i seguenti dati
N_col = 1, 3, 6

b) (6 punti) Scrivere una funzione che prenda in ingresso una matrice A e renda una lista(*) L contenente la somma dei quadrati degli elementi di ogni riga della matrice

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad L = [14, 77]$$

$$L[0] = \sum_{i=0}^{M} A_{0i}^2$$

c) (6 punti) Scrivere una funzione che prenda in ingresso una lista L e la modifichi aggiungendo degli elementi con numeri random fino ad arrivare ad una dimensione di N (passata in input alla funzione).

L = [3,5], N=6, L' = [3,5,1,-2.1,3,0.3]

d) (10 punti) Scrivere una funzione che dati due alberi binari di ricerca b1 e b2 restituisca una lista (*) con gli elementi comuni a b1 e b2

(*) in C, si intende lista in forma collegata con puntatori
In Python si intendono Liste standard Python

Esercizio a)

```c
int* cut(int *A, int R, int C) {
    int remove = C - R;
    printf("Escludo %d colonne: \n", remove);
    int *Ar = (int *)malloc((R) * sizeof(int));
    for (int i = 0; i < R; i++) {//ciclo sulle righe
        //ciclo sulle colonne
        for (int j = 0; j < C; j++) {
            if (j<R)
                Ar[i*R + j] = A[i*C + j];
            else
                printf("Escludo: %d ", A[C * i + j]);
        }
     }
     return Ar;
}
```

Compito del 18/06/2020: C

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 18/06/2020

Esercizi:

a) (8 punti) Data una matrice A di float NxM, con M>N, scrivere una funzione che ritagli e restituisca una matrice quadrata Q NxN contenente l'elemento A_{00} e stampi il numero di colonne escluse e gli elementi esclusi colonna per colonna.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad Q = \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}$$

Stampa a schermo i seguenti dati

N_col = 1, 3, 6

b) (6 punti) Scrivere una funzione che prenda in ingresso una matrice A e renda una lista(*) L contenente la somma dei quadrati degli elementi di ogni riga della matrice

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad L = [14, 77]$$

$$L[0] = \sum_{i=0}^{M} A_{0i}^2$$

c) (6 punti) Scrivere una funzione che prenda in ingresso una lista L e la modifichi aggiungendo degli elementi con numeri random fino ad arrivare ad una dimensione di N (passata in input alla funzione).

L = [3,5], N=6, L' = [3,5,1,-2.1,3,0.3]

d) (10 punti) Scrivere una funzione che dati due alberi binari di ricerca b1 e b2 restituisca una lista (*) con gli elementi comuni a b1 e b2

(*) in C, si intende lista in forma collegata con puntatori
In Python si intendono Liste standard Python

Esercizio b)

```
void square(int *A, int R, int C, struct list ** ptrptr) {
        int sum = 0;
        for (int i = 0; i < R; i++) {//ciclo sulle righe
                sum =  0;
                for (int j = 0; j < C; j++) {//ciclo sulle colonne
                        sum += A[i*C + j] * A[i*C + j];
                }
                suf_insert(ptrptr, sum);
        }
}
```

Compito del 18/06/2020: C

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 18/06/2020

Esercizi:

a) (8 punti) Data una matrice A di float NxM, con M>N, scrivere una funzione che ritagli e restituisca una matrice quadrata Q NxN contenente l'elemento A_{00} e stampi il numero di colonne escluse e gli elementi esclusi colonna per colonna.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad Q = \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}$$

Stampa a schermo i seguenti dati

N_col = 1, 3, 6

b) (6 punti) Scrivere una funzione che prenda in ingresso una matrice A e renda una lista(*) L contenente la somma dei quadrati degli elementi di ogni riga della matrice

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad L = [14, 77]$$

$$L[0] = \sum_{i=0}^{M} A_{0i}^2$$

c) (6 punti) Scrivere una funzione che prenda in ingresso una lista L e la modifichi aggiungendo degli elementi con numeri random fino ad arrivare ad una dimensione di N (passata in input alla funzione).

L = [3,5], N=6, L' = [3,5,1,-2.1,3,0.3]

d) (10 punti) Scrivere una funzione che dati due alberi binari di ricerca b1 e b2 restituisca una lista (*) con gli elementi comuni a b1 e b2

(*) in C, si intende lista in forma collegata con puntatori
In Python si intendono Liste standard Python

Esercizio c)

```
void add_random(int max, struct list * ptrptr) {
      int count = cont_elements(ptrptr);
      int add = max - count;
      for (int i = 0; i < add; i++) {
            suf_insert(&ptrptr, rand());
      }
}
```

Compito del 18/06/2020: C

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 18/06/2020

Esercizi:

a) (8 punti) Data una matrice A di float NxM, con M>N, scrivere una funzione che ritagli e restituisca una matrice quadrata Q NxN contenente l'elemento A_{00} e stampi il numero di colonne escluse e gli elementi esclusi colonna per colonna.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \; Q = \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix}$$

Stampa a schermo i seguenti dati

N_col = 1, 3, 6

b) (6 punti) Scrivere una funzione che prenda in ingresso una matrice A e renda una lista(*) L contenente la somma dei quadrati degli elementi di ogni riga della matrice

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \; L = [14, 77]$$

$$L[0] = \sum_{i=0}^{M} A_{0i}^2$$

c) (6 punti) Scrivere una funzione che prenda in ingresso una lista L e la modifichi aggiungendo degli elementi con numeri random fino ad arrivare ad una dimensione di N (passata in input alla funzione).

L = [3,5], N=6, L' = [3,5,1,-2.1,3,0.3]

d) (10 punti) Scrivere una funzione che dati due alberi binari di ricerca b1 e b2 restituisca una lista (*) con gli elementi comuni a b1 e b2

(*) in C, si intende lista in forma collegata con puntatori
In Python si intendono Liste standard Python

Esercizio d)

```
Esercizio d)
void common(struct btree * ptr1, struct btree *
            ptr2, struct list **ptrptr) {
      Boolean found = FALSE;
      if (ptr1 != NULL) {
            common(ptr1->left_ptr, ptr2, ptrptr);
            found = search(ptr2, ptr1->value);
            if (found)
                  suf_insert(ptrptr, ptr1->value);
            common(ptr1->right_ptr, ptr2, ptrptr);
      }
}
```

Compito del 09/07/2020: Python

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 09/07/2020

Esercizi:

a) (4 punti) Data una matrice MxN di float, scrivere una funzione che restituisca la coppia di indici i, j dell'elemento con il massimo valore.

b) (4 punti) Data una matrice MxN di float, scrivere una funzione che crei due liste una con gli indici dei massimi di ciascuna riga e una con gli indici dei massimi di ciascuna colonna

c) (6 punti) Definire una classe/struct per modellare un carrello acquisti di uno shop online. Definire una classe/struct per modellare un item in vendita con ID e prezzo. Il carrello deve contenere una lista di oggetti/struct item, un ID e l'ID dell'utente.

d) (8 punti) Scrivere un metodo della classe carrello/una funzione per calcolare il totale del costo degli oggetti nel carrello

e) (8 punti) Scrivere un metodo della classe carrello/una funzione per aggiungere un item al carrello

Esercizio a)

```
def indexes_max(m):
    max=0
    r_max=0
    c_max=0
    for i in range(len(m)):
        count = 0
        for el in m[i]:
            if el>max:
                max = el
                r_max = count
                c_max = i
            count+=1
    return r_max, c_max
```

Compito del 09/07/2020: Python

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 09/07/2020

Esercizi:

a) (4 punti) Data una matrice MxN di float, scrivere una funzione che restituisca la coppia di indici i, j dell'elemento con il massimo valore.

b) (4 punti) Data una matrice MxN di float, scrivere una funzione che crei due liste una con gli indici dei massimi di ciascuna riga e una con gli indici dei massimi di ciascuna colonna

c) (6 punti) Definire una classe/struct per modellare un carrello acquisti di uno shop online. Definire una classe/struct per modellare un item in vendita con ID e prezzo. Il carrello deve contenere una lista di oggetti/struct item, un ID e l'ID dell'utente.

d) (8 punti) Scrivere un metodo della classe carrello/una funzione per calcolare il totale del costo degli oggetti nel carrello

e) (8 punti) Scrivere un metodo della classe carrello/una funzione per aggiungere un item al carrello

Esercizio b)

```python
def indexes_max_r_c(m):
    max_col=0
    L_row=[]
    L_col=[]
    for i in range(len(m)):
        max_riga=0
        count = 0
        for el in m[i]:
            if(el>max_riga):
                max_riga = el
                r_max = i
                c_max = count
            count+=1
        L_row.append([r_max, c_max])
    r_max=0
    c_max=0

    for j in range(len(m[0])):
        max_col=0
        for i in range(len(m)):
            if(max_col < m[i][j] ):
                max_col = m[i][j]
                r_max = i
                c_max = j
        L_col.append([r_max, c_max])

    return L_row, L_col
```

