

LABORATORIO DI PYTHON

DEFINIZIONI DI BOOLEANI, SELEZIONE ED USO DEGLI INPUT

13 Marzo 2019

CORREZIONE ESERCIZI PER OGGI

Scrivere una funzione che non ha nessun parametro, **non restituisce nulla**, ma stampa a video il valore (approssimato) di \sqrt{e} (radice quadrata del numero di Nepero).

Scrivere una funzione che non ha nessun parametro, **non restituisce nulla**, ma stampa a video il valore (approssimato) di \sqrt{e} (radice quadrata del numero di Nepero).

```
1 import math
2 def radicedie():
3     """Funzione che stampa la radice quadrata del numero di
4     Nepero"""
5     print(math.sqrt(math.e))
```

Sia C il capitale iniziale di un investimento. Sia r il tasso di interesse (espresso come decimale, es 0.03), sia n il numero di volte che gli interessi vengono calcolati ogni anno e sia t il numero di anni. Il capitale finale M si calcola allora come:

$$M = C \left(1 + \frac{r}{n}\right)^{nt}$$

Scrivere una funzione che ha come parametri C , r , n , t e **restituisce** il valore di M , ma **non stampa nulla**. Nello stesso file scrivere poi un esempio che, **usando la funzione**, stampa: **Capitale finale per investimento di 10.000, calcolo mensile, tasso 8%, per 2 anni: 11728.879317453097**

ESERCIZIO 2 PER CASA

```
1 def calcolaCapitaleFinale(C, r, n, t):
2     """ Funzione che calcola il capitale finale di un
3         investimento.
4         C = capitale iniziale ,
5         r = tasso di interesse decimale
6         n = numero di calcoli dell'interesse
7         t = numero di anni
8         """
9     M = C*(1+(r/n))**(n*t)
10    return M
11
12 Mf = calcolaCapitaleFinale(10000, 0.08, 12, 2)
13 print("Capitale finale per investimento di 10.000, calcolo
14     mensile, tasso 8%, per 2 anni: ", Mf)
```

BOOLEANI E SELEZIONE (`if`)

```
1  if (condizione) :  
2      istruzioni interne all'if  
3      ...  
4      istruzioni interne all'if  
5  istruzioni fuori dalla selezione
```

- La condizione (espressione booleana, $\mathbb{B} = \{\text{True}, \text{False}\}$) viene valutata.
- **Solo se** l'espressione booleana vale **True** allora si eseguono le istruzioni all'interno del costrutto di selezione **if** (notare l'indentazione).
- Se la condizione vale **False**, le istruzioni all'interno dell'**if non** sono eseguite, si passa alle istruzioni successive.


```
1 def sconto(age):  
2     if (age >= 60):  
3         return "Sconto pensionati"
```

- Indicano uguaglianza, disuguaglianza o relazioni d'ordine:
 - $x==y$ (x uguale a y)
 - $x!=y$ (x diverso da y)
 - $x>y$ (x maggiore di y)
 - $x<y$ (x minore di y)
 - $x>=y$ (x maggiore o uguale a y)
 - $x<=y$ (x minore o uguale a y)

La valutazione di tali espressioni restituisce un booleano.

- $x==2$
- $3==x$

le precedenti sono valide espressioni di **confronto**, mentre la seguente...

- $4=x$
no!

- `(a and b)` vale `True` se e solo se **sia a che b** valgono `True`
- `(a or b)` vale `True` se e solo se **almeno uno tra a e b** vale `True`
- `(not a)` vale `True` se `a` vale `False`, e viceversa.

```
1  if (condizione) :  
2      istruzioni interne all'if  
3      ...  
4      istruzioni interne all'if  
5  else :  
6      istruzioni interne all'else  
7      ...  
8      istruzioni interne all'else  
9  istruzioni fuori dalla selezione
```

- La condizione viene valutata.
- **Se** l'espressione booleana vale **True** allora si eseguono **solo** le istruzioni all'interno dell'**if**
- **Altrimenti** (cioè se la condizione vale **False**), si eseguono **solo** le istruzioni all'interno dell'**else**.
- In ogni caso si passa poi alle istruzioni successive

```
1 def sconto(age):  
2     if (age >= 60):  
3         return "Sconto pensionati"  
4     else:  
5         return "Biglietto intero"
```

```
1  if (condizione) :  
2      istruzioni interne all'if  
3  elif (condizione) :  
4      istruzioni interne a un elif  
5      ...  
6  elif (condizione) :  
7      istruzioni interne a un elif  
8      ...  
9  else :  
10     istruzioni interne all'else
```

