

Esercitazione Python n. 9 -- 26 novembre 2019

Obiettivo dell'esercitazione è esercitarsi nell'utilizzo dei dizionari e nella lettura di file.

All'intero dell'esercitazione vi sono cartelle relative ad ogni singolo esercizio, denominate EsercN, al cui interno sono definiti i file .py e gli eventuali file di input relativi allo specifico esercizio.

I file .py incorporano un codice di test per le vostre funzioni.

Per ogni esercizio, aprite il file .py relativo e modificate SOLO il contenuto della funzione. Eseguendo il file .py si otterrà il responso del test sulla console.

Esercizi

- **Ex1(l):** Scrivere una funzione che riceve in ingresso una lista **l** di stringhe e costruisce il dizionario in cui le stringhe in **l** sono chiavi ed il valore associato a ciascuna stringa è il numero di volte in cui essa compare in **l**. Se la lista **l** in ingresso è vuota, la funzione deve restituire il dizionario vuoto `{}`. Ad esempio, se la **l** vale `['casa', 'orso', 'cane', 'casa', 'orso', 'casa']` allora il dizionario deve valere (l'ordine come al solito per i dizionari non è rilevante):
`{'casa': 3, 'orso': 2, 'cane': 1}`
- **Ex2(s,n):** Scrivere una funzione che riceve in ingresso una stringa **s** composta solo di caratteri alfabetici minuscoli e spazi e un numero **n**, e calcoli il dizionario che ha come chiavi lettere minuscole e come valore il numero di parole di **s** che cominciano per quella lettera e sono lunghe almeno **n** caratteri. Il dizionario NON deve contenere come chiavi le lettere per cui non ci sono parole lunghe almeno **n** caratteri. Se la stringa in ingresso **s** è vuota la funzione deve restituire il dizionario vuoto `{}`. Ad esempio, se **s** vale `'tanto va la gatta al lardo che ci lascia lo zampino'` ed **n** vale 3 la funzione deve restituire:
`{'t': 1, 'g': 1, 'l': 2, 'c': 1, 'z': 1}`
- **Ex3(file)** Scrivere una funzione che prende in ingresso il nome di un **file** di testo csv contenente tutte le partite giocate in un torneo di calcio, nel seguente formato:

Prima riga:

`Nome_Squadra1, Nome_Squadra2, Numero_gol_Squadra1, Numero_gol_Squadra2`

Altre righe:

`Squadra1, Squadra2, gol_Squadra1, gol_Squadra2`

Leggere il file e costruire il dizionario con chiave il nome della squadra e valore i punti in classifica. Assumete che la squadra che vince ottiene 3 punti, la perdente ottiene 0 punti, e, in caso di pareggio, entrambe ottengono 1 punto.

Ad esempio se **file** contiene

```
Nome_Squadra1, Nome_Squadra2, Numero_gol_Squadra1, Numero_gol_Squadra2
Chelsea, Everton, 2, 0
Arsenal, Tottenham, 0, 0
Chelsea, Arsenal, 0, 1
Tottenham, Everton, 1, 2
```

la funzione deve restituire:

```
{'Chelsea': 3, 'Everton': 3, 'Arsenal': 4, 'Tottenham': 1}
```

Nota: potete assumere che il csv in input sia ben formato, cioè non abbia righe che non rispettano lo schema indicato dalla prima riga, e che ogni riga abbia un valore specificato e del formato atteso per ciascun campo (in particolare non ci sono valori mancanti e gli ultimi due valori di ciascuna riga sono trasformabili in intero).

- **Ex4(file)** un gruppo di amici si trova la sera a giocare a poker, esattamente in 4 ogni sera ma i giocatori NON sono sempre gli stessi. Dopo ogni partita aggiungono una riga, in un file che conserva tutti i risultati, specificando quanto ognuno ha vinto o perso, nel seguente formato:

```
Nome1 vincita/perdita,Nome2 vincita/perdita,Nome3 vincita/perdita,Nome4 vincita/perdita
```

Dove vincita/perdita è un numero intero con segno. Scrivere una funzione che riceve in ingresso il nome di un **file** di testo csv con le caratteristiche descritte sopra e restituisce un dizionario che ha come chiavi i nomi di tutti i giocatori e come valore in corrispondenza di ciascuna chiave una lista composta da 2 elementi, in cui il primo è un valore che indica il numero di partite giocate ed il secondo è un valore che indica la vincita/perdita complessiva del giocatore. Si assuma che il file in input non contenga la riga che specifica il formato dei dati, e che le righe presenti in esso rispettino sempre tale formato. Ad esempio se **file** contiene

```
Paolo 10,Mario -5,Anna -5,Giorgio 0
Paolo 5,Anna 8,Stefano -10,Paola -3
```

la funzione deve restituire:

```
{'Paolo': [2, 15], 'Mario': [1, -5], 'Anna': [2, 3], 'Giorgio': [1, 0], 'Stefano': [1, -10], 'Paola': [1, -3]}
```

- **Ex5(file)** Scrivere una funzione che riceve in ingresso il nome di un **file** di testo contenente, per ogni riga, una serie di numeri interi separati da virgola, e restituisce un dizionario che ha come chiavi i numeri che appaiono nel file e come valori degli insiemi tali che per ciascuna chiave k l'insieme associato contenga il numero di riga di ciascuna riga del file in cui appare k . Assumete che le righe del file siano numerate a partire da 1. Se il file è vuoto la funzione deve restituire un dizionario vuoto. Ad esempio se **file** contiene

```
10,-5,-5,0
10,-5,8,-3
```

la funzione deve restituire:

```
{10: {1,2}, -5: {1,2}, 0: {1}, 8: {2}, -3: {2}}
```

- **Ex6(file)** Scrivere una funzione che prende in ingresso il nome di un **file** di testo csv contenente tutte le partite giocate nei vari anni, nel seguente formato:

Prima riga:

```
Nome_Squadra1,Nome_Squadra2,Numero_gol_Squadra1,Numero_golSquadra2
```

Altre righe:

```
Squadra1,Squadra2,golSquadra1,golSquadra2
```

Leggere il file e costruire il dizionario con chiave il nome della squadra s e come valore la lista di tutte le squadre x tali che s ha battuto almeno una volta x e x non ha mai battuto s . Ad esempio se **file** contiene

```
Nome_Squadra1,Nome_Squadra2,Numero_gol_Squadra1,Numero_gol_Squadra2
Chelsea,Everton,2,0
Everton,Chelsea,3,1
```

```
Arsenal,Tottenham,0,0  
Chelsea,Arsenal,0,1  
Tottenham,Everton,1,2
```

la funzione deve restituire:

```
{'Chelsea': [], 'Everton': ['Tottenham'], 'Arsenal': ['Chelsea'],  
'Tottenham': []}
```

Si assuma che il file csv in input sia sempre ben formato (come nell'esercizio 3).