Esercizi in Laboratorio

Informatica@SEFA 2017/2018 - Laboratorio 2

Massimo Lauria <massimo.lauria@uniroma1.it>
http://massimolauria.net/courses/infosefa2017/

Lunedì, 9 Ottobre 2017

Errata corrige (tuple e liste)

Immutabilità, tuple e liste

La scorsa lezione vi ho detto una **sciocchezza**

```
tupla = (1,2,[3,4])
tupla[0] = "mod"
tupla[-1][0] = "mod"
print(tupla)
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
(1, 2, ['mod', 4])
```

Identità

La funzione `id()` associa un numero ad ogni oggetto.

Due oggetti che esistono simultaneamente non hanno mai lo stesso numero.

```
print(id(4))                                    1
lista1 = [1,2,4]                                2
lista2 = [1,2,4]                                3
print(id(lista1))                               4
print(id(lista2))                               5
lista1.append(6)                                6
print(lista1)                                   7
print(id(lista1))                               8
```

```
4404862416
4408445640
4408445512
[1, 2, 4, 6]
4408445640
```

id(), tuple,stringhe e liste

Operatore x+=y su sequenze le estende.

```
lista    = [1,2,3]                                                    1
tupla    = (1,2,3)                                                    2
stringa  = "abc"                                                      3
print("L:", id(lista)," T:", id(tupla), " S:", id(stringa))           4
                                                                      5
lista   += [4,5,6]                                                    6
tupla   += (4,5,6)                                                    7
stringa += "def"                                                      8
print("L:", id(lista)," T:", id(tupla), " S:", id(stringa))           9
                                                                      10
lista   = lista    + [7,8,9]                                          11
tupla   = tupla    + (7,8,9)                                          12
stringa = stringa  + "ghi"                                            13
print("L:", id(lista)," T:", id(tupla), " S:", id(stringa))           14
```

```
L: 4524888776  T: 4524729184  S: 4523204256
L: 4524888776  T: 4524563528  S: 4524883392
L: 4524898632  T: 4524484432  S: 4524906992
```

Ma la tupla non è 'immutable'?

Sì, ma l'immutabilità riguarda l'identità degli oggetti e non il loro valore.

```
tupla = (1,2,[3,4])                                    1
print(id(tupla[0]),id(tupla[1]),id(tupla[2]))          2
tupla[-1][0] = "mod"                                   3
print(id(tupla[0]),id(tupla[1]),id(tupla[2]))          4
```

```
4359609712 4359609744 4363209416
4359609712 4359609744 4363209416
```

L’identità degli elementi della tupla non cambia.

Ma cambia il valore! Quindi la tupla in effetti non può essere usata, ad esempio, come chiave dei database.

per rispondere anticipo un po’ la sintassi dei **dizionari**

```
database = {}                                          1
database[(1,2,(3,4),5)] = "prima prova"                2
print( database[(1,2) + ((3,4),5)])                    3
database[(1,2,[3,4],5)] = "seconda prova"              4
```

Prompt e Linea di comando

Prompt e linea di comando

Quando lanciate il terminale vi trovate davanti all'interprete dei comandi per lavorare su file.

Su Mac e Linux

```
blabla@bla:~/$
```

Su Windows

```
C:\Users>
```

File e cartelle

I comandi `ls` (Mac/Linux) e `dir` (Windows) vi dicono che file ci sono nella **cartella corrente**.

Ci si può muovere tra le cartelle con il comando `cd` (Change Directory).

```
massimo@lauria:~$

massimo@lauria:~$ ls
Applications      Dropbox           Pictures          personal
Desktop           Library           Public            setup_anaconda.sh
Documents         Movies            config
Downloads         Music             lavori

massimo@lauria:~$ cd Documents
massimo@lauria:~/Documents$
```

Piccolo tutorial sulla linea di comando

Un piccolo tutorial su come fare le operazioni di base e come muoversi tra cartelle.

Django Girls Tutorial (italiano)

- muoversi tra le cartelle
- copiare, muovere, rinominare, cancellare file

Eseguire python

Per lanciare l'interprete interattivo

```
blabla@bla:~/$ python3
```

```
blabla@bla:~/$ ipython3
```

Per eseguire un programma

```
blabla@bla:~/$ ipython3 nome_file.py
```

Interfaccia testuale S.O. vs Python

```
lauria@macbook15: ~/$ cd Documenti

lauria@macbook15: ~/Documenti$ cd ..

lauria@macbook15: ~/$
```

Interfaccia testuale S.O. vs Python

```
lauria@macbook15: ~/$ cd Documenti

lauria@macbook15: ~/Documenti$ cd ..

lauria@macbook15: ~/$ python3
```

```
>>>
```

Interfaccia testuale S.O. vs Python

```
lauria@macbook15: ~/$ cd Documenti

lauria@macbook15: ~/Documenti$ cd ..

lauria@macbook15: ~/$ python3
```

```
>>> print(5+ 0.2)
5.2

>>> exit() # oppure premo Ctrl-D
```

Interfaccia testuale S.O. vs Python

```
lauria@macbook15: ~/$ cd Documenti

lauria@macbook15: ~/Documenti$ cd ..

lauria@macbook15: ~/$ python3
```

```
>>> print(5+ 0.2)
5.2

>>> exit() # oppure premo Ctrl-D
```

```
lauria@macbook15: ~/$
```

Interfaccia testuale S.O. vs Python

```
lauria@macbook15: ~/$ cd Documenti

lauria@macbook15: ~/Documenti$ cd ..

lauria@macbook15: ~/$ python3
```

```
>>> print(5+ 0.2)
5.2

>>> exit() # oppure premo Ctrl-D
```

```
lauria@macbook15: ~/$ python3 nomeprogramma.py
```