Compito del 09/07/2020: Python

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 09/07/2020

Esercizi:

a) (4 punti) Data una matrice MxN di float, scrivere una funzione che restituisca la coppia di indici i, j dell'elemento con il massimo valore.

b) (4 punti) Data una matrice MxN di float, scrivere una funzione che crei due liste una con gli indici dei massimi di ciascuna riga e una con gli indici dei massimi di ciascuna colonna

c) (6 punti) Definire una classe/struct per modellare un carrello acquisti di uno shop online. Definire una classe/struct per modellare un item in vendita con ID e prezzo. Il carrello deve contenere una lista di oggetti/struct item, un ID e l'ID dell'utente.

d) (8 punti) Scrivere un metodo della classe carrello/una funzione per calcolare il totale del costo degli oggetti nel carrello

e) (8 punti) Scrivere un metodo della classe carrello/una funzione per aggiungere un item al carrello

Esercizio c)

```python
class acquisto:
    def __init__(self, id=None, prezzo=None):
        self.id = id
        self.prezzo = prezzo

    def __str__(self):
        return f'identificativo: {self.id} | prezzo: {self.prezzo} euro'

    def __repr__(self):
        return f'{self.id}{self.prezzo}'

class carrello:
    def __init__(self, user=None, acquisto=None):
        self.acquisti = []
        self.acquisti.append(acquisto)
        self.user = user

    def __str__(self):
        return f'{self.user}'

    def __repr__(self):
        return f'{self.user}'

    def stampa_carrello(carrello):
        for i in carrello.acquisti:
            print(i)
```

Compito del 09/07/2020: Python

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 09/07/2020

Esercizi:

a) (4 punti) Data una matrice MxN di float, scrivere una funzione che restituisca la coppia di indici i, j dell'elemento con il massimo valore.

b) (4 punti) Data una matrice MxN di float, scrivere una funzione che crei due liste una con gli indici dei massimi di ciascuna riga e una con gli indici dei massimi di ciascuna colonna

c) (6 punti) Definire una classe/struct per modellare un carrello acquisti di uno shop online. Definire una classe/struct per modellare un item in vendita con ID e prezzo. Il carrello deve contenere una lista di oggetti/struct item, un ID e l'ID dell'utente.

d) (8 punti) Scrivere un metodo della classe carrello/una funzione per calcolare il totale del costo degli oggetti nel carrello

e) (8 punti) Scrivere un metodo della classe carrello/una funzione per aggiungere un item al carrello

Esercizio c)

```python
class acquisto:
    def __init__(self, id=None, prezzo=None):
        self.id = id
        self.prezzo = prezzo

    def __str__(self):
        return f'identificativo: {self.id} | prezzo: {self.prezzo} euro'

    def __repr__(self):
        return f'{self.id}{self.prezzo}'

class carrello:
    def __init__(self, user=None, acquisto=None):
        self.acquisti = []
        self.acquisti.append(acquisto)
        self.user = user

    def __str__(self):
        return f'{self.user}'

    def __repr__(self):
        return f'{self.user}'

    def stampa_carrello(carrello):
        for i in carrello.acquisti:
            print(i)
```

Compito del 09/07/2020: Python

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 09/07/2020

Esercizi:

a) (4 punti) Data una matrice MxN di float, scrivere una funzione che restituisca la coppia di indici i, j dell'elemento con il massimo valore.

b) (4 punti) Data una matrice MxN di float, scrivere una funzione che crei due liste una con gli indici dei massimi di ciascuna riga e una con gli indici dei massimi di ciascuna colonna

c) (6 punti) Definire una classe/struct per modellare un carrello acquisti di uno shop online. Definire una classe/struct per modellare un item in vendita con ID e prezzo. Il carrello deve contenere una lista di oggetti/struct item, un ID e l'ID dell'utente.

d) (8 punti) Scrivere un metodo della classe carrello/una funzione per calcolare il totale del costo degli oggetti nel carrello

e) (8 punti) Scrivere un metodo della classe carrello/una funzione per aggiungere un item al carrello

Esercizio d) ed e)

```python
def calcola_totale(carrello):
    tot = 0
    for i in carrello.acquisti:
        tot += i.prezzo
    return tot

def add_acquisto(carrello, acquisto):
    carrello.acquisti.append(acquisto)
```

Compito del 09/07/2020: C

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 09/07/2020

Esercizi:

a) (4 punti) Data una matrice MxN di float, scrivere una funzione che restituisca la coppia di indici i, j dell'elemento con il massimo valore.

b) (4 punti) Data una matrice MxN di float, scrivere una funzione che crei due liste una con gli indici dei massimi di ciascuna riga e una con gli indici dei massimi di ciascuna colonna

c) (6 punti) Definire una classe/struct per modellare un carrello acquisti di uno shop online. Definire una classe/struct per modellare un item in vendita con ID e prezzo. Il carrello deve contenere una lista di oggetti/struct item, un ID e l'ID dell'utente.

d) (8 punti) Scrivere un metodo della classe carrello/una funzione per calcolare il totale del costo degli oggetti nel carrello

e) (8 punti) Scrivere un metodo della classe carrello/una funzione per aggiungere un item al carrello

Esercizio a)

```
void arg_max_mat(float* M, int R, int C,
                 int* i, int* j){
  *i=0;
  *j=0;
  float max_val=M[0];
  for(int ii=0; ii < R; ii++)
    for(int jj=0; jj<C; jj++){
      if (M[ii*C + jj] > max_val){
        max_val = M[ii*C + jj];
        *i = ii;
        *j = jj;
      }
  }
}
```

Compito del 09/07/2020: C

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 09/07/2020

Esercizi:

a) (4 punti) Data una matrice MxN di float, scrivere una funzione che restituisca la coppia di indici i, j dell'elemento con il massimo valore.

b) (4 punti) Data una matrice MxN di float, scrivere una funzione che crei due liste una con gli indici dei massimi di ciascuna riga e una con gli indici dei massimi di ciascuna colonna

c) (6 punti) Definire una classe/struct per modellare un carrello acquisti di uno shop online. Definire una classe/struct per modellare un item in vendita con ID e prezzo. Il carrello deve contenere una lista di oggetti/struct item, un ID e l'ID dell'utente.

d) (8 punti) Scrivere un metodo della classe carrello/una funzione per calcolare il totale del costo degli oggetti nel carrello

e) (8 punti) Scrivere un metodo della classe carrello/una funzione per aggiungere un item al carrello

Esercizio b)

```
void max_rows_cols(float* M, int R, int C, struct list**
                   max_rows, struct list** max_cols){
  //get max of every row

  for (int r=0; r < R; r++){
    float max_row=M[r*C]; //max init as first row element
    for (int c=0; c < C; c++){
      if (M[r*C + c] > max_row)
        max_row = M[r*C + c];
    }
    suf_insert(max_rows, max_row);
  }
  for (int c=0; c < C; c++){
      float max_col=M[c]; //max init as first row element
      for (int r=0; r < R; r++){
        if (M[r*C + c] > max_col)
          max_col = M[r*C + c];
      }
      suf_insert(max_cols, max_col);
    }
}
```

Compito del 09/07/2020: C

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 09/07/2020

Esercizi:

a) (4 punti) Data una matrice MxN di float, scrivere una funzione che restituisca la coppia di indici i, j dell'elemento con il massimo valore.

b) (4 punti) Data una matrice MxN di float, scrivere una funzione che crei due liste una con gli indici dei massimi di ciascuna riga e una con gli indici dei massimi di ciascuna colonna

c) (6 punti) Definire una classe/struct per modellare un carrello acquisti di uno shop online. Definire una classe/struct per modellare un item in vendita con ID e prezzo. Il carrello deve contenere una lista di oggetti/struct item, un ID e l'ID dell'utente.

d) (8 punti) Scrivere un metodo della classe carrello/una funzione per calcolare il totale del costo degli oggetti nel carrello

e) (8 punti) Scrivere un metodo della classe carrello/una funzione per aggiungere un item al carrello

Esercizio c)

```
struct carrello {
  int id_carrello;
  int user_id;
  struct item_list* items;
};
```

```
struct item_list{
  int id_item;
  struct item_list* next;
  float price;
};
```

Compito del 09/07/2020: C

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 09/07/2020

Esercizi:

a) (4 punti) Data una matrice MxN di float, scrivere una funzione che restituisca la coppia di indici i, j dell'elemento con il massimo valore.

b) (4 punti) Data una matrice MxN di float, scrivere una funzione che crei due liste una con gli indici dei massimi di ciascuna riga e una con gli indici dei massimi di ciascuna colonna

c) (6 punti) Definire una classe/struct per modellare un carrello acquisti di uno shop online. Definire una classe/struct per modellare un item in vendita con ID e prezzo. Il carrello deve contenere una lista di oggetti/struct item, un ID e l'ID dell'utente.

