Sapienza Università di Roma
Corso di Laurea in Ingegneria Informatica e Automatica A.A. 2023-24

Esercitazione Python n. 0 -- 29 Settembre 2024

Obiettivo dell'esercitazione è prendere confidenza con Python, con l'ambiente Spyder, sul proprio computer.

Ovviamente è possibile consultare il materiale didattico disponibile sulla pagina web del corso (https://classroom.google.com/u/1/c/NjIwOTY0ODk3MDAx). Si ricorda che le note relative alle lezioni Python possono essere lette con l'applicazione JupyterLab.

Esercizi

1) Usare la console di Spyder per chiedere all'interprete interattivo di Python di:
    a) valutare il quadrato dell'intero 9
    b) valutare la divisione del numero intero 18 per l'intero 3
    c) restituire la concatenazione delle due parole "casa" e "bella", separate da uno spazio
    d) assegnare alle variabili x e y, rispettivamente, i due numeri reali 8.1 e 9.4 e restituire il valore assoluto della differenza tra x e y
    e) assegnare alla variabile s la stringa "casa" e restituire 3 ripetizioni concatenate di s
    f) valutare il fattoriale di un numero intero casuale compreso tra 0 e 100 (pensare ad importare prima le funzioni necessarie)

2) Usare Spyder come editor per definire (e poi eseguire), per ognuna delle espressioni descritte ai punti a), b), c), d), e), e f) un programma che la calcoli e stampi il risultato a schermo.

Sapienza Università di Roma
Corso di Laurea in Ingegneria Informatica e Automatica A.A. 2023-24

Esercitazione Python n. 1 -- 3 Ottobre 2023

Obiettivi dell'esercitazione:

- prendere confidenza con l'ambiente del laboratorio di Via Tiburtina, con Python e con Spyder;
- scrivere i primi programmi con prendono dati in input;
- prendere confidenza con le espressioni matematiche con interi e reali;
- prendere confidenza con le stringhe e le funzioni su di esse.

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-4.5**
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 esercizio1.py, Esercizio 2 nel file esercizio2.py, e così via. Per farlo usare l'ambiente Spyder. Ovviamente è possibile consultare il materiale didattico disponibile sulla pagina web del corso (https://classroom.google.com/u/1/c/NjIwOTY0ODk3MDAx). Si ricorda che le note relative alle lezioni Python possono essere lette con l'applicazione JupyterLab.

Suggerimenti utili

Si ricorda che

- Importando il modulo **math** è possibile usare funzioni matematiche come *cos()* (per calcolare il coseno del valore di un angolo espresso in radianti) e variabili che contengono il valore (approssimato) di alcune costanti matematiche, come la variabile **pi** che contiene il valore approssimato della costante pi-greco (3.14…) ;
- "Moltiplicando" una stringa *s* per un intero *n* si ottiene la concatenazione di *s* con se stessa per *n* volte;
- Data una stringa *s*, *s[i]* restituisce il carattere in posizione *i* di *s* (si noti che il primo elemento di *s* è associato all'indice 0).
- La funzione *len()* con in input una stringa *s* (i.e., *len(s)*) restituisce la lunghezza di *s*, cioè il numero dei suoi caratteri.

Esercizi

1) Scrivere un programma che prende in input un intero e ne stampa il quadrato. Ad esempio, se l'intero inserito è 4 il programma deve stampare 16.

2) Scrivere un programma che prende in input 2 numeri reali **x** e **y**, e un numero intero **n** e stampa il risultato della divisione di **x** per **y**, arrotondato ad **n** cifre decimali.

3) Scrivere un programma che prende in input un reale **x** che rappresenta un valore in gradi e restituisce il valore del coseno di **x**.

4) Scrivere un programma che prenda in ingresso i 3 coefficienti (**a**, **b** e **c**) di un'equazione di secondo grado (**$y = a x^2 + b x + c$**) ed un valore per la **x** e stampi il corrispondente valore della **y**

5) Scrivere un programma che prende in input una stringa **s** e un intero **n** e stampa la stringa **s** ripetuta **n** volte. Ad esempio se la stringa è 'casa' e l'intero è 3 il programma deve stampare 'casacasacasa'.

6) Scrivere un programma che prende in input un carattere e stampa un quadrato di lato 3 usando quel carattere. Ad esempio se il carattere è '*' il programma deve stampare:
   ***
   ***
   ***

7) Scrivere un programma che chiede in input all'utente un numero di ore **hh**, numero di minuti **mm** e numero di secondi **ss** e stampa a video l'equivalente in numero secondi. Ad esempio, se hh = 2, mm = 1 e ss = 11, il programma deve stampare "*7271*".

8) Scrivere un programma che prenda in ingresso una stringa (**s)** e stampa la stringa un numero di volte pari alla sua lunghezza.

9) Scrivere un programma che memorizzi in un'unica stringa la filastrocca:

Apelle, figlio di Apollo
Fece una palla di pelle di pollo
Tutti i pesci vennero a galla
Per vedere la palla di pelle di pollo
Fatta da Apelle figlio di Apollo.

e:

a) stampi la stringa in modo tale che vada a capo alla fine di ogni verso
b) stampi la lunghezza della stringa
c) stampi il numero totale di occorrenze di 'll'
d) stampi il numero di occorrenze di 'll' in ogni verso nel formato:
   "Nel verso 1, 'll' compare 2 volte.''
   "Nel verso 2, 'll' compare 3 volte''
   ecc.
e) stampi il contenuto della filastrocca senza punteggiatura e andate a capo
f) conti il numero totale di occorrenze della lettera A (senza contare separatamente le occorrenze della forma minuscola e della forma maiuscola)

10) Scrivere un programma che prendi in input una stringa s e un carattere c e sostituisce la prima (eventuale) occorrenza di c con il carattere '1', la seconda (eventuale) con il carattere '2' e la terza (eventuale) con il carattere '3'. Se ci sono altre occorrenze del carattere c NON devono essere sostituiti.

Esercitazione Python n. 2 -- 10 Ottobre 2023

Obiettivo dell'esercitazione è prendere confidenza con l'uso delle istruzioni per il controllo del flusso in Python, e nello specifico con le istruzioni **for** e **if-then-else**.

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-4.5**
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 esercizio1.py, Esercizio 2 nel file esercizio2.py, e così via. Per farlo usare l'ambiente Spyder. Ovviamente è possibile consultare il materiale didattico disponibile sulla pagina web del corso (https://classroom.google.com/u/1/c/NjIwOTY0ODk3MDAx). Si ricorda che le note relative alle lezioni Python possono essere lette con l'applicazione JupyterLab.
La consegna deve essere effettuata entro l'orario di fine dell'esercitazione.

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.

Fate attenzione che gli input siano richiesti all'utente UNO PER VOLTA e NELL'ORDINE RIPORTATO nell'esercizio e che le vostre stampe riportino a video i messaggi ESATTAMENTE nel formato atteso.

Esercizi

1) Scrivere un programma che prende in ingresso una stringa **s** non vuota e verifica se il primo e l'ultimo carattere sono uguali. In caso positivo stampa "*caratteri iniziale e finale uguali*" altrimenti stampa "*caratteri iniziale e finale diversi*". Ad esempio, se **s=**"ambasciata" il programma deve stampare "*caratteri iniziale e finale uguali*".

2) Scrivere un programma che prende in ingresso un anno e verifica se esso è un anno bisestile, stampando a video i messaggi "*anno bisestile*" o "*anno non bisestile*".
   Nota: Un anno è bisestile se e solo se è divisibile per 4 ma non per 100, oppure è divisibile per 400.

3) Scrivere un programma che prende in ingresso un numeratore n ed un denominatore d e stampa a video di che tipo è la frazione $\frac{\mathbf{n}}{\mathbf{d}}$ tra "propria", "apparente" o "impropria". Si ricorda che una frazione è propria se il numeratore è minore del denominatore, apparente se il numeratore è un multiplo del denominatore, e impropria se il numeratore è maggiore del denominatore ma non è un suo multiplo.

