

10.07.2017

Esame di Fondamenti Informatica per Ingegneria Gestionale - A.A. 2016/2017

Durata 1h45' - Compito B

Istruzioni (leggere attentamente)

Nota importante: la mancata osservanza delle seguenti regole può comportare la perdita di informazioni necessarie alla valutazione dell'esame.

Cartella di esame e registrazione dei dati dello studente

La cartella Esame contenente il compito da svolgere si trova sul Desktop. Entrare nella cartella e, prima di iniziare il compito, eseguire il programma registrazione.pyc. Inserire i dati personali fornendo (separatamente) Numero di Matricola, Cognome e Nome. Il programma genera un file studente.txt che *non deve essere modificato manualmente*. Verificare che i dati nel file studente.txt siano corretti; in caso di errore potete rieseguire il programma registrazione.pyc.

Svolgimento degli esercizi

Leggere attentamente il testo e risolvere gli esercizi proposti.

Per ogni esercizio avete una cartella EsercN che contiene un file dal nome B_ExN.py (dove N è il numero dell'esercizio). Questo è il file che dovrete modificare con la vostra soluzione. Non create nuovi file.

Per verificare la correttezza di un esercizio **DOVETE** usare il programma TestEx.pyc (basta cliccarci sopra 2 volte) che proverà la vostra soluzione con un certo numero di casi di test. **Si noti che per la correzione verranno usati insieme di dati di test diversi.**

E' possibile consultare la documentazione ufficiale del linguaggio Python, ma **non è possibile usare libri o appunti**. In caso di problemi tecnici chiedere ai docenti o ai tecnici del laboratorio.

Esercizi

- **B_Ex1(s) (8 punti)** Completare la funzione **B_Ex1(s)** che prende come input una stringa **s** e restituisce il numero di caratteri corrispondenti a cifre contando ogni cifra una sola volta.

Esempio: Se **s** = '22' allora la funzione deve restituire 1.

- **B_Ex2(n) (8 punti)** Completare la funzione **B_Ex2(n)** che prende in ingresso un numero **n** e restituisce *la lista ordinata di tutti i numeri primi da 2 a n inclusi utilizzando il procedimento del Crivello di Eratostene*. In base a questo criterio si crea un elenco di tutti i numeri naturali da 2 a n, detto setaccio. Dopodiché, *si deve aggiungere il primo numero del setaccio all'elenco dei numeri primi trovati, e si eliminano dal setaccio tutti i suoi multipli*. Si prosegue in questo modo fino ad esaurire i numeri nel setaccio. Ad esempio, se **n** = 7 la funzione deve restituire la lista [2, 3, 5, 7]. Si noti che se **n** è 0 o 1 la funzione deve restituire la lista vuota.

Nota: l'algoritmo può essere implementato semplicemente in modo iterativo

- **B_Ex3(files, s) (8 punti)** Implementare la funzione **B_Ex3(files, s)** che prende in ingresso una lista di stringhe (**files**) e una stringa. Gli elementi della lista sono nomi di file testo, privi di punteggiatura. La funzione deve restituire *la lista ordinata del numero di occorrenze di s in ciascuno dei file che appaiono in files*. Eventuali maiuscole devono essere ignorate, per cui ad esempio "Casa" e "casa" vanno considerate la stessa parola.

Esempio: si supponga che **files** = [“file1”, “file2”] e **s** = “panca”. Si supponga che **file1** contenga il testo:

“Sopra la panca la capra campa sotto la panca la capra muore”

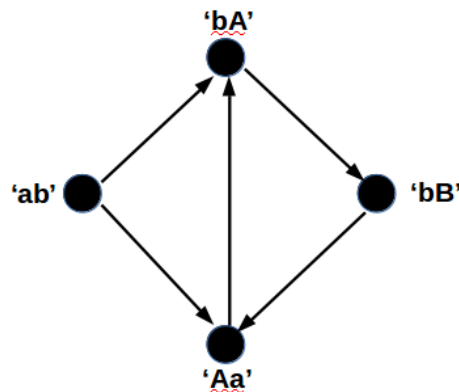
e che **file2** contenga il testo:

“Sotto la panca dorme un gatto”

La funzione deve restituire la lista [1, 2], perché la stringa appare 2 volte nel primo file e 1 nel secondo. Se invece **s** = “dorme”, la funzione dovrebbe restituire [1].

Nota: si ricordi che, data una stringa, il metodo `lower()` restituisce una copia della stringa con caratteri minuscoli.

- **B_Ex4(g, u) (8 punti)** Implementare la funzione **B_Ex4(g, u)** che (tra gli altri parametri) riceve in ingresso un dizionario **g** che rappresenta un grafo *diretto*, i cui nodi sono stringhe. Il parametro **u** è un nodo del grafo. La funzione deve restituire la *concatenazione in ordine alfabetico* delle etichette dei nodi che *non hanno* **u** come vicino.



Ad esempio, se il grafo fosse quello in figura, avremmo **g** = {‘bA’: [‘bB’], ‘bB’: [‘Aa’], ‘Aa’: [‘bA’], ‘ab’: [‘bA’, ‘Aa’]}. Se **u** fosse il nodo ‘bA’ la funzione dovrebbe restituire ‘bAbB’. Si noti che un nodo può essere vicino di se stesso, qualora abbia un cappio (non è questo il caso dell’esempio, per cui ‘bA’ *non ha* se stesso come vicino).