d) (8 punti) Scrivere un metodo della classe carrello/una funzione per calcolare il totale del costo degli oggetti nel carrello

e) (8 punti) Scrivere un metodo della classe carrello/una funzione per aggiungere un item al carrello

Esercizio d)

```
float total(struct carrello * c){
  struct item_list* items = c->items;
  float tot=0;
  while(items){
    tot+=items->price;
    items = items->next;
  }
  return tot;
}
```

Compito del 09/07/2020: C

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 09/07/2020

Esercizi:

a) (4 punti) Data una matrice MxN di float, scrivere una funzione che restituisca la coppia di indici i, j dell'elemento con il massimo valore.

b) (4 punti) Data una matrice MxN di float, scrivere una funzione che crei due liste una con gli indici dei massimi di ciascuna riga e una con gli indici dei massimi di ciascuna colonna

c) (6 punti) Definire una classe/struct per modellare un carrello acquisti di uno shop online. Definire una classe/struct per modellare un item in vendita con ID e prezzo. Il carrello deve contenere una lista di oggetti/struct item, un ID e l'ID dell'utente.

d) (8 punti) Scrivere un metodo della classe carrello/una funzione per calcolare il totale del costo degli oggetti nel carrello

e) (8 punti) Scrivere un metodo della classe carrello/una funzione per aggiungere un item al carrello

Esercizio e)

```
void pre_insert_item(struct item_list** ptrptr,
                     int id, float price){
 struct item_list* tmp = *ptrptr;
 (*ptrptr)=(struct item_list*)malloc(sizeof(struct item_list));
 (*ptrptr)->next = tmp;
 (*ptrptr)->id_item = id;
 (*ptrptr)->price = price;

}

void add_item(struct carrello* c, int id, float price){
  pre_insert_item(&(c->items), id, price);
}
```

Compito del 23/07/2020: teoria

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 23/07/2020

Esercizi:

a) (6 punti) [**Teoria**] Indicare, motivando la risposta, il numero **minimo** di bit per rappresentare una data. Si assuma che gli anni siano compresi in [0,9999].

b) (6 punti) [**Teoria**] Indicare, motivando la risposta, una sequenza di inserimento in un albero binario di ricerca dei seguenti valori interi affinché l'albero risulti bilanciato: [1,10,-2,1,12,2,3,8].

c) (6 punti) [**Codice**] Scrivere una funzione che dato un numero intero in base dieci restituisca una lista i cui elementi sono le cifre del numero, rappresentate come interi. Es. f(2034) -> [2,0,3,4]

d) (2 punti) [**Codice**] Definire una classe/struct che rappresenti una persona contenente il nome e il cognome rappresentati come stringhe.

e) (10 punti) [**Codice**] [Solo C] esteso lo struct al punto d) per rappresentare il nodo di una lista.
[Tutti] Scrivere una funzione che consenta di inserire gli oggetti/struct del punto b) in ordine alfabetico (prima per cognome e poi per nome).
Es. insert("Mario","Rossi", list); insert("Paolo","Rossi", list); insert("Mario","Neri");

Esercizio a)

Ogni data si puo' scrivere nella forma:
gg/mm/aaaa

Con: gg in [1,31]; mm in [1,12]; aaaa in [0,9999]

Quindi:

- Giorno: da 1 a 31 -> 2^5= 32, sufficiente per gg
- Mese da 1 a 12 -> 2^4 = 16, sufficiente per mm
- Anno da 0 a 9999 -> 2^14 = 16.384, sufficiente per aaaa

- Totale bit: 5+4+14 = 23

Compito del 23/07/2020: teoria

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 23/07/2020

Esercizi:

a) (6 punti) [**Teoria**] Indicare, motivando la risposta, il numero **minimo** di bit per rappresentare una data. Si assuma che gli anni siano compresi in [0,9999].

b) (6 punti) [**Teoria**] Indicare, motivando la risposta, una sequenza di inserimento in un albero binario di ricerca dei seguenti valori interi affinché l'albero risulti bilanciato: [1,10,-2,1,12,2,3,8].

c) (6 punti) [**Codice**] Scrivere una funzione che dato un numero intero in base dieci restituisca una lista i cui elementi sono le cifre del numero, rappresentate come interi. Es. f(2034) -> [2,0,3,4]

d) (2 punti) [**Codice**] Definire una classe/struct che rappresenti una persona contenente il nome e il cognome rappresentati come stringhe.

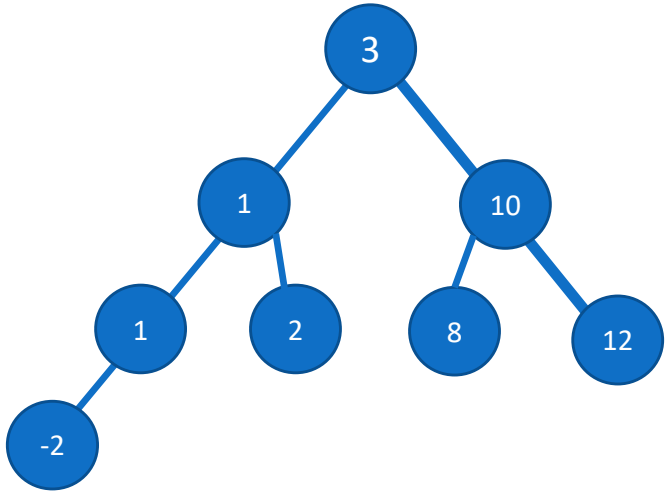
e) (10 punti) [**Codice**] [Solo C] esteso lo struct al punto d) per rappresentare il nodo di una lista.
[Tutti] Scrivere una funzione che consenta di inserire gli oggetti/struct del punto b) in ordine alfabetico (prima per cognome e poi per nome).
Es. insert("Mario","Rossi", list); insert("Paolo","Rossi", list); insert("Mario","Neri");

Esercizio b)

Possibile ordine per ottenere un albero bilanciato:

3,10,1,8,12,2,1, -2

- Un albero binario si dice 'Albero binario di ricerca' quando: il valore codificato su ciascun nodo è maggiore o uguale del valore codificato sul figlio sinistro del nodo stesso e minore secco del valore codificato sul figlio destro
- Un albero binario si dice anche bilanciato quando ogni nodo che si trova su un livello diverso dall'ultimo o dal penultimo ha due figli



In questo esempio
Entrambe le regole sono rispettate

Compito del 23/07/2020: Python

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 23/07/2020

Esercizi:

a) (6 punti) [**Teoria**] Indicare, motivando la risposta, il numero **minimo** di bit per rappresentare una data. Si assuma che gli anni siano compresi in [0,9999].

b) (6 punti) [**Teoria**] Indicare, motivando la risposta, una sequenza di inserimento in un albero binario di ricerca dei seguenti valori interi affinché l'albero risulti bilanciato: [1,10,-2,1,12,2,3,8].

c) (6 punti) [**Codice**] Scrivere una funzione che dato un numero intero in base dieci restituisca una lista i cui elementi sono le cifre del numero, rappresentate come interi. Es. f(2034) -> [2,0,3,4]

d) (2 punti) [**Codice**] Definire una classe/struct che rappresenti una persona contenente il nome e il cognome rappresentati come stringhe.

e) (10 punti) [**Codice**] [Solo C] esteso lo struct al punto d) per rappresentare il nodo di una lista.
[Tutti] Scrivere una funzione che consenta di inserire gli oggetti/struct del punto b) in ordine alfabetico (prima per cognome e poi per nome).
Es. insert("Mario","Rossi", list); insert("Paolo","Rossi", list); insert("Mario","Neri");

Esercizio c)

```
def list_digits(n):
    l=[]
    s = str(n)
    #calcolo il numero di cifre di n
    maxp = len(s)
    p=maxp-1
    while p >=0:
    #tramite la divisione per 10^p mi calcolo le cifre
        q = n//(10**p)
        #metto le cifre in una lista di appoggio
        l.append(q)
        n = n-q*(10**p)
        p-=1
        #restituisco la lista con le cifre
    return l
```

Compito del 23/07/2020: Python

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 23/07/2020

Esercizi:

a) (6 punti) [**Teoria**] Indicare, motivando la risposta, il numero **minimo** di bit per rappresentare una data. Si assuma che gli anni siano compresi in [0,9999].

b) (6 punti) [**Teoria**] Indicare, motivando la risposta, una sequenza di inserimento in un albero binario di ricerca dei seguenti valori interi affinché l'albero risulti bilanciato: [1,10,-2,1,12,2,3,8].

c) (6 punti) [**Codice**] Scrivere una funzione che dato un numero intero in base dieci restituisca una lista i cui elementi sono le cifre del numero, rappresentate come interi. Es. f(2034) -> [2,0,3,4]

d) (2 punti) [**Codice**] Definire una classe/struct che rappresenti una persona contenente il nome e il cognome rappresentati come stringhe.