4) Scrivere un programma che chiede in input all'utente due stringhe aventi la stessa lunghezza e stampa la stringa composta dai caratteri alternati delle due stringhe. *Esempi*:
   - inserendo nell'ordine "casa" e "mora", il programma stampa "cmaosraa"
   - inserendo nell'ordine "pippo" e "pluto", il programma stampa "ppilpuptoo"

5) Scrivere un programma che chiede in input all'utente una stringa **s** ed un intero positivo **n** e stampa una nuova stringa in cui ogni carattere di **s** è ripetuto **n** volte. *Esempio*:
   - inserendo la stringa "casa" e l'intero "2", il programma stampa 'ccaassaa'

6) Scrivere un programma che chiede in input all'utente due stringhe, **s1** ed **s2**, e stampa la stringa composta da tutti i caratteri che appaiono in **s1** ma NON in **s2**, nell'ordine in cui appaiono in **s1**. Esempio:
   - inserendo nell'ordine le stringhe "casa" e "martellare" , il programma stampa "cs"
   - inserendo nell'ordine le stringhe "cassa" e "martello", il programma stampa "css"

7) Scrivere un programma python che chiede in input all'utente un intero n>2 e stampa tutti i numeri pari compresi tra 2 e n uno per riga (includendo 2 ed n). *Esempio*:
   - Inserendo l'intero "4", il programma stampa "2" e a capo "4"
   - Inserendo l'intero "7", il programma stampa "2", a capo "4" e infine a capo "6"

8) Scrivere un programma python che chiede in input all'utente un numero intero maggiore di zero e stampa a schermo tutti i suoi divisori interi positivi (come lista, senza andare a capo). *Esempio*:
   - Inserendo l'intero "6", il programma stampa "1, 2, 3, 6"

9) Scrivere un programma che prende in ingresso 3 interi **a, b, c**, e determina se essi possano rappresentare le lunghezze dei lati di un triangolo (cioè se siano tutti positivi, e se ciascuno sia minore della somma degli altri due); in caso affermativo stampare il tipo del triangolo tra "*scaleno"*, "*isoscele"*, "*equilatero"*, in caso negativo "*input non valido*".

10) Scrivere un programma python che chiede in input all'utente due numeri interi x e y compresi nell'intervallo [0,10] (assumete che i numeri immessi siano contenuti nell'intervallo) e stampa tutti i numeri fino a 10 esclusi x e y uno per riga. Esempio:
    - Inserendo gli interi "1" e "2", il programma stampa "0" a capo "3" a capo "4" a capo "5" a capo "6" a capo "7" a capo "8" a capo "9" a capo "10"

11) Scrivere un programma che:
    - chiede all'utente di inserire in input nell'ordine un valore che rappresenta una temperatura ed un carattere tra 'F' e 'C', rappresentante la scala utilizzata per la temperatura (C= Celsius, F= Fahrenheit).
    - Stampa a video lo stato dell'acqua alla temperatura indicata tra "*solida*", "*liquida*" o "*gassosa*".

    Nota: Si ricorda che l'acqua è solida quando la temperatura è minore o uguale a 0 $^{\circ}$C ed è gassosa se la temperatura è maggiore o uguale a 100 $^{\circ}$C.
    La formula per convertire la temperatura tra Celsius e Fahrenheit è: $C = (F - 32)/1.8$, dove C indica la temperatura in gradi Celsius e F indica la temperatura in gradi Fahrenheit.

12) Scrivere programma che chiede in input all'utente 2 numeri interi positivi n1 e n2 e stampa (1 per riga) in ordine crescente i multipli di n1 (incluso n1) che sono strettamente più piccoli di n2. Esempi:
    - inserendo gli interi 5 e 16, il programma stampa 5, a capo 10, a capo 15
    - inserendo gli interi 3 e 15, il programma stampa 3, a capo 6, a capo 9, a capo 12
    - inserendo gli interi 7 e 8, il programma stampa 7

Sapienza Università di Roma
Corso di Laurea in Ingegneria Informatica e Automatica A.A. 2023-24

Esercitazione Python n. 3 -- 17 Ottobre 2023

Obiettivo dell'esercitazione è prendere confidenza con la definizione e l'uso delle funzioni in Python, e esercitarsi ulteriormente con le istruzioni **for** e **if-then-else**.

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-4.5**. Una volta avviata la macchina virtuale, svolgete gli esercizi così come indicato nel testo. Usate l'ambiente Spyder per svolgere gli esercizi. Ovviamente è possibile consultare il materiale didattico disponibile sulla pagina web del corso (https://classroom.google.com/u/1/c/NjIwOTY0ODk3MDAx). Si ricorda che le note relative alle lezioni Python possono essere lette con l'applicazione JupyterLab.

La consegna deve essere effettuata entro l'orario di fine dell'esercitazione.

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.

Fate attenzione che gli input siano richiesti all'utente UNO PER VOLTA e NELL'ORDINE RIPORTATO nell'esercizio e che le vostre stampe riportino a video i messaggi ESATTAMENTE nel formato atteso.

Esercizi

N.B.: Negli esercizi da 1) a 4) le funzioni richieste dovranno essere definite nel file chiamato **funzioni_esercizi.py**, presente nella cartella dell'esercitazione, e importate nel file predisposto per ogni esercizio, dove dovrà essere definito il programma richiesto. Per gli esercizi da 4) a 10) le funzioni devono essere completate all'interno dei file che sono stati predisposti con lo stesso nome e, come visto a lezione, possono essere testate eseguendo il file stesso, con i casi di test forniti. Notate infatti che ogni file incorpora del codice python per eseguire alcuni test sulla funzione. NON modificate questo codice, ma **SCRIVETE SOLO il contenuto della funzione**. Non spostate i file dalla loro posizione e non create nuovi file. **Si noti che, come in sede di esame, per la correzione verranno usati insiemi di dati di test diversi.**

1) Definire una funzione Python, chiamata **isIntero(s)**, che prende in ingresso una stringa **s** e restituisce `True` se la stringa rappresenta un numero intero, `False` altrimenti. Si precisa che una stringa rappresenta un intero se non è la stringa vuota, se è composta da sole cifre, oppure comincia con un segno '+' o '-' e per il resto è composta da sole cifre. Ad esempio, se **s** vale '-2', allora deve restituire `True`, mentre se **s** vale '3.2' o '3p', allora deve restituire `False`. Scrivere un programma che chiede all'utente di inserire un numero, invoca la funzione appena definita dandole in input il numeri inserito e stampa "Ok" se la funzione restituisce `True` e "Riprova" se la funzione restituisce `False`.

2) Definire una funzione Python, chiamata **divisoreNonComune(n1,n2,d)**, che prende in input tre numeri interi **n1**, **n2** e **d** maggiori di 0, e restituisce il valore booleano `True` se **d** è un divisore di **n1** che NON è divisore di **n2**. Ad esempio, se **n1** vale 24, **n2** vale 6 e **d** vale 4, allora deve restituire `True`, mentre se **n1** vale 24, **n2** vale 6 e **d** vale 6, allora deve restituire `False`. Scrivere un programma che chiede all'utente di inserire i tre numeri, verifica se sono interi (usando la funzione **isIntero** definita nel precedente esercizio) e se sono maggiori di 0, e, in caso non lo siano, stampa un messaggio di errore, altrimenti invoca la funzione appena definita dandole in input i numeri inseriti e stampa un messaggio che ha la forma "**d** (non) è un divisore di **n1**,

non in comune con **n2**". Nei due casi di esempio forniti sopra, il programma stampa rispettivamente: "4 è un divisore di 24, non in comune con 6" e "6 non è un divisore di 24, non in comune con 6".

3) Diciamo che una stringa è palindroma di livello 1 se il primo e ultimo carattere sono uguali, di livello 2 se il primo è uguale all'ultimo e il secondo al penultimo, e così via. Ovviamente si devono confrontare unicamente i caratteri diversi dallo spazio e dai caratteri di punteggiatura. Scrivere una funzione Python **livelloPalindromicita(s)** che prende in input una stringa **s** e calcola il livello massimo di palindromicità della stringa. Ovviamente, se la stringa ha il primo e ultimo carattere diversi, allora la funzione restituisce 0. Ad esempio, se la stringa **s** vale 'alidfefcila', allora la funzione restituisce è 3, se **s** vale 'Ai lati d'Italia', allora restituisce 6 e se **s** vale 'Carlo', allora restituisce 0. Scrivere un programma Python che chiede all'utente di inserire una stringa, invoca la funzione appena definita su di essa e stampa il suo livello massimo di palindromicità, precisando se la stringa è palindroma. Nei tre casi di esempio forniti sopra, il programma stampa rispettivamente: "La frase non è palindroma. Il livello massimo di palindormicità è 3" nel primo caso, "La frase è palindroma", nel secondo caso e "La frase non ha nessun livello di palindromicità", nell'ultimo caso.

4) **A_Ex1(s)** Completare la funzione Python A_Ex1(s) (contenuta nel file A_Ex1.py) realizzando una funzione che, ricevendo in ingresso una stringa **s**, restituisca una nuova stringa composta dai caratteri più frequenti nella stringa nell'ordine in cui compaiono. Ad esempio se **s** vale `'caso palese'` allora la funzione deve restituire `'ase'` in quanto i tre caratteri compaiono tutti due volte nella stringa e due è la frequenza massima (si noti che nella stringa restituita ogni carattere compare una sola volta). Nel caso in cui la **s** in input sia la stringa vuota la funzione deve restituire una stringa vuota.

