Esercitazione Python n. 10 -- 6 Dicembre 2022

Obiettivo dell'esercitazione è prendere confidenza con Python e con l'ambiente IDLE.

In laboratorio, per avviare la macchina virtuale, dovete selezionarla all'interno della schermata di VirtualBox (schermata visualizzata all'accensione del PC), cliccando sull'opzione **LXLE-BIAR-4.3**
Una volta avviata la macchina virtuale, svolgete gli esercizi così come indicato nel testo. Scrivete i vostri programmi nei file che abbiamo predisposto: Esercizio 1 nel file A_Ex1.py, Esercizio 2 nel file A_Ex2.py, e così via. Per farlo usare l'ambiente IDLE di Python. Ovviamente è possibile consultare il materiale didattico disponibile sulla pagina web del corso (https://classroom.google.com/u/0/c/NTQ1Njg4NzE1ODA5). Si ricorda che le note relative alle lezioni Python possono essere lette con l'applicazione Jupyter Notebook. Sul sito del corso è comunque distribuita anche una versione pdf delle stesse.
La consegna deve essere effettuata entro le 23:59 di Mercoledì 7 dicembre.

LE ESERCITAZIONI SVOLTE CONSEGNATE OLTRE QUESTO TERMINE, O CHE NON RISPETTANO IL FORMATO INDICATO PER LA CONSEGNA, NON VERRANNO CONSIDERATE. In particolare, vi chiediamo di NON caricare un esercizio svolto per volta, di NON usare formati di compressione diversi da .zip, di NON rinominare i file o metterli in sottocartelle.

Esercizi

- **A_Ex1(l):** Scrivere una funzione che riceve in ingresso una lista **l** di stringhe e restituisce il dizionario in cui le stringhe di **l** sono le chiavi ed il valore associato a ciascuna chiave *k* è il numero di volte in cui la stringa *k* compare nella lista. Se la lista **l** in ingresso è vuota, la funzione deve restituire il dizionario vuoto `{}`. Ad esempio, se **l**=`['casa','orso','cane','casa','orso','casa']` allora il dizionario restituito sarà `{'casa': 3, 'orso': 2, 'cane': 1}`. Notate che l'ordine per i dizionari non è rilevante.

- **A_Ex2(file):** Scrivere una funzione che prende in ingresso il nome di un file di testo contenente solo caratteri alfabetici, spazi e carattere newline ('\n'), e restituisce un dizionario con chiave le parole presenti nel file e valore l'insieme delle righe in cui la parole compare, assumete che la numerazione delle righe parta dalla riga 1. Se il file contiene:

  ```
  tanto va la gatta al lardo che ci lascia lo zampino
  tanto va la gatta
  lascia lo zampino
  ```

  Allora la funzione deve restituire: {'tanto': {1, 2}, 'va': {1, 2}, 'la': {1, 2}, 'gatta': {1, 2}, 'al': {1}, 'lardo': {1}, 'che': {1}, 'ci': {1}, 'lascia': {1, 3}, 'lo': {1, 3}, 'zampino': {1, 3}}

- **A_Ex3(fileName):** Scrivere una funzione che prende ingresso una stringa **fileName,** nome di un file csv, contenente le informazioni sugli esami superati dagli studenti (nel formato `Matricola,Voto,Materia`), e restituisce un dizionario con chiave il nome dell'esame e valore la lista ordinata delle matricole (rappresentate come interi) che hanno superato quell'esame, cioè hanno preso almeno 18. Ad esempio, se il file contiene

  ```
  Matricola,Voto,Materia
  1345,29,Fisica
  1987,17,Fondamenti
  1346,27,Analisi
  ```

```
1896,30,Geometria
1753,30,Fisica
```

La funzione deve restituire

```
{'Fisica': [1345,1753], 'Analisi': [1346], 'Geometria': [1896]}
```

- **A_Ex4(fileName)**: Scrivere una funzione che prende in ingresso una stringa fileName, nome di un file csv che ha lo stesso formato del file descritto nell'esercizio precedente (A_Ex3), e restituisce il dizionario con chiave la matricola (intero) e valore il numero di esami superati. Se il file contiene i dati di sopra, la funzione deve restituire {1345: 1, 1346: 1, 1896: 1, 1753: 1}.

