Sapienza Università di Roma
Corso di Laurea in Ingegneria Informatica e Automatica
A.A. 2019-20

Esercitazione Python n. 11 -- 10 dicembre 2019

Obiettivo dell'esercitazione è esercitarsi nella lettura di file e nell'utilizzo di espressioni regolari e dizionari.
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,c1,c2):** scrivere una funzione Python che riceve in ingresso una lista **l** di stringhe e due caratteri **c1**e **c2**, e calcoli il numero di stringhe in cui i caratteri **c1** e **c2** compaiono in posizioni consecutive. Ad esempio, se **l** vale `['palla','pallone','casa','casolare']` e **c1** vale `'l'` e **c2** vale `'a'` la funzione deve restituire `3` in quanto le stringhe in cui `'a'` ed `'l'` compaiono in posizioni adiacenti sono `'palla'`, `'pallone'` e `'casolare'`.

- **Ex2(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 sequenza 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."

  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()`. Due parole sono separate da almeno un carattere non alfabetico. Ignorate le lettere accentate, che non saranno presenti nel file. Prendendo come input il file:

  `tanto va **Al**o **d**estin**a**ndo in giro che era **a**rre**s**tato ca**sa** propria`

  deve restituire come risultato `2`.

- **Ex3(file):** scrivere la funzione Python che, preso in ingresso il nome di un **file** contenente una matrice nel formato:

```
Numero_righe Numero_colonne
Prima riga con valori separati da spazi
…
Ultima riga con valori separati da spazi
```

  Calcoli l'indice della riga con il numero maggiore di valori (strettamente) negativi. Se ci sono più righe con lo stesso numero di valori negativi, restituire l'indice più grande. Ad esempio, se il file è:

```
4 4
10 3 4 -1
7 2 3 4
-9 4 2 3
0 -3 2 1
```

Deve restituire `3`, poiché le righe di indice 0, 2 e 3 hanno tutte 1 valore negativo, ma 3 è più grande di 0 e 2.

- **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 a un 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.
Leggere il **file** e costruire il dizionario con chiave il nome dell'oggetto e con valore una lista contenente due nomi, il nome del proprietario e il nome dell'ultimo erede che lo ha ricevuto. Ad esempio se 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 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)**: un quadrato latino è una scacchiera quadrata con un simbolo su ogni casella, in modo che *ognuno di essi* compaia una e una sola volta in ogni riga ed in ogni colonna.

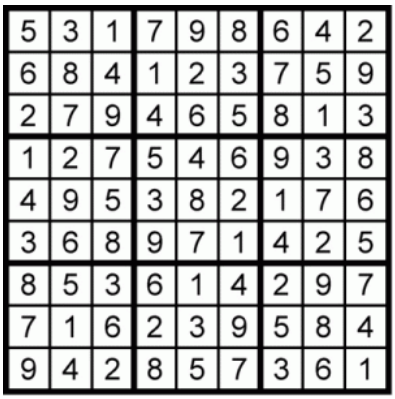
A	B	C	D	E
B	C	E	A	D
C	E	D	B	A
D	A	B	E	C
E	D	A	C	B

Scrivere una funzione Python che prende in ingresso un file **file** nel seguente formato:

ABCDE
BCEAD
CEDBA
DABEC
EDACB

e restituisce il valore booleano `True` se esso rappresenta un quadrato latino, `False` altrimenti.

Ex6(file): Sudoku è il celebre gioco nel quale bisogna completare una griglia 9x9 in modo tale che 1) ogni riga contenga tutti i numeri da 1 a 9, 2) ogni colonna contenga tutti i numeri da 1 a 9 e 3) ognuna delle sottogriglie disgiunte 3x3 contenga tutti i numeri da 1 a 9.



5	3	1	7	9	8	6	4	2
6	8	4	1	2	3	7	5	9
2	7	9	4	6	5	8	1	3
1	2	7	5	4	6	9	3	8
4	9	5	3	8	2	1	7	6
3	6	8	9	7	1	4	2	5
8	5	3	6	1	4	2	9	7
7	1	6	2	3	9	5	8	4
9	4	2	8	5	7	3	6	1

Scrivere una funzione Python che, preso in ingresso il nome di un **file** contenente la possibile soluzione di un Sudoku, nel formato

```
5 3 1 7 9 8 6 4 2
6 8 4 1 2 3 7 5 9
      […]
9 4 2 8 5 7 3 6 1
```

Restituisca il valore booleano `True` se il **file** rappresenta un Sudoku completo valido, `False` altrimenti. Si noti che la soluzione di un Sudoku è un quadrato latino, ma il viceversa non è necessariamente vero.