Esercitazione Python n. 11 -- 13 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 14 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

Ex1(file): scrivere la funzione Python che, preso in ingresso il nome di un **file** di testo, calcoli, usando le espressioni regolari, quante sono le sequenze **non sovrapposte** di 2 parole consecutive aventi la seguente proprietà:

> "La due parole sono composte da almeno due caratteri ed hanno a stessa lettera iniziale e la stessa lettera finale, ignorando la distinzione fra maiuscole e minuscole."

Ad esempio, prendendo come input il file contenente il seguente testo:

```
tanto va Aldino annaspando in giro che era andato al bar
```

la funzione deve restituire come risultato `1`.

Se invece la funzione prende in input

```
Ho trovato delle ossa orsa ora vado velato
```

deve restituire come risultato `2`, infatti le due sequenze sono `ossa orsa` e `vado velato`, mentre `orsa ora` non va bene perché si sovrappone con la prima.

Suggerimenti (per tutti gli esercizi su espressioni regolari): Usate il flag `re.IGNORECASE` per non fare differenza tra maiuscole e minuscole. Potete usare le funzioni `re.finditer()` o `re.findall()` a vostra scelta. Se usate la funzione `re.findall()`, per contare il numero di soluzioni trovate basta usare la funzione `len()` applicata al risultato della `re.findall()`. Si ricorda che una parola è una sequenza di caratteri alfanumerici più l'underscore, preceduta e seguita da almeno un carattere non alfanumerico e non underscore (se però la parola è all'inizio del file è solo seguita da questo tipo di caratteri, mentre se è alla fine del file è solo preceduta da questi).

Ex2(file): scrivere una funzione Python che, preso in ingresso il nome di un **file** di testo, calcoli, usando le espressioni regolari, quante sono le sequenze non sovrapposte di 2 parole consecutive con la seguente proprietà:

> "Almeno 2 lettere della prima parola sono presenti anche nella seconda ma in ordine inverso, cioè se la prima lettera viene prima della seconda nella prima parola, deve venire dopo nella seconda."

Ignorate la differenza fra maiuscole e minuscole. Ad esempio, prendendo come input il file contenente il seguente testo:

```
tanto va Aldo destinando in giro che era arrestato casa propria
```

la funzione deve restituire come risultato `2`.

Ex3(file): scrivere la funzione Python che, preso in ingresso il nome di un **file** di testo calcoli, usando le espressioni regolari, quante volte compaiono in una stessa riga almeno tre parole consecutive tutte con la stessa doppia, ignorando la differenza fra maiuscole e minuscole.

Ad esempio, prendendo come input il file contenente il seguente testo:

```
va Aldo oziando dalla stalla alla casa
chiedendo solo di andare via da casa
```

la funzione deve restituire come risultato `1`.

Ex4(file): Scrivere una funzione Python che prende in ingresso il nome di un **file** csv contenente tutte le eredità di una famiglia nel seguente formato:

```
Oggetto,Antenato,Erede
```

Assumete che il proprietario dell'oggetto sia l'antenato che compare la prima volta (dall'inizio del file) assieme all'oggetto e che se l'antenato NON ha l'oggetto allora l'erede NON lo riceve. Assumete inoltre che l'ordine delle righe conti: in una riga, l'antenato A possiede un oggetto solo se A è il proprietario, oppure se esiste una riga precedente in cui A riceve l'oggetto da un antenato che ha l'oggetto.

La funzione deve leggere il **file** e restituire il dizionario con chiavi i nomi degli oggetti nel **file** e come valore associato a ciascuna chiave *k* una lista contenente due nomi, il nome del proprietario e il nome dell'ultimo erede che ha ricevuto l'oggetto *k*. Ad esempio se il **file** contiene:

```
Oggetto,Antenato,Erede
Anello_di_smeraldi,Maria,Paola
Anello,Silvia,Paolo
Anello_di_smeraldi,Paola,Anna
Anello_di_smeraldi,Anna,Giorgia
```

la funzione deve restituire:

```
{'Anello_di_smeraldi': ['Maria','Giorgia'], 'Anello': ['Silvia',
'Paolo']}
```

Invece se il **file** contiene:

```
Oggetto,Antenato,Erede
Anello_di_smeraldi,Maria,Paola
Anello,Silvia,Paolo
Anello_di_smeraldi,Anna,Giorgia
Anello_di_smeraldi,Paola,Anna
```

la funzione deve restituire:
`{'Anello_di_smeraldi':['Maria','Anna'],'Anello':['Silvia','Paolo']}`
(infatti, alla riga `Anello_di_smeraldi,Anna,Giorgia` Anna non ha l'oggetto, perché non lo ha ricevuto in una riga precedente).