e) (10 punti) [**Codice**] [Solo C] esteso lo struct al punto d) per rappresentare il nodo di una lista.
[Tutti] Scrivere una funzione che consenta di inserire gli oggetti/struct del punto b) in ordine alfabetico (prima per cognome e poi per nome).
Es. insert("Mario","Rossi", list); insert("Paolo","Rossi", list); insert("Mario","Neri");

Esercizio d) e)

```
class person:
    def __init__(self, f, l):
        self.name = f
        self.lastname = l
    def __repr__(self):
        return self.name+' '+self.lastname

def insert_ordered(p, l):#p è un oggetto Person e l è una list.
    i=0
    while i < len(l) and p.lastname > l[i].lastname:
        #avanzo secondo il cognome
        i+=1
        # se trovo un cognome uguale
    while i < len(l) and
        l[i].lastname == p.lastname and p.name > l[i].name :
    # avanzo secondo il nome
        i+=1

    #alla fine inserisco.
    l.insert(i, p)
```

Compito del 23/07/2020: C

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 23/07/2020

Esercizi:

a) (6 punti) [**Teoria**] Indicare, motivando la risposta, il numero **minimo** di bit per rappresentare una data. Si assuma che gli anni siano compresi in [0,9999].

b) (6 punti) [**Teoria**] Indicare, motivando la risposta, una sequenza di inserimento in un albero binario di ricerca dei seguenti valori interi affinché l'albero risulti bilanciato: [1,10,-2,1,12,2,3,8].

c) (6 punti) [**Codice**] Scrivere una funzione che dato un numero intero in base dieci restituisca una lista i cui elementi sono le cifre del numero, rappresentate come interi. Es. f(2034) -> [2,0,3,4]

d) (2 punti) [**Codice**] Definire una classe/struct che rappresenti una persona contenente il nome e il cognome rappresentati come stringhe.

e) (10 punti) [**Codice**] [Solo C] esteso lo struct al punto d) per rappresentare il nodo di una lista.
[Tutti] Scrivere una funzione che consenta di inserire gli oggetti/struct del punto b) in ordine alfabetico (prima per cognome e poi per nome).
Es. insert("Mario","Rossi", list); insert("Paolo","Rossi", list); insert("Mario","Neri");

Esercizio c)

```
struct list {
        int value;
        struct list * next_ptr;
};

void list_digit(struct list_char **list, int n);

void list_digit(struct list **list, int n){
        int len = 0, i = 0, cifra = n;
        while (n != 0) {//calclo il numero di cifre
                len++;
                n /= 10;
        }
        for (i = 0; i < len; i++) {//inserisco ogni cifra in lista
        //tramite divisione per dieci, visto che il numero
        // è in base 10
          cifra = cifra % 10;
          suf_insert(list, cifra);
        }
}
```

Compito del 23/07/2020: C

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 23/07/2020

Esercizi:

a) (6 punti) [**Teoria**] Indicare, motivando la risposta, il numero **minimo** di bit per rappresentare una data. Si assuma che gli anni siano compresi in [0,9999].

b) (6 punti) [**Teoria**] Indicare, motivando la risposta, una sequenza di inserimento in un albero binario di ricerca dei seguenti valori interi affinché l'albero risulti bilanciato: [1,10,-2,1,12,2,3,8].

c) (6 punti) [**Codice**] Scrivere una funzione che dato un numero intero in base dieci restituisca una lista i cui elementi sono le cifre del numero, rappresentate come interi. Es. f(2034) -> [2,0,3,4]

d) (2 punti) [**Codice**] Definire una classe/struct che rappresenti una persona contenente il nome e il cognome rappresentati come stringhe.

e) (10 punti) [**Codice**] [Solo C] esteso lo struct al punto d) per rappresentare il nodo di una lista.
[Tutti] Scrivere una funzione che consenta di inserire gli oggetti/struct del punto b) in ordine alfabetico (prima per cognome e poi per nome).
Es. insert("Mario","Rossi", list); insert("Paolo","Rossi", list); insert("Mario","Neri");

Esercizio d) e)

```c
struct person {
  char name[200];
  char lastname[200];
  struct list * next_ptr;
};
void pre_insert_person(struct person ** ptrptr, char* nome, char *cognome);
void order_insert_person(struct person ** ptrptr, char* nome, char *cognome);
void init_person(struct person ** ptrptr);

void init_person(struct person ** ptrptr) {
  *ptrptr = NULL;
}
//inserimento in testa
void pre_insert_person(struct person ** ptrptr, char* name, char
                                                    *lastname) {
  struct person** tmp_ptr;
  tmp_ptr = *ptrptr;

  *ptrptr = (struct person *)malloc(sizeof(struct person));
  strcpy((*ptrptr)->name, name);
  strcpy((*ptrptr)->lastname, lastname);
  (*ptrptr)->next_ptr = tmp_ptr;
}
```

Compito del 23/07/2020: C

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 23/07/2020

Esercizi:

a) (6 punti) [**Teoria**] Indicare, motivando la risposta, il numero **minimo** di bit per rappresentare una data. Si assuma che gli anni siano compresi in [0,9999].

b) (6 punti) [**Teoria**] Indicare, motivando la risposta, una sequenza di inserimento in un albero binario di ricerca dei seguenti valori interi affinché l'albero risulti bilanciato: [1,10,-2,1,12,2,3,8].

c) (6 punti) [**Codice**] Scrivere una funzione che dato un numero intero in base dieci restituisca una lista i cui elementi sono le cifre del numero, rappresentate come interi. Es. f(2034) -> [2,0,3,4]

d) (2 punti) [**Codice**] Definire una classe/struct che rappresenti una persona contenente il nome e il cognome rappresentati come stringhe.

e) (10 punti) [**Codice**] [Solo C] esteso lo struct al punto d) per rappresentare il nodo di una lista.
[Tutti] Scrivere una funzione che consenta di inserire gli oggetti/struct del punto b) in ordine alfabetico (prima per cognome e poi per nome).
Es. insert("Mario","Rossi", list); insert("Paolo","Rossi", list); insert("Mario","Neri");

Esercizio d) e)

```
//inserimento ordinato
void  order_insert_person(struct  person  **  ptrptr,  char*  name,  char
*lastname) {
  while (*ptrptr != NULL && strcmp((*ptrptr)->lastname, lastname) <= 0) {
    if (strcmp((*ptrptr)->lastname, lastname) == 0) {
      //se il cognome è uguale, devo guardare il nome
      if (strcmp((*ptrptr)->name, name) < 0)
        ptrptr = &((*ptrptr)->next_ptr);
      else
        pre_insert_person(ptrptr, name, lastname);
        return;
    }
    else
      ptrptr = &((*ptrptr)->next_ptr);
  }
  pre_insert_person(ptrptr, name, lastname);
}
```

Compito del 03/09/2020: Python

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 03/09/2020

Esercizi:

a) [TEORIA] (2 punti) Convertire il numero decimale 743, in esadecimale illustrando tutti i passaggi.

b) [TEORIA] (4 punti) disegnare un albero binario di ricerca contenente numeri interi di profondità almeno 2, in modo che siano rispettati i seguenti punti: i) l'albero sia bilanciato; ii) quando si effettua la stampa con visita simmetrica (supponendo di visitare prima il sottoalbero sinistro), i numeri vengano stampati in ordine decrescente.

c) (7 punti) Scrivere una funzione che prenda in ingresso un array A di n elementi e restituisca la matrice AP quadrata in modo che la prima riga della matrice contenga gli elementi di A, la seconda gli elementi di A al quadrato, la terza gli elementi di A al cubo, etc. Esempio con n=3 (si ricorda che n deve essere generico).

$$A = (1\ 2\ 3) \rightarrow AP = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 4 & 9 \\ 1 & 8 & 27 \end{pmatrix}$$

d) (2 punti) Definire la struttura del nodo di un albero binario di ricerca che abbia come contenuto informativo: *name*, *age*, *state (booleano)*.

e) (15 punti) Data una lista di elementi (array in C): es. age_random = [33,121, 56, 8, 90, 2, 45, 7]

1. Scrivere una funzione che prenda in ingresso age_random e restituisca una nuova lista age con gli elementi ordinati in modo decrescente (3 punti)
2. Dati: i) una lista (array in C) age (del punto precedente); ii) una lista(array in C) es. *name* = ['Anna', 'Luca', 'Luigi', 'Marco', 'Stefano', 'Sandra', 'Massimo', 'Noemi']. Scrivere una funzione che prenda in ingresso le due liste age e name e che inserisca in ordine alfabetico gli utenti in un albero binario di ricerca avente come nodo quello definito al punto d). Sia state uguale a 0 per default. (esempi di inserimento: ['Anna', 2, 0], ['Luca', 7, 0], ['Luigi', 8, 0], ['Marco', 33, 0], etc.) (8 punti)
3. Scrivere una funzione che permetta la ricerca per nome e la variazione dello stato. Tale funzione deve prendere in ingresso il nuovo stato e variarlo se l'attuale è diverso dal vecchio, altrimenti deve segnalare l'errore. (4 punti)

