Esercitazione di Laboratorio Informatica@DSS (2023/2024)

Massimo Lauria

Laboratorio 7 - 13/11/2023

Lo scopo del laboratorio è di esercitarsi e misurare la propria preparazione: gli esercizi non sono troppo difficili, se si sono seguite le lezioni. Non vi viene comunque messo alcun voto.

Modalità di lavoro: gli studenti devono lavorare in autonomia o in piccoli gruppi, senza disturbare i colleghi. Il lavoro di gruppo è fruttuoso solo se tutti partecipano e se ognuno scrive una propria soluzione per tutti gli esercizi.

Il docente cercherà per quanto possibile di non occupare il tempo del laboratorio per introdurre materiale nuovo, anche se a volte questo sarà necessario. Il docente è a disposizione per aiutare gli studenti, che possono iniziare a lavorare anche prima che il docente arrivi in aula, se lo desiderano

Raccomandazioni: leggete bene il testo degli esercizi prima di chiedere chiarimenti. In ogni caso sarò in aula con voi.

Uso dei file di test: per aiutarvi a completare questa esercitazione avete a disposizione dei programmi di test per testare la vostra soluzione. Questi sono simili a quelli che avrete in sede di esame, pertanto vi consiglio di imparate ad usarli. Per portare a termine l'esercizio è necessario

- scrivere un file di soluzione **col nome specificato**;
- avere dentro la funzione **col nome specificato**;
- porre il file di test corrispondente **nella stessa cartella**;
- eseguire in quella cartella il comando `python3 <fileditest>`

dove `<fileditest>` va ovviamente sostituito con il nome del file di test appropriato per l'esercizio su cui state lavorando. Per ogni esercizio ci sta un file di test indipendente, così da poter lavorare sugli esercizi uno alla volta con più agio.

Il risultato di ogni test è una schermata (o più schermate) nella quale si mostra:

- se è stato trovato il file con il nome corretto,
- se il file contiene la funzione con il nome corretto,
- se chiamate alla funzione con diversi valori dei parametri terminano restituendo risultati corretti.

Per ogni funzione scritta vengono eseguite chiamate con diversi valori dei parametri. L'esito dei test viene riportato con il carattere

- `E` se la chiamata non può essere eseguita,
- `F` se la funzione non restituisce il risultato corretto,
- `.` se la funzione restituisce il risultato corretto.

1 Lista di numeri

Scrivete una funzione `crealista1(a, b)` che:

- si aspetti come argomenti due numeri `a` e `b`, con `b` che deve essere un numero intero non negativo;
- crei e restituisca una lista contenente i primi `b` interi a partire da `a`.

Ad esempio, l'istruzione `s = crealista1(14.3, 5)` deve assegnare a `s` la lista `[15, 16, 17, 18, 19]`.

```
s = crealista1(14.3, 5)
print(s)
s = crealista1(14, 7)
print(s)
```

```
[15, 16, 17, 18, 19]
[14, 15, 16, 17, 18, 19, 20]
```

Il programma Python deve essere salvato nel file: `crealista1.py`

File di test: `test_crealista1.py`

2 Azzerare elementi negativi

Scrivere una funzione `azzeranegativi(valori)` che:

- si aspetti come argomento una lista valori numerici (`int` o `float`)
- modifichi la lista ricevuta, azzerando tutti gli elementi negativi.

La funzione non deve restituire nulla (ovvero restituisce `None`), ma invece deve modificare la lista passata come argomento. In particolare è possibile scrivere questa funzione senza utilizzare l'istruzione `return`. Guardate nell'esempio successivo come si deve comportare la funzione

```
L = [1.2, -0.4, 5, -2, 12 ]

print(azzeranegativi(L))
print(L)
```

```
None
[1.2, 0, 5, 0, 12]
```

Il programma Python deve essere salvato nel file: `azzeranegativi.py`

File di test: `test_azzeranegativi.py`

3 Verificare se una sequenza è bitonica

Scrivere una funzione `bitonica(L)` che:

- si aspetti come argomento una lista `L`;
- restituisca `True` se la sequenza è **bitonica**, e `False` altrimenti.

Per quanto riguarda il nostro esercizio, una sequenza è considerata bitonica se è composta da una prima parte **strettamente** crescente con almeno due elementi, seguita da una parte **strettamente** decrescente, anche questa composta da almeno due elementi. Ad esempio sono bitoniche le seguenti sequenze:

- `2 4 13 34 23`
- `3 4 1`
- `27 48 113 134 23 12`

Osservate che, come negli esempi, l'ultimo elemento della parte crescente è anche il primo della parte decrescente. Nessuna sequenza bitonica può avere quindi meno di tre elementi.

La lista

- `2 4 6 6 3 -2`

non è bitonica perché il segmento `6 6` non è strettamente né crescente né decrescente.

```
print(bitonica([2,4,13,34,23]))
print(bitonica([3,4,1]))
print(bitonica([27,48,113,134,23,12]))
print(bitonica([2,4,6,6,3,-2]))
```

```
True
True
True
False
```

Il programma Python deve essere salvato nel file: `bitonica.py`

File di test: `test_bitonica.py`

4 Massimi locali

Scrivere una funzione `massimilocali(L)` che:

- si aspetti come argomento una lista `L`;
- crei e restituisca una lista contenente i massimi locali presenti in sequenza.

Un elemento di una sequenza è un massimo locale se:

- non si trova né in prima né in ultima posizione;
- è preceduto e seguito da elementi **strettamente** minori dell'elemento stesso.

Una lista contenente meno di tre elementi non contiene alcun massimo locale, quindi in questo caso deve essere restituita una lista vuota.

```
s = massimilocali([3, 6, 4, 4, 7, 5,8])
print(s)
```

```
[6, 7]
```

Il programma Python deve essere salvato nel file: `massimilocali.py`

File di test: `test_massimilocali.py`

5 Lista di numeri (2)

Scrivere una funzione `crealista2(a, b, passo=1)`, con un parametro opzionale, che si comporti come la funzione al punto precedente `crealista1` nel caso venga invocata con due argomenti, mentre nel caso di invocazione con tre argomenti crei e restituisca una lista di `b` interi a partire dal primo intero **maggiore o uguale** ad `a` con passo uguale al terzo argomento. Ad esempio, l'istruzione

```
s = crealista2(14.6, 5, 3)
```

deve assegnare a `s` la lista `[15, 18, 21, 24, 27]`

Il terzo parametro può essere un intero qualunque. Ad esempio `crealista2(14.6, 5, -3)` deve restituire la lista `[15, 12, 9, 6, 3]`.

Il programma Python deve essere salvato nel file: `crealista2.py`

File di test: `test_crealista2.py`