5) **A_Ex2(s1,s2)** Completare la funzione Python A_Ex2(s1,s2) (contenuta nel file A_Ex2.py) realizzando una funzione che, ricevendo in ingresso due stringhe, **s1** ed **s2**, restituisca il numero di caratteri che compaiono lo stesso numero di volte (diverso da 0) nelle due stringhe. Ad esempio se **s1** vale `'caso palese'` e **s2** vale `'casa blue'` allora la funzione deve restituire `4`, poiché i caratteri `'c'`, `'a'`, `' '` (spazio) e `'l'` compaiono lo stesso numero di volte in **s1** e **s2**. Ovviamente se una delle due stringhe in input è vuota la funzione deve restituire `0`. Si noti che la funzione **deve** restituire sempre un valore di tipo intero.

6) **A_Ex3(s)** Completare la funzione Python A_Ex3(s) (contenuta nel file A_Ex3.py) realizzando una funzione che, prendendo in ingresso una stringa **s** composta di sole cifre e di lunghezza massima 10, restituisce `True` se almeno un carattere di indice *i* è uguale alla cifra corrispondente. Ad esempio, se la stringa **s** vale `'999379'` allora la funzione deve restituire `True` poiché il carattere di indice 3 è la cifra `'3'`. Se invece la stringa **s** è `'234567890'` allora la funzione deve restituire `False`, perché in nessun carattere di indice *i* è uguale alla cifra corrispondente. Ovviamente, se la stringa in input è vuota, la funzione deve restituire `False`.

7) **A_Ex4(s,n)** Completare la funzione Python A_Ex4(s,n) (contenuta nel file A_Ex4.py) realizzando una funzione che, prendendo in ingresso una stringa **s** ed un numero intero positivo **n** (quindi maggiore di 0), restituisce la stringa che contiene, una sola volta, i caratteri di **s** il cui codice Unicode sia esattamente divisibile per **n**. Ad esempio, se **s** è `'stringa di prova'` ed **n** vale `3`, allora la funzione deve restituire `'rio'` perché solo questi caratteri hanno la proprietà desiderata. Notate che sia la `'r'` che la `'i'` compaiono una sola volta nella stringa restituita, anche se compaiono due volte in **s**. Inoltre, nella stringa restituita i caratteri devono comparire nello stesso ordine in cui compaiono in **s**.

8) **A_Ex5(s)** Completare la funzione Python A_Ex5(s) (contenuta nel file A_Ex5.py) realizzando una funzione che, prendendo in input una stringa **non vuota s** contenente numeri interi positivi separati dal carattere `'-'` (trattino), restituisce il massimo intero contenuto in **s**. Ad esempio, se **s** vale `'12-123-45-6-78'` la funzione deve restituire `123`. Si assuma che `'-'` non compaia mai in prima o ultima posizione (quindi se la stringa contiene un solo numero, non contiene alcun trattino). Si noti che la funzione **deve** restituire sempre un valore di tipo intero.

9) **A_Ex6(s)** Completare la funzione Python A_Ex6(s) (contenuta nel file A_Ex6.py) realizzando una funzione che riceve in ingresso una stringa **s**, e restituisce la frequenza (un numero intero) del carattere alfabetico maiuscolo che appare più volte in **s**. Se **s** non contiene alcun carattere alfabetico maiuscolo, la funzione deve restituire il numero intero 0. Ad esempio, se **s** vale `'aHa^^&^HH'`, la funzione deve restituire il numero intero `3`. Se invece **s** vale `'CIAO'`, la funzione deve restituire il numero intero `1`. Si noti che la funzione **deve** restituire sempre un valore di tipo intero.

10) **A_Ex7(b1,b2)** Completare la funzione Python A_Ex7(b1,b2) (contenuta nel file A_Ex7.py) realizzando una funzione che prende in input due stringhe **b1** e **b2** contenenti solo caratteri `'0'` ed `'1'` (interpretati come bit) e restituisce una stringa di `'0'` ed `'1'` che corrisponde all'AND bit a bit di **b1** e **b2** letti da sinistra a destra. Si noti che l'AND bit a bit effettua l'AND solo dei bit che occupano la stessa posizione nelle stringhe **b1** e **b2**, e per ogni posizione *i* produce 1 solo se entrambi i bit in posizione *i* in **b1** e **b2** sono pari ad 1. Per i bit di **b1** che non hanno un corrispondente bit in **b2** (in questo caso **b1** è più lunga di **b2**) la funzione deve calcolare l'AND con lo `'0'`. Analogamente nel caso in cui **b2** sia più lunga di **b1**. Ad esempio, se **b1** vale `'100'` e **b2** vale `'1011'`, la funzione deve restituire la stringa `'1000'`. Se invece **b1** vale `''` (stringa vuota) e **b2** vale `'111'` la funzione deve restituire '000'. Se entrambe **b1** e **b2** sono vuote, la funzione deve restituire la stringa vuota.

Sapienza Università di Roma
Corso di Laurea in Ingegneria Informatica e Automatica A.A. 2023-24

Esercitazione Python n. 4 -- 24 Ottobre 2023

Obiettivo dell'esercitazione è prendere confidenza con l'uso delle funzioni in Python, ripassare le istruzioni **for** e **if-then-else** ed esercitarsi con l'istruzione **while**.

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-4.5**. Una volta avviata la macchina virtuale, svolgete gli esercizi così come indicato nel testo. Usate l'ambiente Spyder per svolgere gli esercizi. Ovviamente è possibile consultare il materiale didattico disponibile sulla pagina web del corso (https://classroom.google.com/u/1/c/NjIwOTY0ODk3MDAx). Si ricorda che le note relative alle lezioni Python possono essere lette con l'applicazione JupyterLab.

La consegna deve essere effettuata entro l'orario di fine dell'esercitazione.

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.

Fate attenzione che gli input siano richiesti all'utente UNO PER VOLTA e NELL'ORDINE RIPORTATO nell'esercizio e che le vostre stampe riportino a video i messaggi ESATTAMENTE nel formato atteso.

Ogni esercizio richiede che sia completata una funzione all'interno del file predisposto con lo stesso nome e, come visto a lezione, può essere testata eseguendo il file stesso, con i casi di test forniti. Notate infatti che ogni file incorpora del codice python per eseguire alcuni test sulla funzione. **NON modificate questo codice, ma SCRIVETE SOLO il contenuto della funzione. Non spostate i file dalla loro posizione e non create nuovi file. Si noti che, come in sede di esame, per la correzione verranno usati insiemi di dati di test diversi.**

Esercizi

1) **A_Ex1(s1, s2)** Scrivere una funzione che prende in ingresso due stringhe **s1** e **s2** e restituisce il prefisso più lungo comune alle due stringhe. Ad esempio, se **s1** è 'amaca' e **s2** è 'amaranto', la funzione deve restituire la stringa 'ama'. Se invece, **s1** è 'amaca' e **s2** è 'stringa' la funzione deve restituire la stringa vuota ''. Se (almeno) una delle due stringhe è vuota la funzione deve restituire ''.
2) **A_Ex2(s)** Scrivere una funzione che prende in ingresso una stringa **s** composta da almeno un carattere e la scandisce carattere dopo carattere partendo dal primo, sino a quando o la stringa finisce o si incontra un carattere il cui codice Unicode è maggiore di 100. La funzione restituisce:
    - La stringa vuota se la stringa è stata consumata senza trovare il carattere.
    - Il primo carattere con codice Unicode maggiore di 100, nel caso in cui si trova un carattere il cui codice Unicode è maggiore di 100

    Esempio:
    - Se **s** è "CONTE", restituisce ""

- Se **s** è “abaco”, restituisce “o”
- Se **s** è “adamo”, restituisce “m”

3) **A_Ex3(n)** Scrivere una funzione che prende in ingresso un intero **n** e restituisce `True` se **n** è un numero primo, `False`, altrimenti.

Esempio:

- Se **n** è “7”, restituisce `True`
- Se **n** è “4096”, restituisce `False`

4) **A_Ex4(n1, n2)** Scrivere una funzione che prende in ingresso due numeri interi **n1** e **n2** e restituisce il loro prodotto calcolato senza usare l’operatore * (moltiplicazione) o / (divisione).

Esempio:

- Se **n1** e **n2** sono “5” e “2”, restituisce “10”
- Se **n1** e **n2** sono “-3” e “4”, restituisce “-12”

Nota: Gli interi in input possono essere positivi, negativi o pari a 0.

5) **A_Ex5(s)** Scrivere una funzione che prende in ingresso una stringa **s** e restituisce il carattere che compare più volte in **s.** Se c’è più di un carattere con queste caratteristiche, restituisce quello che fra questi ha tra le sue occorrenze l'indice maggiore.