Compito del 03/09/2020: teoria

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 03/09/2020

Esercizi:

a) [TEORIA] (2 punti) Convertire il numero decimale 743, in esadecimale illustrando tutti i passaggi.

b) [TEORIA] (4 punti) disegnare un albero binario di ricerca contenente numeri interi di profondità almeno 2, in modo che siano rispettati i seguenti punti: i) l'albero sia bilanciato; ii) quando si effettua la stampa con visita simmetrica (supponendo di visitare prima il sottoalbero sinistro), i numeri vengano stampati in ordine decrescente.

c) (7 punti) Scrivere una funzione che prenda in ingresso un array A di n elementi e restituisca la matrice AP quadrata in modo che la prima riga della matrice contenga gli elementi di A, la seconda gli elementi di A al quadrato, la terza gli elementi di A al cubo, etc. Esempio con n=3 (si ricorda che n deve essere generico).

$$A = (1\ 2\ 3) \rightarrow AP = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 4 & 9 \\ 1 & 8 & 27 \end{pmatrix}$$

d) (2 punti) Definire la struttura del nodo di un albero binario di ricerca che abbia come contenuto informativo: *name*, *age*, *state (booleano)*.

e) (15 punti) Data una lista di elementi (array in C): es. age_random = [33,121, 56, 8, 90, 2, 45, 7]

1. Scrivere una funzione che prenda in ingresso age_random e restituisca una nuova lista age con gli elementi ordinati in modo decrescente (3 punti)
2. Dati: i) una lista (array in C) age (del punto precedente); ii) una lista(array in C) es. *name* = ['Anna', 'Luca', 'Luigi', 'Marco', 'Stefano', 'Sandra', 'Massimo', 'Noemi']. Scrivere una funzione che prenda in ingresso le due liste age e name e che inserisca in ordine alfabetico gli utenti in un albero binario di ricerca avente come nodo quello definito al punto d). Sia state uguale a 0 per default. (esempi di inserimento: ['Anna', 2, 0], ['Luca', 7, 0], ['Luigi', 8, 0], ['Marco', 33, 0], etc.) (8 punti)
3. Scrivere una funzione che permetta la ricerca per nome e la variazione dello stato. Tale funzione deve prendere in ingresso il nuovo stato e variarlo se l'attuale è diverso dal vecchio, altrimenti deve segnalare l'errore. (4 punti)

Soluzione:

Esercizio a) (2 punti)

- Si converte in binario: [743]_10 = [7*10^2+4*10+3]_10 =
  = [111*1010*1010+100*1010+11]_2 =

```
      1010*1010
       0000
      11010
       0000
      1010
     1100100

        1100100*111
         1100100
        1100100
       1100100
     1010111100

      1010*100
       0000
      0000
     1010
    101000
```

```
1010111100+
0000101000+
0000000011=
1011100111
```

[1011100111]_2
Impacco i bit partendo da destra:
[0111]_2 = [7]_10=[7]_16 (meno significativo)
[1110]_2 = [14]_10=[E]_16
[0010]_2 = [2]_10=[2]_16

Quindi:
[743]_10 = [2E7]_16

Oppure si usa algoritmo dei resti successivi arrivando alla stessa conclusione

Compito del 03/09/2020: Teoria

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 03/09/2020

Esercizi:

a) [TEORIA] (2 punti) Convertire il numero decimale 743, in esadecimale illustrando tutti i passaggi.

b) [TEORIA] (4 punti) disegnare un albero binario di ricerca contenente numeri interi di profondità almeno 2, in modo che siano rispettati i seguenti punti: i) l'albero sia bilanciato; ii) quando si effettua la stampa con visita simmetrica (supponendo di visitare prima il sottoalbero sinistro), i numeri vengano stampati in ordine decrescente.

c) (7 punti) Scrivere una funzione che prenda in ingresso un array A di n elementi e restituisca la matrice AP quadrata in modo che la prima riga della matrice contenga gli elementi di A, la seconda gli elementi di A al quadrato, la terza gli elementi di A al cubo, etc. Esempio con n=3 (si ricorda che n deve essere generico).

$$A = (1\,2\,3) \rightarrow AP = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 4 & 9 \\ 1 & 8 & 27 \end{pmatrix}$$

d) (2 punti) Definire la struttura del nodo di un albero binario di ricerca che abbia come contenuto informativo: *name*, *age*, *state (booleano)*.

e) (15 punti) Data una lista di elementi (array in C): es. age_random = [33,121, 56, 8, 90, 2, 45, 7]

1. Scrivere una funzione che prenda in ingresso age_random e restituisca una nuova lista age con gli elementi ordinati in modo decrescente (3 punti)
2. Dati: i) una lista (array in C) age (del punto precedente); ii) una lista(array in C) es. *name* = ['Anna', 'Luca', 'Luigi', 'Marco', 'Stefano', 'Sandra', 'Massimo', 'Noemi']. Scrivere una funzione che prenda in ingresso le due liste age e name e che inserisca in ordine alfabetico gli utenti in un albero binario di ricerca avente come nodo quello definito al punto d). Sia state uguale a 0 per default. (esempi di inserimento: ['Anna', 2, 0], ['Luca', 7, 0], ['Luigi', 8, 0], ['Marco', 33, 0], etc.) (8 punti)
3. Scrivere una funzione che permetta la ricerca per nome e la variazione dello stato. Tale funzione deve prendere in ingresso il nuovo stato e variarlo se l'attuale è diverso dal vecchio, altrimenti deve segnalare l'errore. (4 punti)

Esercizio b): (4 punti)

Dati i vincoli imposti nel testo, e la definizione classica di albero binario di ricerca, NON è possibile stampare in ordine decrescente, con visita da sinistra a destra!
Tutti i nodi del sottoalbero sinistro sono minori della radice.

Compito del 03/09/2020: Python

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 03/09/2020

Esercizi:

a) [TEORIA] (2 punti) Convertire il numero decimale 743, in esadecimale illustrando tutti i passaggi.

b) [TEORIA] (4 punti) disegnare un albero binario di ricerca contenente numeri interi di profondità almeno 2, in modo che siano rispettati i seguenti punti: i) l'albero sia bilanciato; ii) quando si effettua la stampa con visita simmetrica (supponendo di visitare prima il sottoalbero sinistro), i numeri vengano stampati in ordine decrescente.

c) (7 punti) Scrivere una funzione che prenda in ingresso un array A di n elementi e restituisca la matrice AP quadrata in modo che la prima riga della matrice contenga gli elementi di A, la seconda gli elementi di A al quadrato, la terza gli elementi di A al cubo, etc. Esempio con n=3 (si ricorda che n deve essere generico).

$$A = (1\,2\,3) \rightarrow AP = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 4 & 9 \\ 1 & 8 & 27 \end{pmatrix}$$

d) (2 punti) Definire la struttura del nodo di un albero binario di ricerca che abbia come contenuto informativo: *name*, *age*, *state (booleano)*.

e) (15 punti) Data una lista di elementi (array in C): es. age_random = [33,121, 56, 8, 90, 2, 45, 7]

1. Scrivere una funzione che prenda in ingresso age_random e restituisca una nuova lista age con gli elementi ordinati in modo decrescente (3 punti)
2. Dati: i) una lista (array in C) age (del punto precedente); ii) una lista(array in C) es. *name* = ['Anna', 'Luca', 'Luigi', 'Marco', 'Stefano', 'Sandra', 'Massimo', 'Noemi']. Scrivere una funzione che prenda in ingresso le due liste age e name e che inserisca in ordine alfabetico gli utenti in un albero binario di ricerca avente come nodo quello definito al punto d). Sia state uguale a 0 per default. (esempi di inserimento: ['Anna', 2, 0], ['Luca', 7, 0], ['Luigi', 8, 0], ['Marco', 33, 0], etc.) (8 punti)
3. Scrivere una funzione che permetta la ricerca per nome e la variazione dello stato. Tale funzione deve prendere in ingresso il nuovo stato e variarlo se l'attuale è diverso dal vecchio, altrimenti deve segnalare l'errore. (4 punti)

Esercizio c): (8 punti)

```
def matrix_from_array(a):
    dim = len(a)
    A = []
    for i in range(dim):
        row = []
        for j in range(dim):
            element = math.pow(a[j], i+1)
            row.append(element)
        A.append(row)
    return A
```

Compito del 03/09/2020: Python

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 03/09/2020

Esercizi:

a) [TEORIA] (2 punti) Convertire il numero decimale 743, in esadecimale illustrando tutti i passaggi.

b) [TEORIA] (4 punti) disegnare un albero binario di ricerca contenente numeri interi di profondità almeno 2, in modo che siano rispettati i seguenti punti: i) l'albero sia bilanciato; ii) quando si effettua la stampa con visita simmetrica (supponendo di visitare prima il sottoalbero sinistro), i numeri vengano stampati in ordine decrescente.