- **A_Ex5(s,n):** Scrivere una funzione che riceve in ingresso una stringa **s,** composta da soli caratteri alfabetici minuscoli e spazi, ed un numero **n** e calcoli il dizionario avente come chiavi lettere minuscole e come valore associato a ciascuna chiave *k* il numero di parole di **s** che cominciano per *k* e sono lunghe almeno **n** caratteri. Il dizionario NON deve contenere come chiavi lettere che non sono iniziali di almeno una parola lunga 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, 'c' : 1, 'l': 2, 'z': 1}`

- **A_Ex6(fileName)** Scrivere una funzione che riceve in input una stringa **fileName,** nome di un file di testo, contenente in ogni riga una serie di numeri interi separati da virgole, e restituisce un dizionario che ha come chiavi i numeri (in formato intero) che appaiono nel file e come valore associato a ciascuna chiave *k* un insieme contenente i numeri di riga in cui appare *k*. Assumete che il numero identificativo delle righe parta dal valore 1. Ad esempio se il 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}}
  ```

- **A_Ex7(fileName)** Scrivere una funzione che prende in ingresso una stringa **fileName,** nome di un file csv, contenente tutte le partite giocate in un torneo di calcio. Il file ha il seguente formato:

  Prima riga:
  `Nome_Squadra1,Nome_Squadra2,Numero_gol_Squadra1,Numero_gol_Squadra2`
  Altre righe:
  `Squadra1,Squadra2,golSquadra1,golSquadra2`

  La funzione deve restituire il dizionario che ha come chiavi i nomi delle squadre che compaiono nel file csv e come valore associato a ciascuna chiave *k* i punti in classifica della squadra *k*. 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}`

- **Ex8(file):** scrivere una funzione Python che prende in ingresso il nome di un **file** csv contenente le informazioni sulle amicizie e inimicizie che si creano in un gruppo di persone nel seguente formato:

```
Nome1, Nome2, relazione
```

dove la relazione può essere solo un valore tra `'amici'` e `'nemici'`. La relazione è sempre simmetrica ed una relazione riportata ad una certa riga può essere modificata da una relazione indicata in una riga successiva. Ad esempio, se **file** contiene:

```
Nome1, Nome2, relazione
Paolo, Marco, amici
Anna, Maria, amici
Paola, Anna, amici
Marco, Giorgio, amici
Giorgio, Marco, nemici
```

la riga 2 dice che Paolo è amico di Marco, e vice-versa (analogamente le altre righe), mentre notiamo che alla riga 5 Marco e Giorgio sono amici, ma alla riga 6 diventano nemici.
La funzione deve leggere il **file** e costruire un dizionario avente come chiavi i nomi di tutte le persone che compaiono nel file, e per valore associato a ciascuna chiave *k* la lista ordinata in ordine lessicografico crescente degli amici di *k* (rimasti al termine della lista). Ogni volta in cui 2 persone diventano amiche dovete aggiungere il nome di ciascuno dei due alla lista degli amici dell'altro, se non era già presente (non ci devono essere duplicati nella lista). Se due persone diventano nemiche dovete eliminare il nome di ciascuno dei due dalla lista degli amici dell'altro (se c'era, altrimenti non dovete fare niente). In riferimento al file d'esempio precedente, la funzione deve restituire il seguente dizionario:

```
{'Marco': ['Paolo'], 'Maria': ['Anna'], 'Paolo': ['Marco'], 'Paola':
['Anna'], 'Giorgio': [], 'Anna': ['Maria', 'Paola']}
```