Esempio:

- Se **s** è “pippo”, restituisce “p”
- Se **s** è “clarabella”, restituisce “a” (sia ‘a’ che ‘l’ compaiono 3 volte, che è il massimo numero di volte in cui compare un carattere nella stringa, ma c’è una occorrenza di ‘a’ che ha l'indice maggiore di tutte le occorrenze di ‘l’)

Nota: Si assuma che la stringa **s** in input non sia mai la stringa vuota.

6) **A_Ex6(s1, s2)** Scrivere una funzione che prende in ingresso due stringhe **s1** e **s2** non nulle, della stessa lunghezza e corrispondenti a due numerali da interpretare CP2 e restituisce una stringa corrispondente al numerale in CP2 della somma di s1 e s2 (se la somma ha avuto successo) o 'ERRORE' in tutti gli altri casi. Si ricorda che in una somma in CP2 0+1=1, 1+1=0 col riporto di 1 e 1+1+1=1 col riporto di 1. Inoltre, si ricorda che se gli addendi hanno segni differenti il risultato è sempre corretto; se hanno lo stesso segno il risultato è corretto solo se ha lo stesso segno degli addendi.

Esempio:

- Se **s1** è “000” e **s2** è “01” restituisce “ERRORE”
- Se **s1** è “101” e **s2** è “001” restituisce “110”

7) **A_Ex7(ex, cp2)** Scrivere una funzione che prende in ingresso, nell'ordine, due stringhe **ex** e **cp2** non nulle, della stessa lunghezza e corrispondenti a due numerali da interpretare in eccesso e CP2, rispettivamente; la funzione e restituisce 'ERRORE' se la lunghezza delle stringhe non è corretta, 'ex' se $(ex)_{ex} >= (cp2)_{cp2}$, 'cp2', altrimenti.

Esempio:

- Se **ex** è “000” e **cp2** è “01” restituisce “ERRORE”
- Se **ex** è “001” e **cp2** è “001” restituisce “cp2”

8) **A_Ex8(numBin)** Scrivere una funzione che prende in ingresso una stringa **numBin** corrispondente a un binario puro con parte frazionaria nella forma di una sequenza di caratteri '01' con un punto al suo interno (almeno un carattere a sinistra del punto, almeno un carattere a destra del punto), e.g., '101.01' La funzione assume sempre corretto il formato della stringa e restituisce una stringa corrispondente al valore in base 10 di **numBin**.

Esempio:

- Se **numBin** è "1.1" la funzione restituisce "1.5"
- Se **numBin** è "0.01" la funzione restituisce "0.25"

9) Scrivere una funzione che riceve in ingresso una stringa **s** composta da zeri e uno e un intero **n**>=len(s) e calcola il CP2 di **s** esteso a **n** bit, ovvero ricopia sino al primo 1 compreso, poi complementa ed estende il numerale a **n** bit. Assumere che l'ingresso sia corretto.

Esempio:

- Se **s** vale '11' e **n** = 5 la funzione deve restituire '00001'**.**

10) Scrivere una funzione che riceve in ingresso una stringa **s** e due caratteri **c1** e **c2** non necessariamente distinti e non necessariamente presenti in **s**; la funzione calcola la lunghezza della più corta sequenza di caratteri consecutivi di **s** che contiene sia **c1** che **c2**.

Esempio:

- Se **s** vale 'casale' e **c1** = 'l' e c2 = 'a', la funzione deve restituire 2;
- Se **s** vale 'casale' e **c1** = 'a' e c2 = 'a', la funzione deve restituire 1;
- Se **s** vale 'casale' e **c1** = 'z' e c2 = 'f', la funzione deve restituire 0.

Sapienza Università di Roma
Corso di Laurea in Ingegneria Informatica e Automatica A.A. 2023-24

Esercitazione Python n. 5 -- 31 Ottobre 2023

Obiettivo dell'esercitazione è prendere confidenza con l'uso delle funzioni in Python, ripassare le istruzioni **for**, **if-then-else, while,** ripassare le funzioni ed esercitarsi con le istruzioni **break** e **continue**.

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-4.5**. Una volta avviata la macchina virtuale, svolgete gli esercizi così come indicato nel testo. Usate l'ambiente Spyder per svolgere gli esercizi. Ovviamente è possibile consultare il materiale didattico disponibile sulla pagina web del corso (https://classroom.google.com/u/1/c/NjIwOTY0ODk3MDAx). Si ricorda che le note relative alle lezioni Python possono essere lette con l'applicazione JupyterLab.

La consegna deve essere effettuata entro l'orario di fine dell'esercitazione.

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.

Fate attenzione che gli input siano richiesti all'utente UNO PER VOLTA e NELL'ORDINE RIPORTATO nell'esercizio e che le vostre stampe riportino a video i messaggi ESATTAMENTE nel formato atteso.

Ogni esercizio richiede che sia completata una funzione all'interno del file predisposto con lo stesso nome e, come visto a lezione, può essere testata eseguendo il file stesso, con i casi di test forniti. Notate infatti che ogni file incorpora del codice python per eseguire alcuni test sulla funzione. **NON modificate questo codice, ma SCRIVETE SOLO il contenuto della funzione. Non spostate i file dalla loro posizione e non create nuovi file. Si noti che, come in sede di esame, per la correzione verranno usati insiemi di dati di test diversi.**

Esercizi

1) **A_Ex1(s, n)** Scrivere una funzione che prende in ingresso una stringa **s** un intero **n** e restituisce la sottosequenza di **s** di lunghezza **n** tale per cui la somma dei codici unicode dei caratteri che la costituiscono è massima. Se la stringa ha una lunghezza inferiore ad **n** restituisce la stringa vuota.

Esempio:

- Se **s** è 'scacco matto' e **n** è 3, la funzione deve restituire la stringa 'tto'
- Se **s** è 'scacco' e **n** è 3 la funzione deve restituire la stringa 'sca'
- Se, **s** è 'scacco' e **n** è 7 la funzione deve restituire la stringa vuota ".

2) **A_Ex2(s)** Scrivere una funzione che prende in ingresso una stringa **s** e restituisce una stringa composta da tutti e soli i caratteri alfabetici maiuscoli contenuti in **s**, senza ripetizioni. La stringa restituita deve essere ordinata in ordine alfabetico (cioè secondo l'ordine UNICODE crescente).

Esempio:

- Se **s** è 'cIAo MAmMa', la funzione deve restituire la stringa 'AIM'.

3) **A_Ex3(s)** Scrivere una funzione che prende in ingresso una stringa **s** e restituisce una stringa ottenuta "slittando circolarmente" tutti i caratteri di **s** di un numero di posizioni tali per cui la somma $S=i*c_i$ è massima, dove c_i è il codice UNICODE del carattere di **s** in posizione i-esima, con i=0,1,…n-1, dove n è la lunghezza di **s** e 0 è la posizione più a sinistra.

Esempio:

- Se **s** è 'mamma', la funzione restituisce 'amamm'; infatti, slittando circolarmente tutti i caratteri di **s** di 1 posizione verso destra, otteniamo la stringa 'amamm' per la quale S= 1066; slittandoli di 2 posizioni, otteniamo la stringa 'mamam' per la quale S= 1042; slittandoli di 3 posizioni, otteniamo la stringa 'mmama' per la quale S= 1018; slittandoli di 4 posizioni, otteniamo la stringa 'ammam' per la quale S= 1054; slittandoli di 5 posizioni, otteniamo la stringa 'mamma' per la quale S= 1030.

4) **A_Ex4(n)** Scrivere una funzione che prende in ingresso un numero intero positivo **n** e restituisce True se le sue cifre sono tutti numeri pari, False altrimenti. Per risolvere l'esercizio usare l'istruzione **break**.

Esempio:

- Se **n** è 2416, la funzione restituisce False;
- Se **n** è 4860, la funzione restituisce True.

5) **A_Ex5(n)** Scrivere una funzione che prende in ingresso un numero intero **n** e restituisce una stringa che rappresenta una scacchiera nXn su **n** righe di **n** caratteri ognuna che contiene il carattere ' ' (spazio) per le caselle bianche, il carattere '*' per le caselle nere e comincia con una casella bianca. Controllare che **n** sia compatibile con il requisito che la funzione crei sempre una scacchiera con almeno una casella nera e una bianca, restituendo 'ERRORE' se ciò non è possibile.

Esempio:

- Se **n** è 2, la funzione restituisce la stringa ' *\n* \n';
- Se **n** è 3, la funzione restituisce la stringa ' * \n* *\n * \n'
- Se **n** è 4, la funzione restituisce la stringa:
  ' * *\n* * \n * *\n* * \n'

6) **A_Ex6(v1)** Scrivere una funzione che prende in input due interi **v1** e **v2**, superiori o uguali a 1, che rappresentano rispettivamente la velocità di partenza di due viaggiatori, e restituisce il numero di giorni necessario al secondo viaggiatore per raggiungere il primo, assumendo che il primo viaggi a velocità costante mentre il secondo viaggi alla velocità di partenza il primo giorno, al doppio della velocità di partenza il secondo giorno, al triplo il terzo giorno e così via (problema posto da Fibonacci nel 1200).