c) (7 punti) Scrivere una funzione che prenda in ingresso un array A di n elementi e restituisca la matrice AP quadrata in modo che la prima riga della matrice contenga gli elementi di A, la seconda gli elementi di A al quadrato, la terza gli elementi di A al cubo, etc. Esempio con n=3 (si ricorda che n deve essere generico).

$$A = (1\,2\,3) \rightarrow AP = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 4 & 9 \\ 1 & 8 & 27 \end{pmatrix}$$

d) (2 punti) Definire la struttura del nodo di un albero binario di ricerca che abbia come contenuto informativo: *name*, *age*, *state (booleano)*.

e) (15 punti) Data una lista di elementi (array in C): es. age_random = [33,121, 56, 8, 90, 2, 45, 7]

1. Scrivere una funzione che prenda in ingresso age_random e restituisca una nuova lista age con gli elementi ordinati in modo decrescente (3 punti)
2. Dati: i) una lista (array in C) age (del punto precedente); ii) una lista(array in C) es. *name* = ['Anna', 'Luca', 'Luigi', 'Marco', 'Stefano', 'Sandra', 'Massimo', 'Noemi']. Scrivere una funzione che prenda in ingresso le due liste age e name e che inserisca in ordine alfabetico gli utenti in un albero binario di ricerca avente come nodo quello definito al punto d). Sia state uguale a 0 per default. (esempi di inserimento: ['Anna', 2, 0], ['Luca', 7, 0], ['Luigi', 8, 0], ['Marco', 33, 0], etc.) (8 punti)
3. Scrivere una funzione che permetta la ricerca per nome e la variazione dello stato. Tale funzione deve prendere in ingresso il nuovo stato e variarlo se l'attuale è diverso dal vecchio, altrimenti deve segnalare l'errore. (4 punti)

Esercizio d): (4 punti)

```
class Node:
    def __init__(self, data, username, age, state):
        self.username = username
        self.age = age
        self.state = state
        self.right = None
        self.left = None
```

Compito del 03/09/2020: Python

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 03/09/2020

Esercizi:

a) [TEORIA] (2 punti) Convertire il numero decimale 743, in esadecimale illustrando tutti i passaggi.

b) [TEORIA] (4 punti) disegnare un albero binario di ricerca contenente numeri interi di profondità almeno 2, in modo che siano rispettati i seguenti punti: i) l'albero sia bilanciato; ii) quando si effettua la stampa con visita simmetrica (supponendo di visitare prima il sottoalbero sinistro), i numeri vengano stampati in ordine decrescente.

c) (7 punti) Scrivere una funzione che prenda in ingresso un array A di n elementi e restituisca la matrice AP quadrata in modo che la prima riga della matrice contenga gli elementi di A, la seconda gli elementi di A al quadrato, la terza gli elementi di A al cubo, etc. Esempio con n=3 (si ricorda che n deve essere generico).

$$A = (1\,2\,3) \rightarrow AP = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 4 & 9 \\ 1 & 8 & 27 \end{pmatrix}$$

d) (2 punti) Definire la struttura del nodo di un albero binario di ricerca che abbia come contenuto informativo: *name*, *age*, *state (booleano)*.

e) (15 punti) Data una lista di elementi (array in C): es. age_random = [33,121, 56, 8, 90, 2, 45, 7]

1. Scrivere una funzione che prenda in ingresso age_random e restituisca una nuova lista age con gli elementi ordinati in modo decrescente (3 punti)
2. Dati: i) una lista (array in C) age (del punto precedente); ii) una lista(array in C) es. *name* = ['Anna', 'Luca', 'Luigi', 'Marco', 'Stefano', 'Sandra', 'Massimo', 'Noemi']. Scrivere una funzione che prenda in ingresso le due liste age e name e che inserisca in ordine alfabetico gli utenti in un albero binario di ricerca avente come nodo quello definito al punto d). Sia state uguale a 0 per default. (esempi di inserimento: ['Anna', 2, 0], ['Luca', 7, 0], ['Luigi', 8, 0], ['Marco', 33, 0], etc.) (8 punti)
3. Scrivere una funzione che permetta la ricerca per nome e la variazione dello stato. Tale funzione deve prendere in ingresso il nuovo stato e variarlo se l'attuale è diverso dal vecchio, altrimenti deve segnalare l'errore. (4 punti)

Esercizio e)

```
----------punto 1)
def minimuPosition(values, start):
    minPos = start
    for i in range(start + 1, len(values)):
        if values[i] < values[minPos]:
            minPos = i
    return minPos

def selectionSort(values):
    for i in range(len(values)):
        minPos = minimuPosition(values, i)
        temp = values[minPos]
        values[minPos] = values[i]
        values[i] = temp
```

```
def selectionSort_new_list(values):
    val_old = []
    for i in values:
        val_old.append(i)

    for i in range(len(values)):
        minPos = minimuPosition(values, i)
        temp = values[minPos]
        values[minPos] = values[i]
        values[i] = temp

    val_new=[]
    for i in values:
        val_new.append(i)

    del values[:]
    for i in val_old:
        values.append(i)
```

Compito del 03/09/2020: Python

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 03/09/2020

Esercizi:

a) [TEORIA] (2 punti) Convertire il numero decimale 743, in esadecimale illustrando tutti i passaggi.

b) [TEORIA] (4 punti) disegnare un albero binario di ricerca contenente numeri interi di profondità almeno 2, in modo che siano rispettati i seguenti punti: i) l'albero sia bilanciato; ii) quando si effettua la stampa con visita simmetrica (supponendo di visitare prima il sottoalbero sinistro), i numeri vengano stampati in ordine decrescente.

c) (7 punti) Scrivere una funzione che prenda in ingresso un array A di n elementi e restituisca la matrice AP quadrata in modo che la prima riga della matrice contenga gli elementi di A, la seconda gli elementi di A al quadrato, la terza gli elementi di A al cubo, etc. Esempio con n=3 (si ricorda che n deve essere generico).

$$A = (1\,2\,3) \rightarrow AP = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 4 & 9 \\ 1 & 8 & 27 \end{pmatrix}$$

d) (2 punti) Definire la struttura del nodo di un albero binario di ricerca che abbia come contenuto informativo: *name*, *age*, *state (booleano)*.

e) (15 punti) Data una lista di elementi (array in C): es. age_random = [33,121, 56, 8, 90, 2, 45, 7]

1. Scrivere una funzione che prenda in ingresso age_random e restituisca una nuova lista age con gli elementi ordinati in modo decrescente (3 punti)
2. Dati: i) una lista (array in C) age (del punto precedente); ii) una lista(array in C) es. *name* = ['Anna', 'Luca', 'Luigi', 'Marco', 'Stefano', 'Sandra', 'Massimo', 'Noemi']. Scrivere una funzione che prenda in ingresso le due liste age e name e che inserisca in ordine alfabetico gli utenti in un albero binario di ricerca avente come nodo quello definito al punto d). Sia state uguale a 0 per default. (esempi di inserimento: ['Anna', 2, 0], ['Luca', 7, 0], ['Luigi', 8, 0], ['Marco', 33, 0], etc.) (8 punti)
3. Scrivere una funzione che permetta la ricerca per nome e la variazione dello stato. Tale funzione deve prendere in ingresso il nuovo stato e variarlo se l'attuale è diverso dal vecchio, altrimenti deve segnalare l'errore. (4 punti)

Esercizio e)

```
----------punto 2)
def insert_node(n, name, age, state): #data, username, age, state)

    if name < n.name:
      if n.left:
        insert_node(n.left, name, age, state)

      else:
        n.left = Node(name, age, state)
    else:
      if n.right:
        insert_node(n.right, name, age, state)
      else:
        n.right = Node(name, age, state)

def insert_all(age, name):
  i = 0
  for number in age:
    if i==0:
      root = Node(name[i], number, 0) #creo radice
    else:
      insert_node(root, name[i], number, 0)
    i=i+1
  return root
```

Compito del 03/09/2020: Python

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 03/09/2020

Esercizi:

a) [TEORIA] (2 punti) Convertire il numero decimale 743, in esadecimale illustrando tutti i passaggi.

b) [TEORIA] (4 punti) disegnare un albero binario di ricerca contenente numeri interi di profondità almeno 2, in modo che siano rispettati i seguenti punti: i) l'albero sia bilanciato; ii) quando si effettua la stampa con visita simmetrica (supponendo di visitare prima il sottoalbero sinistro), i numeri vengano stampati in ordine decrescente.

c) (7 punti) Scrivere una funzione che prenda in ingresso un array A di n elementi e restituisca la matrice AP quadrata in modo che la prima riga della matrice contenga gli elementi di A, la seconda gli elementi di A al quadrato, la terza gli elementi di A al cubo, etc. Esempio con n=3 (si ricorda che n deve essere generico).

$$A = (1\,2\,3) \rightarrow AP = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 4 & 9 \\ 1 & 8 & 27 \end{pmatrix}$$

d) (2 punti) Definire la struttura del nodo di un albero binario di ricerca che abbia come contenuto informativo: *name*, *age*, *state (booleano)*.