Ex5(file): scrivere una funzione Python che, preso in ingresso il nome di un **file** di testo contenente dei numeri di targa, uno per riga, come nel seguente esempio:

AA234XX
AX54DS
PP2P3
QQ12345
ZZXZ
BB111ZZ

E costruisce un dizionario con 5 chiavi ('auto', 'moto', 'ciclomotore1', 'ciclomotore2', 'errata') e valore in numero di targhe corrispondenti. Le targhe hanno il formato:

auto: 2 lettere maiuscole 3 cifre 2 lettere maiuscole
moto: 2 lettere maiuscole 5 cifre
ciclomotore1: 5 lettere maiuscole o cifre
ciclomotore2: 6 lettere maiuscole o cifre

Tutte le targhe che non rispettano nessuno di questi formati vanno considerate errate. Nell'esempio precedente, la funzione deve restituire {'auto': 2, 'moto':1, 'ciclomotore1':1, 'ciclomotore2':1, 'errata':1}.

Ex6(file): scrivere una funzione Python che, preso in ingresso il nome di un **file** di testo calcoli, usando le espressioni regolari, quante volte compare una parola con la seguente proprietà:

"la lettera iniziale e finale della parola sono uguali ed
all'interno della parola compare almeno una doppia."

Si noti che la doppia deve comparire all'interno della parola, senza quindi considerare il primo e l'ultimo carattere della parola stessa. Ad esempio, prendendo come input il file contenente il seguente testo:

`tanto **attacca** contro **elettore** in giro **abbastanza** era andato a casa`

la funzione deve restituire come risultato `3`.

Ex7(file): un indirizzo IP è un'etichetta numerica che identifica univocamente un dispositivo all'interno di una rete informatica. Esso è costituito da una sequenza di 4 numeri compresi tra 0 e 255, formati da una, due o tre cifre e separati da un punto, (es. 192.168.0.1). Tra tutti i possibili indirizzi IP, vi è un intervallo dedicato alle reti domestiche, i quali hanno formato 192.168.X.Y, dove X ed Y sono due numeri compresi tra 0 e 255. Scrivere la funzione Python che preso in ingresso il nome di un **file** di testo avente il seguente formato

```
Indirizzo_IP
Indirizzo_IP
    […]
Indirizzo_IP
```

e restituisca un dizionario contenente le seguenti chiavi: 'invalidi', 'domestici' ed 'altri', e come valori associati il numero di indirizzi letti dal file che sono rispettivamente non validi, domestici oppure validi non domestici.

Ex8(file): scrivere una funzione Python che prende in ingresso un file di testo contenente dei codici fiscali, scritti uno per riga e che possono contenere o meno spazi tra i vari campi, e restituisce la lista (nell'ordine

in cui sono nel file) delle date di nascita nel formato dd/mm/aaaa (2 cifre obbligatorie per giorno e mese, 4 per anno). Si assuma che se l'anno xx nel codice fiscale è minore o uguale a 20 allora l'anno corrisponde a 20xx, altrimenti corrisponde a 19xx. Si ricorda che il formato dei codici fiscali è:

ABC XYZ aaMgg WnnnV

Dove ABC e XYZ sono sequenze di lettere MAIUSCOLE prese rispettivamente dal cognome e dal nome, aa denota le ultime due cifre dell'anno, M è una lettera maiuscola che specifica il mese secondo la seguente tabella riportata a lato, gg denota il giorno di nascita con la regola che se è maggiore di 40 allora il sesso è femminile e per calcolare la data corretta bisogna togliere 40. L'ultima parte indica il codice del comune (o stato estero) di nascita, composto da una lettera maiuscola e 3 cifre, mentre l'ultima lettera maiuscola è un carattere di controllo. I 4 campi possono essere separati da spazi bianchi oppure essere attaccati. Se la riga NON contiene un codice fiscale corretto dovete inserire nella lista la stringa 'Codice errato', se il codice del mese è inesistente allora inserite nella lista la stringa 'Mese errato', se il giorno è scorretto allora inserite nella lista la stringa 'Giorno errato'. Nella verifica dei giorni potete ignorare gli anni bisestili ed assumere che Febbraio abbia sempre 28 giorni. Ad esempio, se il file contiene

Lettera	Mese	Lettera	Mese	Lettera	Mese
A	gennaio	E	maggio	P	settembre
B	febbraio	H	giugno	R	ottobre
C	marzo	L	luglio	S	novembre
D	aprile	M	agosto	T	dicembre

VXRTRR71C12H501W
PSCTRS 21S33 P
CVV PSX 11D55 H911T
CVV PSX 11O55 H911T
CVV PSX 11D79 H911T

la funzione deve restituire ['12/03/1971', 'Codice errato', '15/04/2011', 'Mese errato', 'Giorno errato'].