Esempio:

- Se **v1**=20 e **v2**=1, la funzione restituisce 39;
- Se **v1**=10 e **v2**=5, la funzione restituisce 3;
- Se **v1**=1 e **v2**=1, la funzione restituisce 1.

7) **A_Ex7(b1,b2)** Scrivere una funzione che prende in input due stringhe **b1** e **b2** contenenti solo caratteri '0' ed '1' (interpretati come bit) e restituisce una stringa di '0' ed '1' che corrisponde all'AND bit a bit di **b1** e **b2** letti da sinistra a destra. Si noti che l'AND bit a bit effettua l'AND solo dei bit che occupano la stessa posizione nelle stringhe **b1** e **b2**, e per ogni posizione i produce 1 solo se entrambi i bit in posizione i in **b1** e **b2** sono pari ad 1. Per i bit di b1 che non hanno un corrispondente bit in **b2** (in questo caso **b1** è più lunga di **b2**) la funzione deve calcolare l'AND con lo '0'. Analogamente nel caso in cui **b2** sia più lunga di **b1**.

Esempio:

- Se **b1** vale '100' e **b2** vale '1011', la funzione deve restituire la stringa '1000'
- Se **b1** vale '' e **b2** vale '111' la funzione deve restituire '000'
- Se entrambe **b1** e **b2** sono vuote, la funzione deve restituire la stringa vuota.

8) **A_Ex8(n)** Scrivere una funzione che prende in ingresso un numero intero **n** e restituisce una stringa che rappresenta un quadrato nXn dove vengono disegnati pieni solo i bordi e le due diagonali, tutte le altre caselle sono vuote. Usate il carattere ' ' (spazio) per le caselle vuote ed il carattere '*' per le caselle piene. Controllare che il numero **n** sia strettamente positivo, restituendo 'ERRORE' in caso contrario.

Esempi:

- Se **n** è 2, la funzione restituisce la stringa '**\n**', che corrisponde al disegno:

```
**
**
```

- Se **n** è 5, la funzione restituisce la stringa '*****\n** **\n* * *\n** **\n*****\n', che corrisponde al disegno:

```
*****
** **
* * *
** **
*****
```

- Se **n** è 6, la funzione restituisce la stringa '******\n** **\n* ** *\n* ** *\n** **\n******\n', che corrisponde al disegno:

```
******
**  **
* ** *
* ** *
**  **
******
```

9) **A_Ex9(s1,s2)** Scrivere una funzione che riceve in ingresso due stringhe **s1** ed **s2** e calcola la lunghezza della più lunga sottosequenza comune alle due stringhe.

Esempi:

- Se **s1** vale 'casa' ed **s2** vale 'cappello' la funzione deve restituire 2.
- Se **s1** vale ' carcassa' ed **s2** vale 'cappello' la funzione deve restituire 2.

10) **A_Ex10(s)** Scrivere una funzione che riceve in ingresso una stringa **s** (composta da caratteri qualunque) e restituisce una stringa composta da tutti i caratteri che compaiono in **s,** in ordine crescente di codice UNICODE e senza ripetizioni.

Esempi:

- Se **s** vale 'casale' la funzione deve restituire 'acels';
- Se **s** vale 'Tanto va la Gatta' la funzione deve restituire 'GTalnotv'.

Sapienza Università di Roma
Corso di Laurea in Ingegneria Informatica e Automatica A.A. 2023-24

Esercitazione Python n. 6 -- 7 Novembre 2023

Obiettivo dell'esercitazione è iniziare a prendere confidenza con liste, tuple e insiemi.

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-4.5**. Una volta avviata la macchina virtuale, svolgete gli esercizi così come indicato nel testo. Usate l'ambiente Spyder per svolgere gli esercizi. Ovviamente è possibile consultare il materiale didattico disponibile sulla pagina web del corso (https://classroom.google.com/u/1/c/NjIwOTY0ODk3MDAx). Si ricorda che le note relative alle lezioni Python possono essere lette con l'applicazione JupyterLab.

La consegna deve essere effettuata entro l'orario di fine dell'esercitazione.

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.

Fate attenzione che gli input siano richiesti all'utente UNO PER VOLTA e NELL'ORDINE RIPORTATO nell'esercizio e che le vostre stampe riportino a video i messaggi ESATTAMENTE nel formato atteso.

Ogni esercizio richiede che sia completata una funzione all'interno del file predisposto con lo stesso nome e, come visto a lezione, può essere testata eseguendo il file stesso, con i casi di test forniti. Notate infatti che ogni file incorpora del codice python per eseguire alcuni test sulla funzione. **NON modificate questo codice, ma SCRIVETE SOLO il contenuto della funzione. Non spostate i file dalla loro posizione e non create nuovi file. Si noti che, come in sede di esame, per la correzione verranno usati insiemi di dati di test diversi.**

Esercizi

- **A_Ex1(l1, l2)** Scrivere una funzione che, ricevendo in ingresso due liste **l1** e **l2** contenenti numeri interi, e tali che len(l2)>= len(l1), restituisca una nuova lista composta dalla somma degli interi che si trovano nella stessa posizione in **l1** e **l2**. Se **l1** è più corta si assuma che gli interi mancanti siano uguali a 0. Ad esempio, se **l1** vale [3,6] e **l2** vale [3,4,9] la funzione deve restituire [6, 10, 9]. Se **l1** ed **l2** sono entrambe vuote (cioè pari a []), la funzione deve restituire una lista vuota.

- **A_Ex2 (start, n)** Scrivere una funzione che riceve in ingresso due interi non negativi **start** e **n** e restituisce la lista dei primi n numeri dispari >= **start**. Se **n** vale zero, la funzione deve restituire una lista vuota. Ad esempio, se **start** vale 4 e **n** vale 3 la funzione deve restituire [5, 7, 9].

- **A_Ex3(l,n)** Scrivere una funzione che prende in ingresso una lista **l** di stringhe e un intero non negativo **n** e restituisce la stringa composta, in ordine, da tutti i caratteri con indice **n** di tutte le stringhe della lista. Se in una stringa in **l** non c'è il carattere di indice **n** (perché la stringa è troppo corta), la funzione deve inserire il carattere '!'. Se la lista in ingresso è vuota, la funzione deve restituire la stringa vuota. Ad esempio, se **l** vale ['tanto', 'va', 'la', 'gatta', 'al', 'lardo'] e **n** vale 3 la funzione deve restituire la stringa 't!!t!d'

- **A_Ex4(l)** Scrivere una funzione che prende in ingresso una lista **l** di interi e restituisce la stringa composta da soli caratteri 'P' e 'D' che indicano se l'intero nella lista è pari o dispari. Più precisamente, se il numero in posizione *i* in **l** è pari, la stringa restituita dalla funzione contiene 'P' in posizione *i,* 'D' altrimenti. Se la lista è vuota la funzione deve restituire la stringa vuota. Ad esempio, se **l** vale [3,7,8,9] la funzione deve restituire la stringa 'DDPD'.

- **A_Ex5(l)** Scrivere una funzione che prende in ingresso una lista **l** di stringhe e restituisce una lista di interi, tale che ciascun intero *x* nella lista restituita è ottenuto dalla somma dei codici Unicode dei caratteri della stringa che in **l** occupa la stessa posizione che occupa *x* nella lista restituita. Ad esempio, se **l** è ['ama', 'ma', 'amaca'] la funzione dovrebbe restituire la lista [303,206,499], in quanto il codice Unicode di 'a' è 97, quello di 'm'è 109 e quello di 'c'è 99 (e quindi ad esempio ad 'ama' corrisponde l'intero 303). Se la lista in input è vuota, la funzione deve restituire una lista vuota.

- **A_Ex6(l1,l2)** Scrivere una funzione che prende in ingresso due liste **l1** e **l2** contenenti numeri interi e restituisce una lista contenente gli elementi di **l1** che NON compaiono in **l2.** Nella lista restituita gli elementi devono comparire in **ordine crescente**. Si noti che eventuali ripetizioni di un elemento x in **l1** sono tutte non presenti nel risultato se x compare anche solo una volta in l2, mentre sono tutte presenti nel risultato se x non compare in **l2**. Ovviamente, se la lista **l1** è vuota la funzione deve restituire la lista vuota, invece se **l2** è vuota, la funzione deve restituire una lista uguale a **l1**. Ad esempio, se **l1** = [1, 3, 7, 2, 1,-5, 7] e l2 = [1, 3], la funzione deve restituire la lista [-5, 2,7,7].