e) (15 punti) Data una lista di elementi (array in C): es. age_random = [33,121, 56, 8, 90, 2, 45, 7]

1. Scrivere una funzione che prenda in ingresso age_random e restituisca una nuova lista age con gli elementi ordinati in modo decrescente (3 punti)
2. Dati: i) una lista (array in C) age (del punto precedente); ii) una lista(array in C) es. *name* = ['Anna', 'Luca', 'Luigi', 'Marco', 'Stefano', 'Sandra', 'Massimo', 'Noemi']. Scrivere una funzione che prenda in ingresso le due liste age e name e che inserisca in ordine alfabetico gli utenti in un albero binario di ricerca avente come nodo quello definito al punto d). Sia state uguale a 0 per default. (esempi di inserimento: ['Anna', 2, 0], ['Luca', 7, 0], ['Luigi', 8, 0], ['Marco', 33, 0], etc.) (8 punti)
3. Scrivere una funzione che permetta la ricerca per nome e la variazione dello stato. Tale funzione deve prendere in ingresso il nuovo stato e variarlo se l'attuale è diverso dal vecchio, altrimenti deve segnalare l'errore. (4 punti)

Esercizio e)

```
----------punto 3)
def search_modify(n, name, state):
  if n:
    if name < n.name:
      return search_modify(n.left, name, state)
    elif name > n.name:
      return search_modify(n.right, name, state)
    else:
      if n.state == state:
        print('Lo stato era già aggiornato, nessuna variazione! ecco: ', n.state)
      else:
        n.state = state
        print('Stato aggiornato: ', n.state)
      return True, n
  else:
    print('Non esiste questo nome!')
    return False, None
```

Compito del 03/09/2020: C

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 03/09/2020

Esercizi:

a) [TEORIA] (2 punti) Convertire il numero decimale 743, in esadecimale illustrando tutti i passaggi.

b) [TEORIA] (4 punti) disegnare un albero binario di ricerca contenente numeri interi di profondità almeno 2, in modo che siano rispettati i seguenti punti: i) l'albero sia bilanciato; ii) quando si effettua la stampa con visita simmetrica (supponendo di visitare prima il sottoalbero sinistro), i numeri vengano stampati in ordine decrescente.

c) (7 punti) Scrivere una funzione che prenda in ingresso un array A di n elementi e restituisca la matrice AP quadrata in modo che la prima riga della matrice contenga gli elementi di A, la seconda gli elementi di A al quadrato, la terza gli elementi di A al cubo, etc. Esempio con n=3 (si ricorda che n deve essere generico).

$$A = (1\,2\,3) \rightarrow AP = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 4 & 9 \\ 1 & 8 & 27 \end{pmatrix}$$

d) (2 punti) Definire la struttura del nodo di un albero binario di ricerca che abbia come contenuto informativo: *name*, *age*, *state (booleano)*.

e) (15 punti) Data una lista di elementi (array in C): es. age_random = [33,121, 56, 8, 90, 2, 45, 7]

1. Scrivere una funzione che prenda in ingresso age_random e restituisca una nuova lista age con gli elementi ordinati in modo decrescente (3 punti)
2. Dati: i) una lista (array in C) age (del punto precedente); ii) una lista(array in C) es. *name* = ['Anna', 'Luca', 'Luigi', 'Marco', 'Stefano', 'Sandra', 'Massimo', 'Noemi']. Scrivere una funzione che prenda in ingresso le due liste age e name e che inserisca in ordine alfabetico gli utenti in un albero binario di ricerca avente come nodo quello definito al punto d). Sia state uguale a 0 per default. (esempi di inserimento: ['Anna', 2, 0], ['Luca', 7, 0], ['Luigi', 8, 0], ['Marco', 33, 0], etc.) (8 punti)
3. Scrivere una funzione che permetta la ricerca per nome e la variazione dello stato. Tale funzione deve prendere in ingresso il nuovo stato e variarlo se l'attuale è diverso dal vecchio, altrimenti deve segnalare l'errore. (4 punti)

Esercizio c)

```
float power(float v, int p){
        float r = 1;
        for (int i=1; i < p;j++){
                r*=v;
        }
        return r;
}

float* power_matrix(float* vector,  int vec_size){
float* matrix = (float*)malloc(vec_size*vec_size*sizeof(float));
        for (int i =0; i < vec_size; i++){
                for (int j =0; j < vec_size; j++){
                        matrix[i*vec_size+j] = power(vector[j],i+1);
                }
        }
        return matrix;
}

void print_mat(float* m, int M, int N){
        for (int i=0; i < M; i++){
                for (int j=0; j < N; j++){
                        printf("%f ", m[i*M+j]);
                }
                printf("\n");
        }
}
```

Compito del 03/09/2020: C

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 03/09/2020

Esercizi:

a) [TEORIA] (2 punti) Convertire il numero decimale 743, in esadecimale illustrando tutti i passaggi.

b) [TEORIA] (4 punti) disegnare un albero binario di ricerca contenente numeri interi di profondità almeno 2, in modo che siano rispettati i seguenti punti: i) l'albero sia bilanciato; ii) quando si effettua la stampa con visita simmetrica (supponendo di visitare prima il sottoalbero sinistro), i numeri vengano stampati in ordine decrescente.

c) (7 punti) Scrivere una funzione che prenda in ingresso un array A di n elementi e restituisca la matrice AP quadrata in modo che la prima riga della matrice contenga gli elementi di A, la seconda gli elementi di A al quadrato, la terza gli elementi di A al cubo, etc. Esempio con n=3 (si ricorda che n deve essere generico).

$$A = (1\,2\,3) \rightarrow AP = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 4 & 9 \\ 1 & 8 & 27 \end{pmatrix}$$

d) (2 punti) Definire la struttura del nodo di un albero binario di ricerca che abbia come contenuto informativo: *name*, *age*, *state (booleano)*.

e) (15 punti) Data una lista di elementi (array in C): es. age_random = [33,121, 56, 8, 90, 2, 45, 7]

1. Scrivere una funzione che prenda in ingresso age_random e restituisca una nuova lista age con gli elementi ordinati in modo decrescente (3 punti)
2. Dati: i) una lista (array in C) age (del punto precedente); ii) una lista(array in C) es. *name* = ['Anna', 'Luca', 'Luigi', 'Marco', 'Stefano', 'Sandra', 'Massimo', 'Noemi']. Scrivere una funzione che prenda in ingresso le due liste age e name e che inserisca in ordine alfabetico gli utenti in un albero binario di ricerca avente come nodo quello definito al punto d). Sia state uguale a 0 per default. (esempi di inserimento: ['Anna', 2, 0], ['Luca', 7, 0], ['Luigi', 8, 0], ['Marco', 33, 0], etc.) (8 punti)
3. Scrivere una funzione che permetta la ricerca per nome e la variazione dello stato. Tale funzione deve prendere in ingresso il nuovo stato e variarlo se l'attuale è diverso dal vecchio, altrimenti deve segnalare l'errore. (4 punti)

Esercizio d)

```
struct btree {
    struct btree* left;
    struct btree* right;
    char* name;
    int age;
    Boolean state;

};
```

Compito del 03/09/2020: C

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 03/09/2020

Esercizi:

a) [TEORIA] (2 punti) Convertire il numero decimale 743, in esadecimale illustrando tutti i passaggi.

b) [TEORIA] (4 punti) disegnare un albero binario di ricerca contenente numeri interi di profondità almeno 2, in modo che siano rispettati i seguenti punti: i) l'albero sia bilanciato; ii) quando si effettua la stampa con visita simmetrica (supponendo di visitare prima il sottoalbero sinistro), i numeri vengano stampati in ordine decrescente.

c) (7 punti) Scrivere una funzione che prenda in ingresso un array A di n elementi e restituisca la matrice AP quadrata in modo che la prima riga della matrice contenga gli elementi di A, la seconda gli elementi di A al quadrato, la terza gli elementi di A al cubo, etc. Esempio con n=3 (si ricorda che n deve essere generico).

$$A = (1\,2\,3) \rightarrow AP = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 4 & 9 \\ 1 & 8 & 27 \end{pmatrix}$$

d) (2 punti) Definire la struttura del nodo di un albero binario di ricerca che abbia come contenuto informativo: *name*, *age*, *state (booleano)*.

e) (15 punti) Data una lista di elementi (array in C): es. age_random = [33,121, 56, 8, 90, 2, 45, 7]