Si usano **quando ci sono più di due casi**

- **Se** la condizione dell'**if** vale **True** allora si eseguono **solo** le istruzioni all'interno dell'**if**
- **Altrimenti** (cioè se la condizione dell'**if** vale **False**), si valuta la condizione del primo **elif**.
- **Se** tale condizione vale **True** allora si eseguono **solo** le istruzioni all'interno di tale **elif**
- Se invece è falsa, si passa al successivo **elif**, se presente ...e così via ...
- Se tutte le condizioni testate nell'ordine sono false, ed è presente un **else**, si eseguono **solo** le istruzioni all'interno dell'**else**.

Modificare la funzione sconto di modo che restituisca:

- “**Gratis**” se l’età è minore di 6
- “**Sconto bambini**” se l’età è minore o uguale a 12
- “**Biglietto intero**” se è compresa tra 12 e 60
- “**Sconto pensionati**” se l’età è maggiore o uguale a 60
- “**Gratis**” se l’età è maggiore di 70

Ricordarsi di testare i casi limite!

```
>>> sconto(0)
'Gratis'
>>> sconto(5)
'Gratis'
>>> sconto(6)
'Sconto bambini'
>>> sconto(8)
'Sconto bambini'
>>> sconto(12)
'Sconto bambini'
>>> sconto(13)
'Biglietto intero'
```

```
>>> sconto(44)
'Biglietto intero'
>>> sconto(59)
'Biglietto intero'
>>> sconto(60)
'Sconto pensionati'
>>> sconto(70)
'Sconto pensionati'
>>> sconto(71)
'Gratis'
>>> sconto(100)
'Gratis'
```


Qual è il valore di una funzione che non restituisce valori?

```
1 def f():  
2     x = 10  
3  
4 print(f())
```

- **None** è il valore delle espressioni che non restituiscono valori.
- Il suo tipo è **NoneType**

Un nuovo tipo: **NoneType**

- Non è possibile effettuare operazioni tra variabili **NoneType**.
- È possibile confrontare variabili **NoneType**:
 - $M == \text{None} \rightarrow \text{vero (True)}$ se M è **None**.
 - $M != \text{None} \rightarrow \text{vero (True)}$ se M è un valore **diverso** da **None**.

Scrivere una funzione che prende come parametri i coefficienti a e b di un'equazione di primo grado ($ax + b = 0$) e restituisce il valore di x che la rende un'identità (cioè la risolve per x). Nel caso in cui sia *indeterminata* oppure *impossibile*, la funzione **stampa** un opportuno messaggio di errore e poi restituisce **None**.

INPUT

Sintassi:

```
1 <var> = input("stringa descrittiva")
```

Sintassi:

```
1 <var> = input("stringa descrittiva")
```

Esempio:

```
1 nome = input("come ti chiami?")
```

Sintassi:

```
1 <var> = input("stringa descrittiva")
```

Esempio:

```
1 nome = input("come ti chiami?")
```

- All'utente viene visualizzato il messaggio: **come ti chiami?**
- Il programma attende che l'utente risponda e preme Invio
- La risposta viene legata, come **stringa**, alla variabile **nome**

Sintassi:

```
1 <var> = input("stringa descrittiva")
```

Esempio:

```
1 nome = input("come ti chiami?")
```

- All'utente viene visualizzato il messaggio: **come ti chiami?**
- Il programma attende che l'utente risponda e preme Invio
- La risposta viene legata, come **stringa**, alla variabile **nome**

```
1 nome = input("come ti chiami?")  
2 print("Ciao", nome, "!")
```

```
1 risposta = input("Quanti anni hai? ")
2 if (risposta < 18):
3     print("Non puoi ancora votare!")
4 else:
5     print("Vota con giudizio")
```

Questo codice è corretto?


```
1 risposta = input("Quanti anni hai? ")
2 if (risposta < 18):
3     print("Non puoi ancora votare!")
4 else:
5     print("Vota con giudizio")
```

Questo codice è corretto?