- **A_Ex7(l)** Scrivere una funzione che prende in ingresso una lista **l** contenente stringhe non vuote e restituisce (la tupla costituita da) i caratteri più frequenti di ciascuna stringa. Se una stringa ha più di un carattere più frequente, la tupla deve contenere il più piccolo carattere nell'ordine Unicode. Ad esempio, se **l**= ['amaca', 'amaranto', 'rosso'] allora la funzione deve restituire ('a', 'a', 'o').

- **A_Ex8(l)** Scrivere una funzione che prende in ingresso una lista **l** *non vuota* di stringhe e calcola quale carattere alfabetico minuscolo ('a'-'z') compare in più stringhe. Se ci sono più caratteri che compaiono lo stesso numero di volte si scelga quello alfabeticamente più grande. Ad esempio, se **l** = ['casa', 'senape', 'ketchup', 'pasta'], allora il carattere da restituire è 's' che compare in 3 stringhe ed è più grande di 'a' e 'p', che anche compaiono in 3 stringhe.

- **A_Ex9(l)** Scrivere una funzione che prende in ingresso una lista **l** di stringhe e restituisce un insieme contenente tutti e soli i caratteri che appaiono almeno due volte in una delle stringhe. Ad esempio, se **l**=['casa', 'albero', 'bello'], allora l'insieme da restituire sarà {'a', 'l'}. Se la lista **l** in ingresso è vuota, oppure non ci sono caratteri che appaiono almeno due volte in una stringa di **l**, la funzione deve restituire l'insieme vuoto.

- **A_Ex10(l)** Scrivere una funzione che prende in ingresso una lista **l** di stringhe e restituisce un insieme di coppie (x1, x2) tali che x1 e x2 sono stringhe di **l** della stessa lunghezza e x1 è diverso da x2. Notare che una coppia è semplicemente una tupla di lunghezza due. In altri termini, l'insieme deve contenere tutte e sole le coppie formate combinando a due a due, in tutti i possibili modi, le stringhe di **l** che hanno la stessa lunghezza, escludendo i casi in cui una stringa si combina con sé stessa. Ad esempio, se **l**=['tre', 'h', 'plqa', 'a', 'due'], allora l'insieme di coppie da restituire sarà {('tre','due'), ('h','a'), ('a','h'), ('due','tre')}. Se la lista **l** in ingresso è vuota, la funzione deve restituire l'insieme vuoto.

Esercitazione Python n. 7 -- 14 Novembre 2023

Obiettivo dell'esercitazione è esercitarsi con liste, tuple e insiemi.

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-4.5**. Una volta avviata la macchina virtuale, svolgete gli esercizi così come indicato nel testo. Usate l'ambiente Spyder per svolgere gli esercizi. Ovviamente è possibile consultare il materiale didattico disponibile sulla pagina web del corso (https://classroom.google.com/u/1/c/NjIwOTY0ODk3MDAx). Si ricorda che le note relative alle lezioni Python possono essere lette con l'applicazione JupyterLab.

La consegna deve essere effettuata entro l'orario di fine dell'esercitazione.

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.

Fate attenzione che gli input siano richiesti all'utente UNO PER VOLTA e NELL'ORDINE RIPORTATO nell'esercizio e che le vostre stampe riportino a video i messaggi ESATTAMENTE nel formato atteso.

Ogni esercizio richiede che sia completata una funzione all'interno del file predisposto con lo stesso nome e, come visto a lezione, può essere testata eseguendo il file stesso, con i casi di test forniti. Notate infatti che ogni file incorpora del codice python per eseguire alcuni test sulla funzione. **NON modificate questo codice, ma SCRIVETE SOLO il contenuto della funzione. Non spostate i file dalla loro posizione e non create nuovi file. Si noti che, come in sede di esame, per la correzione verranno usati insiemi di dati di test diversi.**

Esercizi

- **A_Ex1(l)** Scrivere una funzione che prende in ingresso una lista **l** di insiemi di numeri interi e restituisce un altro insieme contenente tutti e soli gli elementi che appaiono in uno ed uno solo degli insiemi in **l**. Ad esempio, se x=[{3,2,90},{2,87,23},{2,23,3}], allora l'insieme da restituire sarà {90,87}. Se la lista in ingresso è vuota, la funzione deve restituire un insieme vuoto.

- **A_Ex2(l)** Scrivere una funzione che prende in ingresso una lista **l** di stringhe e restituisce un insieme di tutte e sole le coppie (x,n) (cioè tuple di dimensione due) tali che x è una stringa di **l** ed n è il numero di volte che la stringa x appare in **l**. Ad esempio, se **l**=['jkl', 'h', 'plqa', 'jkl', 'h', 'xkj'], allora la funzione deve restituire l'insieme di coppie {('jkl',2), ('h',2), ('plqa',1), ('xkj',1)}. Se la lista **l** in ingresso è vuota, la funzione deve restituire l'insieme vuoto.

- **A_Ex3(a, b)** Scrivere una funzione che prende in ingresso due insiemi **a** e **b** di coppie (cioè tuple di dimensione due) tali che l'insieme **a** contiene le coppie (nome, cittanascita) (ogni coppia indica che quella persona è nata in quella città) e l'insieme **b** contiene le coppie (citta, regione) (ogni coppia indica che quella città appartiene a quella regione). La funzione deve restituire un altro insieme contenente tutte e sole le coppie (nome, regione) che indicano che quella persona è nata in quella regione. Ad esempio, se **a**={('Giovanni', 'Napoli'), ('Marco', 'Roma'), ('Giuseppe', 'Rieti'), ('Aldo', 'Torino')} e **b**={('Napoli', 'Campania'), ('Benevento', 'Campania'), ('Roma', 'Lazio'), ('Rieti', 'Lazio'), ('Genova', 'Liguria')}, allora l'insieme da restituire sarà {('Giovanni', 'Campania'), ('Marco', 'Lazio'), ('Giuseppe',

'Lazio')}. Si assuma che **a** e **b** siano insiemi che contengono solo coppie i cui elementi sono stringhe, oppure che siano vuoti. Se una città è presente in **a** ma non in **b** allora la persona nata in quella città NON deve essere nell'insieme risultato. Se uno dei due insiemi in ingresso è vuoto, la funzione deve restituire un insieme vuoto.

- **A_Ex4(l,c,n)** Scrivere una funzione che prende in ingresso una lista **l** di stringhe, un carattere **c** ed un numero intero **n** e restituisce una lista ottenuta da **l** eliminando tutte le stringhe che contengono almeno **n** volte il carattere **c**. Ad esempio, se **l** = ['palla','casse','palo'], **c** = 'a' ed **n** = 2 allora la funzione deve restituire la lista ['casse','palo']. Si noti che le stringhe non eliminate compaiono nel risultato nello stesso ordine in cui compaiono in **l**. Se la lista **l** in ingresso è vuota, la funzione deve restituire una lista vuota.

- **A_Ex5(l)** Scrivere una funzione che prende in ingresso una lista **l** di numeri interi positivi e restituisce una lista ottenuta modificando **l** nel seguente modo: ogni volta che un numero all'interno di **l** è più piccolo del successivo, alla lista viene aggiunta in fondo la differenza tra il secondo ed il primo dei due numeri. Ad esempio, se **l**=[10,1,11,31,251], allora la lista da restituire sarà [10,1,11,31,251,10,20,220,10,200,190]. Si noti che il confronto fra un numero ed il successivo deve essere fatto anche per i nuovi elementi inseriti, come mostrato dall'esempio. Se la lista in ingresso è vuota, la funzione deve restituire una lista vuota.

- **A_Ex6(v1,v2)** Scrivere una funzione che prende in ingresso due insiemi **v1** e **v2** di interi, e restituisce l'insieme di tutte e sole le coppie **(n1,n2)** del prodotto cartesiano di **v1** e **v2** tali per cui **n1** è un divisore di **n2** oppure **n2** è un divisore di **n1**. Ad esempio, se **v1**={3,10,28} e v2={2,20}, allora la funzione deve restituire l'insieme {(10,2),(28,2),(10,20)}. Se almeno uno tra i due insiemi **v1** e **v2** è vuoto, la funzione restituisce l'insieme vuoto.