1. Scrivere una funzione che prenda in ingresso age_random e restituisca una nuova lista age con gli elementi ordinati in modo decrescente (3 punti)
2. Dati: i) una lista (array in C) age (del punto precedente); ii) una lista(array in C) es. *name* = ['Anna', 'Luca', 'Luigi', 'Marco', 'Stefano', 'Sandra', 'Massimo', 'Noemi']. Scrivere una funzione che prenda in ingresso le due liste age e name e che inserisca in ordine alfabetico gli utenti in un albero binario di ricerca avente come nodo quello definito al punto d). Sia state uguale a 0 per default. (esempi di inserimento: ['Anna', 2, 0], ['Luca', 7, 0], ['Luigi', 8, 0], ['Marco', 33, 0], etc.) (8 punti)
3. Scrivere una funzione che permetta la ricerca per nome e la variazione dello stato. Tale funzione deve prendere in ingresso il nuovo stato e variarlo se l'attuale è diverso dal vecchio, altrimenti deve segnalare l'errore. (4 punti)

Esercizio e1)

```
struct list * sort_list(int* age_random, int N) {
    struct list *s = NULL;
    for (int i = 0; i < N; i++)
        rordered_insert(&s, age_random[i]);
    return s;
}


void rordered_insert(struct list** ptr, float value) {
    while ((*ptr) != NULL && (*ptr)->value > value) {
        //printf("v: %f, current: %f\n",value, (*ptr)->value);
        ptr = &((*ptr)->next);

    }
    pre_insert(ptr, value);
    //printf("inserted!\n");
}
```

Compito del 03/09/2020: C

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 03/09/2020

Esercizi:

a) [TEORIA] (2 punti) Convertire il numero decimale 743, in esadecimale illustrando tutti i passaggi.

b) [TEORIA] (4 punti) disegnare un albero binario di ricerca contenente numeri interi di profondità almeno 2, in modo che siano rispettati i seguenti punti: i) l'albero sia bilanciato; ii) quando si effettua la stampa con visita simmetrica (supponendo di visitare prima il sottoalbero sinistro), i numeri vengano stampati in ordine decrescente.

c) (7 punti) Scrivere una funzione che prenda in ingresso un array A di n elementi e restituisca la matrice AP quadrata in modo che la prima riga della matrice contenga gli elementi di A, la seconda gli elementi di A al quadrato, la terza gli elementi di A al cubo, etc. Esempio con n=3 (si ricorda che n deve essere generico).

$$A = (1\,2\,3) \rightarrow AP = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 4 & 9 \\ 1 & 8 & 27 \end{pmatrix}$$

d) (2 punti) Definire la struttura del nodo di un albero binario di ricerca che abbia come contenuto informativo: *name*, *age*, *state (booleano)*.

e) (15 punti) Data una lista di elementi (array in C): es. age_random = [33,121, 56, 8, 90, 2, 45, 7]

1. Scrivere una funzione che prenda in ingresso age_random e restituisca una nuova lista age con gli elementi ordinati in modo decrescente (3 punti)
2. Dati: i) una lista (array in C) age (del punto precedente); ii) una lista(array in C) es. *name* = ['Anna', 'Luca', 'Luigi', 'Marco', 'Stefano', 'Sandra', 'Massimo', 'Noemi']. Scrivere una funzione che prenda in ingresso le due liste age e name e che inserisca in ordine alfabetico gli utenti in un albero binario di ricerca avente come nodo quello definito al punto d). Sia state uguale a 0 per default. (esempi di inserimento: ['Anna', 2, 0], ['Luca', 7, 0], ['Luigi', 8, 0], ['Marco', 33, 0], etc.) (8 punti)
3. Scrivere una funzione che permetta la ricerca per nome e la variazione dello stato. Tale funzione deve prendere in ingresso il nuovo stato e variarlo se l'attuale è diverso dal vecchio, altrimenti deve segnalare l'errore. (4 punti)

Esercizio e2)

```
void insert_inorder(struct btree ** ptrptr, int age, char* name) {
    if (*ptrptr != NULL) {
        if (strcmp(name, (*ptrptr)->name)<0)
            insert_inorder(&((*ptrptr)->left), age, name);
        else
            insert_inorder(&((*ptrptr)->right), age, name);
    }
    else {
        (*ptrptr) = (struct btree *)malloc(sizeof(struct btree));

        (*ptrptr)->age = age;
        (*ptrptr)->name = (char*)malloc(strlen(name) * sizeof(char));
        strcpy((*ptrptr)->name, name);
        (*ptrptr)->state = FALSE;
        (*ptrptr)->left = NULL;
        (*ptrptr)->right = NULL;
    }
}
struct btree* insert(char** names, int* ages, int N) {
    struct btree* root = NULL;
    for (int i = 0; i < N; i++) {
        insert_inorder(&root, ages[i], names[i]);
    }
    return root;
}
```

Compito del 03/09/2020: C

Università di Firenze - Facoltà di Ingegneria
Corso di laurea in Ingegneria Meccanica e Gestionale (A-H)
Prova Scritta di Fondamenti di Informatica A.A. 2019/20
Prof. Michela Paolucci - 03/09/2020

Esercizi:

a) [TEORIA] (2 punti) Convertire il numero decimale 743, in esadecimale illustrando tutti i passaggi.

b) [TEORIA] (4 punti) disegnare un albero binario di ricerca contenente numeri interi di profondità almeno 2, in modo che siano rispettati i seguenti punti: i) l'albero sia bilanciato; ii) quando si effettua la stampa con visita simmetrica (supponendo di visitare prima il sottoalbero sinistro), i numeri vengano stampati in ordine decrescente.

c) (7 punti) Scrivere una funzione che prenda in ingresso un array A di n elementi e restituisca la matrice AP quadrata in modo che la prima riga della matrice contenga gli elementi di A, la seconda gli elementi di A al quadrato, la terza gli elementi di A al cubo, etc. Esempio con n=3 (si ricorda che n deve essere generico).

$$A = (1\,2\,3) \rightarrow AP = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 4 & 9 \\ 1 & 8 & 27 \end{pmatrix}$$

d) (2 punti) Definire la struttura del nodo di un albero binario di ricerca che abbia come contenuto informativo: *name*, *age*, *state (booleano)*.

e) (15 punti) Data una lista di elementi (array in C): es. age_random = [33,121, 56, 8, 90, 2, 45, 7]

1. Scrivere una funzione che prenda in ingresso age_random e restituisca una nuova lista age con gli elementi ordinati in modo decrescente (3 punti)
2. Dati: i) una lista (array in C) age (del punto precedente); ii) una lista(array in C) es. *name* = ['Anna', 'Luca', 'Luigi', 'Marco', 'Stefano', 'Sandra', 'Massimo', 'Noemi']. Scrivere una funzione che prenda in ingresso le due liste age e name e che inserisca in ordine alfabetico gli utenti in un albero binario di ricerca avente come nodo quello definito al punto d). Sia state uguale a 0 per default. (esempi di inserimento: ['Anna', 2, 0], ['Luca', 7, 0], ['Luigi', 8, 0], ['Marco', 33, 0], etc.) (8 punti)
3. Scrivere una funzione che permetta la ricerca per nome e la variazione dello stato. Tale funzione deve prendere in ingresso il nuovo stato e variarlo se l'attuale è diverso dal vecchio, altrimenti deve segnalare l'errore. (4 punti)

Esercizio e3)

```
Boolean modify_state(struct btree* r, char* name, Boolean new_state) {
    if (r) {
        if (strcmp(r->name, name) <0)
            return modify_state(r->left, name, new_state);
        else if (strcmp(r->name, name) >0)
            return modify_state(r->right, name, new_state);
        else {
            if (r->state == new_state) {
                printf("Non modifico nulla!!\n");
                return FALSE;
            }
            else {
                printf("Modificato! %s \n", r->name);
                r->state = new_state;
                return TRUE;
            }
        }
    }
    else {
        return FALSE;
    }
}
```

Bibliografia - IMPORT

- Importare librerie:
  - https://docs.python.org/3/py-modindex.html
- Online python tools:
  - https://repl.it/languages/python3
- Documenti ufficiali Python:
  - https://docs.python.it
- Classi:
  - http://www.thinkpython2.com/code/Point1.py
  - http://www.thinkpython2.com/code/Point1_soln.py
  - http://www.thinkpython2.com/code/Time1.py
  - http://www.thinkpython2.com/code/Time1_soln.py

Bibliografia

- http://disi.unitn.it/~teso/courses/informatica/python_functions.html
- https://www.python.it/doc/Howtothink/Howtothink-html-it/chap04.htm
- https://www.python.it/doc/Howtothink/Howtothink-html-it/dex.htm
- Concetti di informatica e fondamenti di Python, Maggioli Editore, Cay HorstMann, Rance D. Necaise, seconda edizione, 2019
- Pensare da informatico - Imparare con Python, Allen Downey, Jeffrey Elkner, Chris Meyers, Green Tea Press Wellesley, Massachusetts

Fondamenti di Informatica

AA 2019/2020

Eng. Ph.D. Michela Paolucci

DISIT Lab http://www.disit.dinfo.unifi.it
Department of Information Engineering, DINFO
University of Florence
Via S. Marta 3, 50139, Firenze, Italy
tel: +39-055-2758515, fax: +39-055-2758570
michela.paolucci@unifi.it