Esercitazione di Laboratorio Informatica@DSS (2023/2024)

Massimo Lauria

Laboratorio 10 - 4/12/2023

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 Messaggio Segreto

Avete ricevuto un'email con curioso titolo **Urgente: messaggio segreto!** ed essendo una persona prudente avete fatto la cosa più ovvia: avete immediatamente aperto il messaggio. Il messaggio ha diversi file di testo allegati, e recita:

```
La chiave per passare l'esame di informatica è in questi file.

In ognuno di questi file è contenuta una parola segreta, che dovrai
decifrare con le tue sofisticare capacità di programmazione e di
analisi. All'interno di ogni file è codificata una parola scritta
in caratteri  maiuscoli, ho nascosto esattamente un carattere
maiuscolo all'interno di alcune righe del file. Devi ignorare le
righe che non contengono caratteri maiuscoli e ignorare anche quelle
che ne contengono più di uno.

L'unione dei caratteri nascosti fornisce la parola segreta.
```

Se volete trovare la parola segreta di un file, dovete scrivere una funzione `messaggio_segreto(nomefile)` che riceva come argomento il nome di un file di testo. Il programma dovrà estrarre il carattere alfabetico maiuscolo da ogni riga del file che ne contenga **esattamente uno**. La funzione dovrà restituire la stringa costituita dalla concatenazione di questi caratteri segreti.

Ad esempio vediamo il messaggio segreto in `messaggio_segreto1.txt`

```
messaggio_segreto('messaggio_segreto1.txt')
```

```
'INFORMATICA'
```

Le trime righe del file sono

```
asrFuoXzys
alziIedjja
bpzoxNbfji
urWymhXvdm
xjfljsuctg
...
```

e vedere per esempio che nella prima riga ci sono due caratteri maiuscoli, e quindi ai fini del messaggio segreto questa riga va ignorata. Nella seconda riga ci sta solo il carattere maiuscolo `I` e quindi questo va preso in considerazione. Nella quinta riga non c'è nessun carattere maiuscolo quindi neppure questa

riga contribuisce al messaggio segreto.

Quali sono le parole segrete contenute in `messaggio_segreto2.txt` e in `messaggio_segreto3.txt` ?

Hint: ricordate che se avete una lista `L` di stringhe, `"".join(L)` è l'espressione che le concatena tutte.

Hint: il metodo `.isupper()` applicato ad una stringa restituisce `True` solo se la stringa è costituita da caratteri maiuscoli. Il metodo può essere usato naturalmente anche su strighe costituite da un solo carattere.

Il programma deve sollevare l'errore `FileNotFoundError` se il file non esiste.

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

File di test: `test_messaggio_segreto.py`

2 Scomposizione in secondi

Scrivere una funzione `formattazione_dhms(secondi)` che, ricevuto un numero di secondi maggiore e uguale a zero, produca una stringa che esprime il numero di giorni, ore, minuti e secondi secondo l'uso del linguaggio comune. Si vedano gli esempi che seguono per chiarimenti. La funzione deve

- aspettarsi come argomento un numero intero `secondi`;
- 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, aiutatevi con il file di test.

```
print(formattazione_dhms(122456))
```

```
1 giorno, 10 ore e 56 secondi.
```

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

File di test: `test_formattazione_dhms.py`

3 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 `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]` (ricordate che in questi intervalli l'estremo iniziale è incluso, e l'estremo finale è 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]`.

```
medie_mobili([2, 4,  3, 6, 8, 9,  10, 12],4)
```

```
[3.75, 5.25, 6.5, 8.25, 9.75]
```

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

File di test: `test_medie_mobili.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`.

Ad esempio, la matrice

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

ha un punto di sella alla riga 1 e colonna 2, il cui valore è 7. Questo poiché 7 è il massimo della riga 1 ed è anche il minimo della colonna 2. Può essere facilmente dimostrato che una matrice contiene al massimo un solo punto di sella.

Scrivere una funzione `sella_mat(M)` che:

- si aspetti come parametro una matrice `M`;
- restituisca la coppia di indici `(i, j)` del punto di sella, se esiste, e restituisca `None` se `M` non ha un punto di sella.

```
M = [[9, 4, 16, 10, 4], [2, 5, 7, 6, 2], [8, 12, 13, 9, 1]]
1
2
sella_mat(M)
3
```

```
(1, 2)
```

Ricordiamo che in Python noi rappresentiamo una matrice $R \times C$ come una lista di lunghezza R, contenente R liste tutte quante di lunghezza C. Potete sempre assumere che l'input di questo esercizio sia una matrice ben formata, con $R \geq 1$ e $C \geq 1$, e che tutte le liste che rappresentano le righe contengano C valori numerici.

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

File di test: `test_sella_mat.py`