- **A_Ex7(l)** Scrivere una funzione che prende in ingresso una lista non vuota **l** di coppie **(s1,n)** dove **s1** è una stringa e **n** un intero positivo, e restituisce la tupla (**s2**,**m**), tale che **s2** è la stringa che compare più volte come prima componente di **l** e **m** è il valore "massimo associato a **s2"**, cioè il massimo tra tutti gli interi **n** tali che **(s2,n)** appartiene a **l**. Se ci sono più stringhe che compaiono lo stesso numero di volte come prima componente in **l**, la funzione restituisce (**s2**,**m**) tali che **m** è maggiore. Se ci sono più stringhe che compaiono lo stesso numero di volte come prima componente in **l** e il cui massimo associato è lo stesso, restituisce (**s2**,**m**) tali che **s2** è alfabeticamente più piccola. Ad esempio, se **l**=[['italiano',6], ['geografia', 10], ['matematica',6],['storia',8], ['matematica', 10], ['storia', 8], ['geografia', 7]], allora la funzione deve restituire ('geografia',10).

- **A_Ex8(l)** Scrivere una funzione che prende in ingresso una lista **l** di coppie **(s,l2)** dove **s** è una stringa e **l2** una lista di stringhe, e restituisce una lista **l3** di coppie **(s,v)**, ordinata rispetto ad **s**, tale che **s** è una stringa che compare come prima componente in una coppia di **l** e che compare al più una volta come prima componente di **l3**, e **v** è l'insieme che contiene tutti e soli gli elementi che compaiono in almeno una lista **l2**, per ogni **l2** tale che **(s,l2)** appartiene a **l**. Ad esempio, se **l**=[('pippo',['blu','giallo']), ('minnie', ['rosa']), ('pippo', ['rosso', 'blu'])], allora la funzione deve restituire la lista [('minnie',{'rosa'}), ('pippo', {'blu', 'rosso', 'giallo'})]. Se la lista **l** in ingresso è vuota, la funzione deve restituire la lista vuota.

Esercitazione Python n. 8 -- 21 Novembre 2023

Obiettivo dell'esercitazione è esercitarsi con l'accesso ai file.

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-4.5**. Una volta avviata la macchina virtuale, svolgete gli esercizi così come indicato nel testo. Usate l'ambiente Spyder per svolgere gli esercizi. Ovviamente è possibile consultare il materiale didattico disponibile sulla pagina web del corso (https://classroom.google.com/u/1/c/NjIwOTY0ODk3MDAx). Si ricorda che le note relative alle lezioni Python possono essere lette con l'applicazione JupyterLab.

La consegna deve essere effettuata entro l'orario di fine dell'esercitazione.

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.

Fate attenzione che gli input siano richiesti all'utente UNO PER VOLTA e NELL'ORDINE RIPORTATO nell'esercizio e che le vostre stampe riportino a video i messaggi ESATTAMENTE nel formato atteso.

Ogni esercizio richiede che sia completata una funzione all'interno del file predisposto con lo stesso nome e, come visto a lezione, può essere testata eseguendo il file stesso, con i casi di test forniti. Notate infatti che ogni file incorpora del codice python per eseguire alcuni test sulla funzione. **NON modificate questo codice, ma SCRIVETE SOLO il contenuto della funzione. Non spostate i file dalla loro posizione e non create nuovi file. Si noti che, come in sede di esame, per la correzione verranno usati insiemi di dati di test diversi.**

Esercizi

- **A_Ex1(fileName):** Scrivere una funzione che prende in ingresso una stringa **fileName**, nome di un file di testo, e restituisce il numero totale di caratteri alfabetici contenuti nel file.

- **A_Ex2(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 una lista contenente tutte e sole le coppie (`Matricola`, `Materia`) per gli esami superati, tali cioè che lo studente con quella matricola abbia preso un voto maggiore o uguale a 18. Notare che una coppia è semplicemente una tupla di lunghezza due. Le coppie devono comparire nella lista nello stesso ordine in cui le informazioni si trovano nel file. 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

  ```
  [('1345','Fisica'),('1346','Analisi'),('1896','Geometria'),('1753','Fisica')]
  ```

- **A_Ex3(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_Ex2**), e restituisce un insieme contenente tutte e sole le `Materie` per cui ci sono almeno due studenti che hanno preso almeno 29 (potete assumere che nel file non ci siano mai due righe uguali, aventi stessa matricola e stessa materia – in altri termini, il file contiene un solo voto per un certo studente ed una certa materia).

- **A_Ex4(fileName,anno):** Scrivere una funzione che prende in ingresso una stringa **fileName** che denota il nome di un file csv che contiene le informazioni sul numero e tipo di oggetti venduti in anni consecutivi da un negozio**,** ed un intero **anno** che denota un anno (che potete assumere siano tra quelli presenti nel file) e restituisce il nome dell'oggetto più venduto in quell'anno. Il file è nel seguente formato (come esempio, ma il numero di anni potrebbe variare):

  ```
  Anno,2010,2011,2012
  Zaino,27,21,11
  Maglione,11,13,16
  Giubbotto,13,15,17
  ```

  Se ci sono più oggetti con lo stesso numero di vendite, si restituisca quello alfabeticamente più grande. Ad esempio, se il file è quello di sopra e l'anno è il 2012 allora il risultato restituito dalla funzione deve essere 'Giubbotto'.

- **A_Ex5(fileName,oggetto):** Scrivere una funzione che prende in ingresso una stringa **fileName** che denota il nome di un file csv nel formato di sopra ed il nome di un **oggetto** e restituisce il numero intero dell'anno in cui c'è stata la maggiore crescita assoluta nelle vendite rispetto all'anno precedente. Se ci sono più anni con la stessa crescita assoluta, restituire quello più grande. Se non c'è mai stata una crescita, la funzione restituisce il primo anno presente nel file. Ad esempio, se il file è quello di sopra ed **oggetto** vale '`Giubbotto`' allora deve restituire 2012, se invece **oggetto** vale '`Zaino`' allora deve restituire 2010 (primo anno presente nel file)

- **A_Ex6(fileName):** Scrivere una funzione che prende in ingresso una stringa **fileName** che denota il nome di un file csv nel formato di sopra e calcola in quale anno (numero intero) le vendite complessive sono state massime. Se ci sono 2 anni con le stesse vendite, si prenda l'anno più grande. Ad esempio, se il file è quello di sopra allora deve restituire 2010, anno in cui ci sono stati venduti in tutto 51 oggetti.

- **A_Ex7(fileName):** Scrivere una funzione che prende in ingresso un file di testo contenente solo lettere alfabetiche, spazi, cifre e caratteri newline ('\n') e restituisce la somma dei numeri presenti nel file. Ad esempio, se il file contiene:

  ```
  Paolo è andato a casa 3 volte
  ed ha mangiato 11 ciambelle ed una TORTA
  ```

  Allora il risultato è 14 (3+11) visto che questi sono i soli numeri presenti nel file.

- **A_Ex8(fileName):** Scrivere una funzione che prende in ingresso un file di testo contenente solo lettere alfabetiche, spazi, cifre e caratteri newline ('\n') e restituisce il numero della riga che contiene più caratteri alfabetici maiuscoli. Se ci sono più righe con lo stesso numero, restituisca quella di indice più grande. Si assuma che la prima riga abbia indice 1. Ad esempio, se il file è quello di sopra, allora il risultato è 2, poiché la seconda riga hanno più caratteri alfabetici maiuscoli della 1.

- **A_Ex9(fileName,squadra)** 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 ed il nome di una squadra e calcola il numero di punti ottenuti da quella squadra. 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 i punti in classifica della squadra **squadra**. 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
```

E la squadra è `Chelsea` la funzione deve restituire 3.

Esercitazione Python n. 9 -- 28 novembre 2023

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

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-4.5**
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 l'orario di fine dell'esercitazione

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 dei numeri delle righe in cui la parola 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 di parole **s,** composta da soli caratteri alfabetici minuscoli e spazi, e 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']}
```

- **Ex9(fileName,c,n):** completare la funzione Python in cui **fileName** è il nome di un file testo che contiene un dizionario**, c** è una vocale, **n** un intero. A partire dal file in input, la funzione assegna il dizionario alla variabile **d**. Il dizionario contiene le informazioni sui paragrafi del testo "I Malavoglia", con la seguente struttura {**k**:**valore**} dove **k** è l'intero corrispondente al numero di paragrafo del testo e **valore** è una lista con due elementi [<testo del paragrafo>, <dizionario frequenza vocali del paragrafo>]. Per esempio, se **fileName** è il file "dizionarioParagrafiMalavoglia.txt" presente nella cartella dell'esercitazione, il dizionario **d** è tale che: **d**[3]=['Giovanni Verga', {'a': 2, 'e': 1, 'i': 2, 'o': 1, 'u': 0}]. La funzione deve restituire la lista delle prime parole di ogni paragafo in cui ci sono più di **n** occorrenze di **c**, nell'ordine in cui compaiono nel dizionario.
  N.B.
  - Nel file da completare è già compresa l'istruzione che carica il dizionario a partire dal file e non va modificata.
  - Ci possono essere dei 'buchi' tra le chiavi del dizionario, corrispondenti a paragrafi vuoti. Per esempio, se il file in input è il file "dizionarioParagrafiMalavoglia.txt", la chiave 2 non esiste.
  - Non si devono fare distinzioni tra vocali minuscole e maiuscole.

