Esercitazione

Informatica@SEFA 2018/2019 - Laboratorio 4

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

Lunedì, 29 Ottobre 2018

Sequence slicing (I)

```
colori = ("giallo","verde","blu","rosso","viola","marrone")

print( colori[1] )

print( colori[2:5] )

print( colori[-4:6] )

print( colori[3:-2] )

print( colori[2:1] )

print( colori[2:2] )
```

```
verde
('blu', 'rosso', 'viola')
('blu', 'rosso', 'viola', 'marrone')
('rosso',)
()
()
```

```
colori = ["giallo","verde","blu","rosso","viola","marrone"]   1
                                                              2
L =[]                                                         3
for i in range(2,5):                                          4
    L.append(colori[i])                                       5
print( L )                                                    6
print( colori[2:5]  )                                         7
```

```
['blu', 'rosso', 'viola']
['blu', 'rosso', 'viola']
```

```
colori = ["giallo","verde","blu","rosso","viola","marrone"]   1
                                                              2
L =[]                                                         3
for i in range(1,-2):                                         4
    L.append(colori[i])                                       5
print( L )                                                    6
print( colori[1:-2]  )                                        7
```

```
[]
['verde', 'blu', 'rosso']
```

Esercitazione

1. scrivere **un** programma python contenente
   - le funzioni che risolvono gli esercizi
   - nient'altro
   - il file deve chiamarsi `lab04.py`
2. scrivete le vostre funzioni nel file `lab04.py`
3. scaricate il file test `test_lab04.py`
4. eseguite, nella cartella che contiene entrambi,

```
$ python3 test_lab04.py
```

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

```
.....................
----------------------------------------------------------
Ran 23 tests in 0.005s

OK
```

Esercizio 8

Costruire una funzione

```
segmenticrescenti(seq)
```

che data una sequenza in input restituisca una **lista di liste** che deve contenere tutte le sotto-sequenze massimali crescenti contenute in `seq`, in ordine.

Ad esempio:

```
segmenticrescenti([1,-1,2,4,3,7,8,8,5])
```

deve restituire

```
[ [1], [-1,2,4], [3,7,8,8], [5] ]
```

Esercizio 9

```
sommeparziali(seq)
```

Prende in input una sequenza di **numeri** e

$$v_0 v_1 v_2 \dots v_n$$

- solleva `TypeError` se nella sequenza ci sono elementi che non possono sommare.
- restituisce una lista della stessa lunghezza di `seq` dove alla posizione i si ha il valore

$$\sum_{j=0}^{i} v_i$$