Laboratorio 6

Informatica@DSS 2019/2020 — (E-N)

Massimo Lauria <massimo.lauria@uniroma1.it>*

Lunedì, 4 Novembre 2019

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. Alla fine della lezione per favore rispondete al questionario, disponibile al link alla fine di questo documento.

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**;

*http://massimolauria.net/courses/informatica2019/

- 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. Controllate i parametri passati in input ed eventualmente sollevate le eccezioni `ValueError` o `TypeError`.

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 Somma di liste

Scrivere una funzione `somma_liste(a, b)` che:

- riceve come parametri due liste `a`, `b` della stessa lunghezza, in cui ciascun elemento è un numero `int`;
- solleva un errore `TypeError` se i parametri ricevuti non sono liste, o se gli elementi delle liste non sono tutti numeri interi;
- solleva un errore di tipo `ValueError` se le liste non hanno la stessa lunghezza;
- crea e restituisce una lista, della stessa lunghezza di quelle ricevute come parametri, in cui l'elemento i-esimo della lista è uguale alla somma dell'elemento i-esimo di `a` e dell'elemento i-esimo di `b`

File della soluzione: `lab06sommaliste.py`

File di test: `test_lab06sommaliste.py`

2 Prodotto Scalare

Le liste sono ovviamente un modo naturale per rappresentare il concetto matematico di vettore. Scrivere una funzione `prodotto_scalare(a, b)` che:

- riceve come parametri due liste non vuote `a`, `b` della stessa lunghezza, in cui ciascun elemento è un numero (`int` oppure `float`);
- solleva un errore `TypeError` se i parametri ricevuti non sono liste, o gli elementi delle liste non sono tutti numeri;
- solleva un errore di tipo `ValueError` se le liste non hanno la stessa lunghezza oppure sono liste vuote;
- restituisce il prodotto scalare delle due liste. Il prodotto scalare tra due vettori NON È UN VETTORE, bensì un numero: il prodotto scalare è la somma dei prodotti degli elementi omologhi: ad esempio, il prodotto scalare tra i vettori $(1, 4, 3)$ e $(5, 5, 1)$ è $1*5+4*5+3*1 = 28$.

File della soluzione: `lab06prodottoscalare.py`

File di test: `test_lab06prodottoscalare.py`

3 Separa elementi

Scrivere una funzione `separa_elementi(valori)` che:

- riceve come parametro una lista `valori`, in cui ciascun elemento è un numero (`int` oppure `float`);
- solleva un errore `TypeError` se il parametro ricevuto non è una lista, o se gli elementi della lista non sono tutti numeri;
- restituisce due liste (notare che è ammessa l'istruzione `return lista1, lista2`). La prima lista deve contenere tutti gli elementi di valori che sono maggiori della media degli elementi di valori, mentre la seconda lista deve contenere tutti gli elementi di valori che sono

minori o uguali alla media degli elementi di valori. Ad esempio, se la lista contenesse i valori `[0.96, 0.2, 0.4, 0.1, 0.55, 0.03, 0.88]`, poiché la media degli elementi è `0.4457`, si dovrebbero restituire le liste `[0.96, 0.55, 0.88]` e `[0.2, 0.4, 0.1, 0.03]` nel caso di una lista vuota, la funzione deve restituire una coppia di liste vuote (non ha senso calcolare la media dei valori)

File della soluzione: `lab06separa.py`

File di test: `test_lab06separa.py`

4 Intersezione

Scrivere una funzione `intersezione(a, b)` che:

- riceve come parametri due liste `a`, `b`;
- solleva un errore `TypeError` se i parametri ricevuti non sono liste;
- restituisce una lista che rappresenta l'intersezione tra le due liste, cioè gli elementi che appartengono sia alla lista `a` che alla lista `b`. Si può assumere che a non contenga elementi duplicati e che `b` non contenga elementi duplicati.

File della soluzione: `lab06intersezione.py`

File di test: `test_lab06intersezione.py`

5 Unione

Scrivere una funzione `unione(a, b)` che:

- riceve come parametri due liste `a`, `b`;
- solleva un errore `TypeError` se i parametri ricevuti non sono liste;
- restituisce una lista che rappresenta l'unione tra le due liste, cioè gli elementi che appartengono alla lista `a` e/o alla lista `b`. Si può assumere che `a` non contenga elementi duplicati e che `b` non contenga elementi duplicati.

File della soluzione: `lab06unione.py`

File di test: `test_lab06unione.py`

Questionario: rispondete **dopo** aver fatto gli esercizi.

http://bit.ly/INFO2019-LAB06