Lezione 26 - Laboratorio

Informatica@DSS 2021/2022

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

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**;
- eseguire in quella cartella il comando

*http://massimolauria.net/informatica2021/

```
$ 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.

Questi esercizi riguardano la rappresentazione di matrici di numeri in Python attraverso liste di liste di numeri. Ad esempio una matrice

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$$

viene rappresentata in Python come la lista

```
[[1,2,3], [4,5,6], [7,8,9], [10,11,12]]
```

In generale una matrice $r \times c$ è rappresentata da una lista contenente a sua volta r liste (che rappresentano le singole righe), e ognuna delle quali deve avere lunghezza c. Imponiamo la condizione che r e c siano strettamente maggiori di 0.

Quindi per noi e per gli esercizi che seguono una **matrice ben formata** è una

- lista di liste, e ognuna delle quali rappresenta una riga;
- ogni riga deve avere la stessa lunghezza (che sarà c);
- ogni matrice deve avere almeno una riga e una colonna.

Vi conviene scrivere una fuzione che esegua questi controlli, da riutilizzare nei vari esercizi.

Stampare una matrice

Scrivere una funzione `stampa_mat(m)` che:

- riceva come parametro una matrice `m`;
- sollevi un'eccezione `TypeError` se l'argomento passato non è una matrice ben formata;
- stampi la matrice `m`, sotto forma di tabella, andando a capo alla fine di ciascuna riga. Ad esempio per una matrice

  ```
  [ [2, 4, 3, 6, 8], [3, 1, 4, 0, 0], [1, 5, 1, 7, 4] ]
  ```

  deve stampare

  ```
  2 4 3 6 8
  3 1 4 0 0
  1 5 1 7 4
  ```

- la funzione non deve restituire nulla.

Per questo esercizio non viene fornito un file di test. Provare a chiamare la funzione con argomenti diversi e verificare l'aspetto della stampa.

1 Somma degli elementi di una matrice numerica

Scrivere una funzione `somma_mat(m)` che:

- riceva come parametro una matrice `m`;
- sollevi un'eccezione `TypeError` se l'argomento passato non è una matrice ben formata;

- sollevi un'eccezione `TypeError` se gli elementi della matrice non sono numeri;
- restituisca la somma di tutti gli elementi della matrice.

File della soluzione: `somma_mat.py`

File di test: `test_somma_mat.py`

2 Minimo degli elementi di una matrice numerica

Scrivere una funzione `min_mat(m)` che:

- riceva come parametro una matrice `m`;
- sollevi un'eccezione `TypeError` se l'argomento passato non è una matrice ben formata;
- sollevi un'eccezione `TypeError` se gli elementi della matrice non sono numeri;
- restituisca il valore dell'elemento minimo nella matrice.

File della soluzione: `min_mat.py`

File di test: `test_min_mat.py`

3 Posizione del massimo elemento di una matrice numerica

Scrivere una funzione `posmax_mat(m)` che:

- riceva come parametro una matrice `m`;
- sollevi un'eccezione `TypeError` se l'argomento passato non è una matrice ben formata;
- sollevi un'eccezione `TypeError` se gli elementi della matrice non sono numeri;
- restituisca una coppia di interi (ad esempio con un `return i, j`) che siano rispettivamente l'indice di riga e di colonna in cui compare il massimo elemento della matrice. In caso di più massimi, deve essere

restituita la posizione del primo di essi (minima riga e, in caso di più massimi sulla riga, minima colonna).

File della soluzione: `posmax_mat.py`

File di test: `test_posmax_mat.py`

4 Somma degli elementi della diagonale di una matrice numerica

Scrivere una funzione `sommadiagonale_mat(m)` che:

- riceva come parametro una matrice `m`;
- sollevi un'eccezione `TypeError` se l'argomento passato non è una matrice ben formata;
- sollevi un'eccezione `TypeError` se gli elementi della matrice non sono numeri;
- sollevi un'eccezione `ValueError` se la matrice non è quadrata, ovvero se il numero di righe è diverso dal numero di colonne;
- restituisce la somma degli elementi sulla diagonale principale. Notare che per una matrice con `n` righe ed `n` colonne si devono sommare solo `n` elementi.

File della soluzione: `sommadiagonale_mat.py`

File di test: `test_sommadiagonale_mat.py`

5 Somme per riga di una matrice numerica

Scrivere una funzione `somme_per_riga_mat(m)` che:

- riceva come parametro una matrice `m`;
- sollevi un'eccezione `TypeError` se l'argomento passato non è una matrice ben formata;
- sollevi un'eccezione `TypeError` se gli elementi della matrice non sono numeri;

- restituisca una lista in cui l'elemento di posto `i` è la somma degli elementi della riga `i`-esima. La lunghezza di tale lista è uguale al numero di righe della matrice.

File della soluzione: `somme_per_riga_mat.py`

File di test: `test_somme_per_riga_mat.py`

6 Somme per colonna di una matrice numerica

Scrivere una funzione `somme_per_colonna_mat(m)` che:

- riceva come parametro una matrice `m`;
- sollevi un'eccezione `TypeError` se l'argomento passato non è una matrice ben formata;
- sollevi un'eccezione `TypeError` se gli elementi della matrice non sono numeri;
- restituisca una lista in cui l'elemento di posto `i` è la somma degli elementi della colonna `i`-esima. La lunghezza di tale lista è uguale al numero di colonne della matrice.

File della soluzione: `somme_per_colonna_mat.py`

File di test: `test_somme_per_colonna_mat.py`

7 Posizione della riga con somma massima

Scrivere una funzione `pos_maxriga_mat(m)` che:

- riceva come parametro una matrice `m`;
- sollevi un'eccezione `TypeError` se l'argomento passato non è una matrice ben formata;
- sollevi un'eccezione `TypeError` se gli elementi della matrice non sono numeri;
- restituisca la posizione della riga che ha massima somma. Ad esempio, se viene chiamata con parametro

```
[ [2, 4, 3, 6, 8], [3, 1, 4, 0, 0], [9, 5, 9, 7, 4] ]
```

deve restituire il valore 2, poiché la riga in posizione 2 ha somma 34, mentre le righe in posizione 0 e 1 hanno rispettivamente somma 23 e 8.

File della soluzione: `pos_maxriga_mat.py`

File di test: `test_pos_maxriga_mat.py`