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.