Interfaccia testuale S.O. vs Python

```
lauria@macbook15: ~/$ cd Documenti

lauria@macbook15: ~/Documenti$ cd ..

lauria@macbook15: ~/$ python3
```

```
>>> print(5+ 0.2)
5.2

>>> exit() # oppure premo Ctrl-D
```

```
lauria@macbook15: ~/$ python3 nomeprogramma.py

 blah blah blah output del programma blah blah
 blah blah blah output del programma blah blah
 blah blah blah output del programma blah blah

lauria@macbook15: ~/$
```

Prompt, terminale e Python

Interfaccia testuale di Mac/Linux

```
blabla@bla:~/$
```

Interfaccia testuale di Windows

```
C:\Users>
```

Python3

```
>>>
```

IPython3

```
In[12]:
```

Convenzioni per le slide

Prompt del terminale

```
$
```

Codice python

```python
if 0 < 1:
    print('ovvio')
else:
    print('irrangiungibile')
```

Output del programma

```
ovvio
```

Esercizi

Segnalare errori

```
def area_rettangolo(base,altezza):                                    1
    """Calcola l'area di un rettangolo                                2
                                                                      3
    Calcola l'area di un rettangolo 'base' e 'altezza'                4
    specificati nei
    paramentri. Se uno di essi è negativo solleva l'eccezione         5
    'ValueError'.                                                     6
    """                                                               7
    if base < 0 or altezza < 0:                                       8
        raise ValueError("valore negativo per base o altezza "        9
    )
    return altezza*base                                               10
                                                                      11
print( area_rettangolo(2,-4) )                                        12
print( area_rettangolo(2, 4) )                                        13
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 9, in area_rettangolo
ValueError: valore negativo per base o altezza
8
```

Testare le soluzioni

1. Scrivere `lab02.py`, modulo che contenga tutte le funzioni richieste.
2. Scaricare `test_lab02.py` e metterlo nella stessa cartella
3. Eseguire, nella cartella che contiene entrambi,

```
$ python3 test_lab02.py
```

Esercizio 8

Migliorare tutte le funzioni della volta scorsa

- controllare se gli input sono corretti
- se non lo sono sollevare l'eccezione `ValueError`

Esercizio 9

Costruire una funzione

```
ghms2(secondi)
```

simile a quella di `lab01`, ma che produca stringhe più sensate. Ad esempio.

input	output
0	`0 secondi.`
2348	`39 minuti e 8 secondi.`
3840	`1 ora e 4 minuti.`
122456	`1 giorno, 10 ore e 56 secondi.`

- attenzione ai plurali e singolari.
- attenzione alla punteggiatura e all'uso di 'e'
- controllare la correttezza degli input
- fate un bel respiro e aiutatevi con il file di test

Esercizio 10

```
ordinati(lista)
```

Prende in input una sequenza di elementi e

- solleva `ValueError` se nella lista ci sono sia numeri che stringhe
- restituisce `True` se sono ordinati dal più basso al più alto
- restituisce `False` se non sono ordinati

Soluzione Esercizi `lab01`

Esercizio 1

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

```
def scontato(prezzo,sconto):                    1
    return prezzo*(100-sconto)/100              2
```

Esercizio 2

```
area_cilindro(raggio,altezza)
```

```
import math                                                    1
def area_cilindro(raggio,altezza):                             2
    area = 2*math.pi*raggio*altezza + 2 * math.pi * raggio**2  3
    return area                                                4
```

Esercizio 4

```
area_parallelepipedo_rettangolo(altezza,larghezza,profondità)
```

```
def area_parallelepipedo_rettangolo(altezza,larghezza,          1
    profondità):
    faccia1 = altezza * larghezza                               2
    faccia2 = altezza * profondità                              3
    faccia3 = larghezza * profondità                            4
    return 2*(faccia1 + faccia2 + faccia3)                      5
```

Esercizio 3

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

```
import math                                                    1
def volume_cilindro(raggio,altezza):                           2
    return altezza * raggio**2 * math.pi                       3
```

Esercizio 5

```
volume_parallelepipedo_rettangolo(altezza,larghezza,profondità)
```

```
def volume_parallelepipedo_rettangolo(altezza,larghezza,      1
    profondità):
    return altezza*larghezza*profondità                       2
```

Esercizio 6

`ghms(secondi)`

```
def ghms(secondi):                                                    1
    sec_in_min = 60                                                   2
    sec_in_ora = sec_in_min * 60                                      3
    sec_in_giorno = sec_in_ora * 24                                   4
                                                                      5
    giorni = secondi // sec_in_giorno                                 6
    secondi %= sec_in_giorno                                          7
                                                                      8
    ore = secondi // sec_in_ora                                       9
    secondi %= sec_in_ora                                             10
                                                                      11
    minuti = secondi // sec_in_min                                    12
    secondi %= sec_in_min                                             13
                                                                      14
    return 'Giorni: ' + str(giorni) +' - Ore: ' + str(ore) \          15
        +' - Minuti: ' + str(minuti) +' - Secondi: '+str(             16
    secondi)
```

Esercizio 7

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

```
def totale_secondi(gg,hh,mm,ss):                                    1
    sec_in_min = 60                                                 2
    sec_in_ora = sec_in_min * 60                                    3
    sec_in_giorno = sec_in_ora * 24                                 4
    return gg*sec_in_giorno + hh*sec_in_ora + mm*sec_in_min +        5
    ss
```