...

```
if (risposta < 18):
```

```
TypeError: unorderable types: str() < int()
```

Cosa significano le ultime due righe dell'errore?

Abbiamo detto che il risultato di `input` è memorizzato come stringa. A noi però serve come intero. Dobbiamo convertirlo esplicitamente con la funzione `int()`

```
1 risposta = int(input("Quanti anni hai? "))
2 if (risposta < 18):
3     print("Non puoi ancora votare!")
4 else:
5     print("Vota con giudizio")
```

Lo stesso vale per altri tipi, es `float`, `complex`...

- Fare attenzione a non confondere il comando `input` con la definizione di una funzione che prenda dei parametri di ingresso
- Fare attenzione alla differenza tra `return` e `print`.

```
1 import primogrado #il nome del file in cui avevamo scritto la
   funzione
2
3 print("Programma che risolve le equazioni ax+b=0")
4 print()
5 a = float(input("scrivi il valore di a: "))
6 b = float(input("scrivi il valore di b: "))
7 print(primogrado.equazione_primo_grado(a,b))
```

Scrivere un programma che chiede il nome e l'età di una persona, poi stampa un messaggio in cui la saluta cordialmente e le dice di che tipologia di biglietto può usufruire (usando la funzione **sconto**)

- Scrivere una funzione che preso come parametro un intero restituisca **True** se questo è pari e **False** altrimenti. Non stampare nulla.
- Scrivere un programma che, *importando il file precedente e usandone la funzione*, chieda all'utente di inserire un numero, e comunichi (stampando) all'utente se il numero è pari o dispari.

Scrivere una funzione `retta_passante_per` che presi quattro parametri x_0, y_0, x_1, y_1 **stampa** l'equazione della retta passante per (x_0, y_0) e (x_1, y_1) , se esiste, oppure stampa un opportuno messaggio di errore. La funzione non restituisce nulla.

(*) Provare a costruire un output “elegante”, analogo all'esempio seguente:

```
>>> retta_passante_per(1,3,2,4)
```

```
La retta passante per (1,3) e (2,4) ha equazione  
y = 1.0x + 2.0
```

ESERCIZI PER CASA

Scrivere una funzione che prende tre valori **float** a , b , c come parametri e che restituisca la/le soluzioni dell'equazione $ax^2 + bx + c = 0$, oppure stampi gli opportuni messaggi nei casi (fare attenzione a considerarli tutti!) in cui l'equazione sia impossibile o indeterminata e restituisca **None** in tali casi.

Scrivere una funzione che, presi tre valori come parametri, li stampi in ordine decrescente. Nello stesso file, scrivere un test per la funzione che chieda i 3 valori come **input** e li legga come **float**.

La tabella seguente riporta in euro le tariffe per il noleggio di uno scooter.

| Scooter | 24 ore | 2 giorni | 3 giorni | 4 giorni | Ogni giorno extra |
|---------|--------|----------|----------|----------|-------------------|
| Euro | 45,00 | 80,00 | 120,00 | 160,00 | 40,00 |

Scrivere una funzione che prenda come parametro il numero di giorni di noleggio e ne calcoli il costo totale. La funzione restituisce sempre un float (se `giorno < 1`, restituisce `0.0`). Nello stesso file, scrivere un test per la funzione che chieda in input il numero di giorni e stampi il costo totale di noleggio.

Dato l'anno, calcolare giorno e mese della Pasqua. La funzione prende come parametro x (l'anno)

| | Valore.. | ...da dividere per | risultati utili | |
|----|-------------------|--------------------|------------------|--------------|
| | | | Quoziente intero | Resto |
| 1 | x | 100 | b | c |
| 2 | $5b + c$ | 19 | - | a |
| 3 | $3(b + 25)$ | 4 | r | s |
| 4 | $8(b + 11)$ | 25 | t | - |
| 5 | $19a + r - t$ | 30 | - | h |
| 6 | $a + 11h$ | 319 | g | - |
| 7 | $60(5 - s) + c$ | 4 | j | k |
| 8 | $2j - k - h + g$ | 7 | - | m |
| 9 | $h - g + m + 110$ | 30 | n (mese) | q |
| 10 | $q + 5 - n$ | 32 | - | p (giorno) |

e restituisce p e n , che rappresentano giorno e mese della Pasqua nell'anno x .