Esercitazione Python n. 10 -- 5 Dicembre 2023

Obiettivo dell'esercitazione è di prendere confidenza con le espressioni regolari.

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-4.5**. Una volta avviata la macchina virtuale, svolgete gli esercizi così come indicato nel testo. Usate l'ambiente Spyder per svolgere gli esercizi. Ovviamente è possibile consultare il materiale didattico disponibile sulla pagina web del corso (https://classroom.google.com/u/1/c/NjIwOTY0ODk3MDAx). Si ricorda che le note relative alle lezioni Python possono essere lette con l'applicazione JupyterLab.

La consegna deve essere effettuata entro l'orario di fine dell'esercitazione.

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.

Fate attenzione che gli input siano richiesti all'utente UNO PER VOLTA e NELL'ORDINE RIPORTATO nell'esercizio e che le vostre stampe riportino a video i messaggi ESATTAMENTE nel formato atteso.

Ogni esercizio richiede che sia completata una funzione all'interno del file predisposto con lo stesso nome e, come visto a lezione, può essere testata eseguendo il file stesso, con i casi di test forniti. Notate infatti che ogni file incorpora del codice python per eseguire alcuni test sulla funzione. **NON modificate questo codice, ma SCRIVETE SOLO il contenuto della funzione. Non spostate i file dalla loro posizione e non create nuovi file. Si noti che, come in sede di esame, per la correzione verranno usati insiemi di dati di test diversi.**

Esercizi

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).

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 **A**ldin**o** **a**nnaspand**o** in giro che era andato al bar

la funzione deve restituire come risultato 1.

Se invece la funzione prende in input

Ho trovato delle **o**ss**a** **o**rs**a** ora **v**ad**o** **v**elat**o**

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.

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 **A**l**d**o **d**estin**a**ndo in giro che **e**r**a** **a**rr**e**stato 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 da**ll**a sta**ll**a a**ll**a 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'].

Esercitazione Python n. 11 -- 12 Dicembre 2023

Obiettivo dell'esercitazione è di prendere confidenza con le espressioni regolari.

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-4.5**. Una volta avviata la macchina virtuale, svolgete gli esercizi così come indicato nel testo. Usate l'ambiente Spyder per svolgere gli esercizi. Ovviamente è possibile consultare il materiale didattico disponibile sulla pagina web del corso (https://classroom.google.com/u/1/c/NjIwOTY0ODk3MDAx). Si ricorda che le note relative alle lezioni Python possono essere lette con l'applicazione JupyterLab.

La consegna deve essere effettuata entro l'orario di fine dell'esercitazione.

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.

Fate attenzione che gli input siano richiesti all'utente UNO PER VOLTA e NELL'ORDINE RIPORTATO nell'esercizio e che le vostre stampe riportino a video i messaggi ESATTAMENTE nel formato atteso.

Ogni esercizio richiede che sia completata una funzione all'interno del file predisposto con lo stesso nome e, come visto a lezione, può essere testata eseguendo il file stesso, con i casi di test forniti. Notate infatti che ogni file incorpora del codice python per eseguire alcuni test sulla funzione. **NON modificate questo codice, ma SCRIVETE SOLO il contenuto della funzione. Non spostate i file dalla loro posizione e non create nuovi file. Si noti che, come in sede di esame, per la correzione verranno usati insiemi di dati di test diversi.**

Esercizi

Ex1(file): Scrivere una funzione Python che prende in input un numero intero **n** e una lista di interi **l** e restituisce la somma dei risultati della divisione intera di **n** per ognuno degli interi nella lista, gestendo l'eccezione eventualmente generata dalla divisione per 0 in modo tale che l'esecuzione non sia interrotta e la divisione errata non abbia impatto sul risultato della somma. Per esempio, se **n**=5 e **l**=[1,5,0,2], la funzione deve restituire 8.

Ex2(file): Scrivere una funzione Python che prende in input il nome di un file csv, che contiene una tabella di interi con almeno una riga e due colonne, e restituisce due liste, la prima che contiene, per tutte le righe, in ordine, la somma dei valori contenuti nella riga, e la seconda che contiene, per tutte le colonne, in ordine, la somma dei valori contenuti nella colonna. Per fare questo, la funzione tenta di creare un array usando il modulo Numpy, e, se ottiene un'eccezione perché il file non esiste, restituisce due liste vuote, mentre se ottiene un'eccezione perché ci sono alcuni dati mancanti nel file, crea un dataframe usando il modulo Pandas.
N.B. Assumiamo che le colonne non abbiano intestazione. In questo caso, la funzione `load_csv` del modulo `Pandas` deve essere usata specificando il parametro `header=None`.

Ex3(m): Scrivere una funzione Python che, ricevendo in ingresso un array numpy quadrato nxn **m**, con n>=0, contenente gli interi 0 e 1 restituisca:

- ‘Principale’ se la diagonale principale (dall'angolo in alto a sinistra a quello in basso a destra) contiene più valori a 1 della diagonale secondaria (dall'angolo in alto a destra a quello in basso a sinistra)
- ‘Secondaria’ se la diagonale secondaria contiene più 1 di quella principale
- ‘Uguali’ se il numero di 1 è uguale.

Per esempio, ricevendo la matrice [[1,0], [1,1]] la funzione deve restituire ‘Principale’.

Ex4(m) Definizione: un elemento di una matrice si dice **sopraffatto verticale** se entrambi gli elementi sopra e sotto di lui sono strettamente maggiori del suo valore o **sopraffatto orizzontale** se entrambi gli elementi a destra e sinistra di lui sono strettamente maggiori del suo valore. Scrivere una funzione Python che, ricevendo in ingresso un array numpy restituisce una tupla con due valori, pari al numero dei sopraffatti verticali e orizzontali.
Per esempio, ricevendo la matrice [[1,1,1], [1,0,0], [1,1,1]] la funzione deve restituire (2,0).

Ex5(m): Definizione: una riga di una matrice si dice **crescente** se tutti i suoi elementi sono ordinati in modo strettamente crescente (da sinistra a destra). Scrivere una funzione Python che, ricevendo in ingresso un array numpy restituisce il numero delle righe crescenti.
Per esempio, ricevendo la matrice [[1,2,4], [1,0,0], [1,1,2] la funzione deve restituire 1.

Ex6(m): Scrivere una funzione Python che, ricevendo in ingresso un array numpy restituisce il numero delle righe ottenibili come somma di altre due righe **adiacenti**.
Per esempio, ricevendo la matrice [[1,2,4], [1,0,0], [2,2,4] la funzione deve restituire 1.

Ex7(anno,file): Scrivere una funzione Python che prende in input un anno, ed un nome di file (excel in formato xlsx) e legge il file scritto in lingua inglese che contiene informazioni su degli artisti con le seguenti colonne:

Artist ID	(Codice Artista)
Name	(Nome)
Nationality	(Nazionalità)
Gender	(Sesso)
Birth Year	(Anno di nascita)
Death Year	(Anno di morte, se manca l’artista è ancora in vita)

Conta quanti autori presenti nel file sono vivi nell’anno fornito ed hanno almeno 50 anni. NOTA BENE: alcune celle potrebbero essere vuote. Per verificare se un valore è NaN o meno si possono usare le funzioni Pandas isna() e notna()

Ex8(anno,area,file): Scrivere una funzione Python che prende in input un anno, un’area (in cm^2) ed un nome di file (excel in formato xlsx) e legge il file scritto in lingua inglese che contiene informazioni su delle opere artistiche con le seguenti colonne:

Artwork ID	(Codice opera)
Title	(Titolo opera)
Artist ID	(Codice Artista)
Name	(Nome)
Date	(Data di creazione opera)
Height (cm)	(Altezza in centimetri)
Width (cm)	(Larghezza in centimetri)

Conta quante opere presenti nel file sono state realizzate STRETTAMENTE PRIMA dell’anno fornito in input ed hanno un’area (calcolata moltiplicando Height e Width) strettamente minore dell’area fornita. NOTA BENE: alcune celle potrebbero essere vuote. Per verificare se un valore è NaN o meno si possono usare le funzioni Pandas isna() e notna()

Ex9(file1,file2): Scrivere una funzione Python che, prende in input due files, uno nel formato dell'esercizio 7 ed uno nel formato dell'esercizio 8 e calcola quante opere presenti nel file 2 sono state realizzate da autori presenti nel file1 NOTA BENE: Gli autori sono identificati dal Artist ID (Codice Artista). Per verificare se un valore è NaN o meno si possono usare le funzioni Pandas isna() e notna()