Lezione 29 - 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.

1 Calcolo delle medie mobili

Scrivere una funzione `medie_mobili(seq, k)` che:

- si aspetti come argomenti una lista di numeri `seq` e un numero `k`;
- sollevi un'eccezione `TypeError` se il primo argomento non è una lista, se il secondo argomento non è un numero intero o se gli elementi della lista non sono numeri;
- sollevi un'eccezione `ValueError` se la sequenza ha lunghezza 0;
- sollevi un'eccezione `ValueError` se non si verifica che `0<k<=len(seq)`;
- nei casi rimanenti la funzione deve costruire e restituire la lista delle `len(seq)-k+1` medie mobili semplici di ordine `k`.

Una media mobile semplice di ordine `k` è la media aritmetica di `k` elementi consecutivi nella sequenza, quindi se la sequenza ha lunghezza `n` si

possono calcolare `n-k+1` medie mobili: dalla porzione di sequenza `[0:k]` alla porzione di sequenza `[n-k:n]` (estremo iniziale incluso, estremo finale escluso). Notare che il numero di medie mobili calcolate dipende dalla lunghezza della sequenza e dal valore di `k`.

Ad esempio, se la funzione `medie_mobili(seq, k)` viene chiamata con parametri `seq = [2, 4, 3, 6, 8, 9, 10, 12]` e `k = 4` deve essere restituita la lista `[3.75, 5.25, 6.5, 8.25, 9.75]`, che rappresenta le medie delle 5 sottosequenze `[2, 4, 3, 6]`, `[4, 3, 6, 8]`, `[3, 6, 8, 9]`, `[6, 8, 9, 10]`, `[8, 9, 10, 12]`.

File della soluzione: `medie_mobili.py`

File di test: `test_medie_mobili.py`

2 Posizione della massima media mobile

Scrivere una funzione `posmax_medie_mobili(seq, k)` che, utilizzando anche la funzione definita nell'esercizio precedente:

- si aspetti come argomenti una lista di numeri `seq` e un numero `k`;
- sollevi un'eccezione `TypeError` se il primo argomento non è una lista, se il secondo argomento non è un numero intero o se gli elementi della lista non sono numeri;
- sollevi un'eccezione `ValueError` se la sequenza ha lunghezza 0;
- sollevi un'eccezione `ValueError` se non si verifica che `0<k<=len(seq)`;
- restituisca la posizione della massima media mobile di ordine `k` della sequenza `seq` (vedi definizione nell'esercizio precedente).

Ad esempio, se viene chiamata con parametri `[2, 4, 3, 6, 8, 9, 10, 12], 4` deve essere restituito il valore 4, poiché la sequenza delle medie mobili è `[3.75, 5.25, 6.5, 8.25, 9.75]` e il massimo compare in posizione 4.

File della soluzione: `posmax_medie_mobili.py`

File di test: `test_posmax_medie_mobili.py`

3 Matrice trasposta

Scrivere una funzione `trasposta_mat(m)` che:

- si aspetti come argomento matrice `m`;
- sollevi un'eccezione `TypeError` se l'argomento passato non è una matrice ben formata;
- crei e restituisca la matrice trasposta di `m`. Qui è utile prima creare la matrice come visto a lezione, per poi modificarne gli elementi in modo da avere la trasposta di `m`.

Ad esempio, se viene passata come parametro la lista `[[2, 3, 7], [5, 3, 1]]` deve essere restituita la lista `[[2, 5], [3, 3], [7, 1]]`.

File della soluzione: `trasposta_mat.py`

File di test: `test_trasposta_mat.py`

4 Punto di sella di una matrice

Un elemento `M[i][j]` di una matrice m si dice punto di sella se è strettamente maggiore di tutti gli altri elementi sulla riga `i` ed è strettamente minore di tutti gli altri elementi sulla colonna `j`. Notare che una matrice contiene al massimo un solo punto di sella.

Ad esempio, la matrice

```
9   4  16  10   4
2   5   7   6   2
8  12  13   9   1
```

ha un punto di sella a riga 1 e colonna 2 (il valore 7) poiché 7 è il massimo della riga 1 ed è anche il minimo della colonna 2.

Scrivere una funzione `sella_mat(M)` che:

- si aspetti come parametro una matrice `M`;
- sollevi un'eccezione `TypeError` se l'argomento passato non è una matrice ben formata oppure contiene dati non numerici;
- restituisca la coppia di indici `(i, j)` del punto di sella, se esiste, e restituisca `None` se `M` non ha un punto di sella.

File della soluzione: `sella_mat.py`

File di test: `test_sella_mat.py`

5 Scomposizione in secondi (seconda versione)

`ghms2(secondi)`

Scrivere una funzione `ghms2(secondi)` molto simile a quella `ghms` di un esercizio precedente, ma che produca stringhe più sensate. La funzione deve

- aspettarsi come argomento un numero intero `secondi`;
- deve sollevare un'eccezione `TypeError` se l'argomento non è un numero intero;
- deve sollevare un'eccezione `ValueError` se l'argomento è un numero minore di 0.

Parte dell'esercizio è proprio capire come devono essere costruite queste stringhe a partire dagli esempi che seguono.

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;
- ci sono molti dettagli, fate un bel respiro e aiutatevi con il file di test.

File della soluzione: `ghms2.py`

File di test: `test_ghms2.py`