Esercizi in Laboratorio

Informatica@SEFA 2018/2019 - Laboratorio 2

Massimo Lauria <massimo.lauria@uniroma1.it>
http://massimolauria.net/courses/infosefa2018/

Lunedì, 15 Ottobre 2018

Esercitazione

1. scrivere **un** programma python contenente
   - le funzioni che risolvono i tre esercizi
   - nient'altro
   - il file deve chiamarsi `lab02.py`
2. scrivete le vostre funzioni nel file `lab02.py`
3. scaricate il file test `test_lab02.py`
4. eseguite, nella cartella che contiene entrambi,

```
$ python3 test_lab02.py
```

5. migliorate fino a che non ottenete una cosa **COME**

```
........
----------------------------------------------------
Ran 8 tests in 0.001s

OK
```

Esercizio 1

Scrivere una funzione

```
scontato(prezzo,sconto)
```

che, dato un prezzo ed uno sconto da 0 a 100, **restituisca** il prezzo scontato. Ad esempio se il prezzo è 500 e lo sconto è 20 (che vuol dire 20 percento), allora il prezzo finale restituito è 400.

Aggiornamento: la funzione deve controllare i valori passati come parametri `prezzo` e `sconto` abbiano senso. In caso contrario dovrà sollevare `ValueError`.

Esercizio 2

```
superficie_cilindro(raggio,altezza)
```

Essenzialmente una reimplementazione dell'esempio visto in classe. Dati l'altezza del cilindro ed il raggio della base, la funzione deve **restituire** la superficie del cilindro. E si deve utilizzare l'approssimazione 'math.pi'.

Aggiornamento: la funzione deve controllare i valori passati come parametri `raggio` e `altezza` abbiano senso. In caso contrario dovrà sollevare `ValueError`.

Esercizio 3

```
volume_cilindro(raggio,altezza)
```

Dati l'altezza del cilindro ed il raggio della base, la funzione deve **restituire** il volume del cilindro. E si deve utilizzare l'approssimazione 'math.pi'.

Aggiornamento: la funzione deve controllare i valori passati come parametri `raggio` e `altezza` abbiano senso. In caso contrario dovrà sollevare `ValueError`.

Se la funzione non solleva un errore

Un test può verificare che per dei valori scorretti venga sollevato l'errore richiesto. Se questo non avviene il test stampa delle informazioni.

Se per esempio la funzione `superficie_cilindro` non solleva l'errore `ValueError` come previsto dal test, l'output può essere una cosa simile a

```
======================================================================
FAIL: test_superficie_cilindro_raggio (__main__.TestLab02Geometria)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test_lab02.py", line 123, in test_superficie_cilindro_raggio
    self.assertRaises(ValueError,superficie_cilindro,-10,0)
AssertionError: ValueError not raised by superficie_cilindro
```

l'ultima riga vi dice cosa è successo.

Esercizio 4

```
ghms(secondi)
```

Scrivere una funzione che prende in input un numero di secondi (intero) e restituisce una stringa con l'equivalente in giorni, ore, minuti e secondi. Ad esempio se secondi=5000 allora la funzione deve restituire la stringa

```
'Giorni: 0 - Ore: 1 - Minuti: 23 - Secondi: 40'
```

Attenti alla formattazione della stringa. No a capo, spazi giusti ecc…

Esercizio 5

```
totale_secondi(gg,hh,mm,ss)
```

La funzione ha in input un certo numero di giorni, ore, minuti e secondi, e deve restituire il totale dei secondi che costituiscono l'intero lasso di tempo.

Ad esempio `totale_secondi(2,14,27,12)` deve restituire `224832`

Costruzione di stringhe (e sequenze)

Metodo più semplice: concatenazione

Date due o più sequenze `seq1`, `seq2`, `seq3`, … **dello stesso tipo** è possibile concatenarle in una **nuova** sequenza.

```
x = (1,2,3,4)+ ('uno','due','tre','quattro')
print(x)

y = ['A','B','C'] + [1,2,3] + ['do','re','mi']
print(y)

z = 'Vince ' + str(10000) + ' dollari: ' + 'Gastone Paperone!'
print(z)
```

```
(1, 2, 3, 4, 'uno', 'due', 'tre', 'quattro')
['A', 'B', 'C', 1, 2, 3, 'do', 're', 'mi']
Vince 10000 dollari: Gastone Paperone!
```