Esercitazione 3

22 novembre 2016

Termine per la consegna dei lavori: **martedì 29 novembre** ore **23.59**.

Istruzioni

I lavori dovranno essere salvati in una cartella che deve contenere tutto e solo ciò che volete venga consegnato e valutato (generalmente sarà sufficiente un file di testo per ognuno degli esercizi).
Per consegnare gli elaborati dovete raggiungere la cartella contenente i file da inviare in modalità terminale (`cd path_della_cartella`) e quindi eseguire il comando:

```
consegna consegna3
```

verrà visualizzata la lista di tutto ciò che è stato inviato.
Consegne successive (entro il termine per la consegna) sovrascriveranno le precedenti, verrà valutata solo l'ultima consegna sottomessa.
È obbligatorio che all'interno di ogni file sia riportato il vostro nome, cognome e numero di matricola (potete riportarli all'interno di una riga commento all'inizio del file, es: `#Mario Rossi 1234567`).

ATTENZIONE!

In questa esercitazioni non sono ammesse importazioni di moduli esterni. Esercizi risolti utilizzando moduli importati riceveranno il punteggio minimo. Python contiene delle **built-in functions**, che potete utilizzare e la cui lista si può trovare a questo indirizzo: `https://docs.python.org/2/library/functions.html`. Si sottolinea, comunque, che questa esercitazione può essere risolta utilizzando **solo** funzioni viste in laboratorio e ogni esercizio deve funzionare con qualsiasi input che rispetti la consegna.

Consiglio generale: usate i descrittori di lista!

Esercizio 1

Sia $n > 0 \in \mathbb{N}$, un intero positivo, generare la lista di tutte e sole le coppie di numeri primi gemelli strettamente minori di n. Esempio:

```
>>> n = 75
>>> #vostro codice ...
>>> #risultato atteso
[(3, 5), (5, 7), (11, 13), (17, 19), (29, 31), (41, 43), (59, 61), (71,
    73)]
```

Per maggiori informazioni sui primi gemelli `https://it.wikipedia.org/wiki/Numeri_primi_gemelli`.

Esercizio 2

Sia $n > 0 \in \mathbb{N}$, un intero positivo, generare la lista dei primi n numeri di fibonacci. Si assumono come valori iniziali della sequenza $F_0 = 0$ e $F_1 = 1$. **Non** sono permessi cicli, **non** è permessa la ricorsione e **non** sono permesse le funzioni. Esempio:

```
>>> n = 17
>>> #vostro codice ...
>>> #risultato atteso
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987]
```

Esercizio 3

Data una stringa di testo composta soli caratteri alfabetici minuscoli e spazi, crearne l'"indice analitico" in questo modo: creare un dizionario contenente come chiavi tutte le lettere alfabetiche contenute nel testo e ad ognuna di queste dovrà essere associato come valore un altro dizionario contenente come chiavi le parole che contengono quella lettera e come valore il numero di occorrenze di essa in quella parola.
Esempio:

```
>>> testo = "quarantotto bottoni rotti"
>>> #vostro codice ...
>>> #risultato atteso
{'a': {'quarantotto': 2},
'b': {'bottoni': 1},
'i': {'bottoni': 1, 'rotti': 1},
'o': {'bottoni': 2, 'quarantotto': 2, 'rotti': 1},
'n': {'bottoni': 1, 'quarantotto': 1}, 'q': {'quarantotto': 1},
'r': {'quarantotto': 1, 'rotti': 1}, 'u': {'quarantotto': 1},
't': {'bottoni': 2, 'quarantotto': 3, 'rotti': 2}}
```

Esercizio 4

Dato un intero $m \in [1, 10]$ generare la lista di numeri palindromi con un numero di cifre pari a m avente il massimo numero di cifre diverse.
ATTENZIONE: valori di m superiori a 7 potrebbero richiedere tempo di calcolo elevato.
Esempio:

```
>>> m = 5
>>> #vostro codice ...
>>> #risultato atteso
[10201, 10301, 10401, 10501, 10601, 10701, 10801, 10901, 12021, ...
```

Esercizio 5

Dato un testo in chiaro composto esclusivamente da caratteri alfabetici minuscoli (senza spazi) di **lunghezza pari** ed una chiave di cifratura k, scrivere uno script che implementa il cifrario di Hill. Il cifrario di Hill è un cifrario a sostituzione basato sull'algebra lineare. Nella funzione di cifratura, ogni lettera è per prima cosa codificata in numero: per semplicità codificheremo ogni lettera con la sua posizione all'interno dell'alfabeto: $a = 0, b = 1, c = 2, \ldots, z = 25$. Consideriamo il testo suddiviso in blocchi di lunghezza 2 e codifichiamo le lettere come descritto sopra ottenendo dei vettori $\in \mathbb{Z}_{26}^2$: ad ognuno di questi si applica la cifratura con una chiave $\in \mathbb{Z}_{26}^{2\times 2}$, semplicemente applicando la moltiplicazione matrice per vettore (modulo 26). Vediamo un semplice esempio: consideriamo il testo "help". I blocchi di lunghezza 2 che costituiscono questa stringa sono "he" e "lp". Applicando la codifica otteniamo i vettori:

$$\text{“he”} \to \begin{pmatrix} h \\ e \end{pmatrix} \to \begin{pmatrix} 7 \\ 4 \end{pmatrix}, \text{“lp”} \to \begin{pmatrix} l \\ p \end{pmatrix} \to \begin{pmatrix} 11 \\ 15 \end{pmatrix}.$$

Considerata la chiave $k \in \mathbb{Z}_{26}^{2\times 2}$:

$$k = \begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix},$$

calcoliamo

$$\begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix} \begin{pmatrix} 7 \\ 4 \end{pmatrix} = \begin{pmatrix} 7 \\ 8 \end{pmatrix},$$

$$\begin{pmatrix} 3 & 3 \\ 2 & 5 \end{pmatrix} \begin{pmatrix} 11 \\ 15 \end{pmatrix} = \begin{pmatrix} 0 \\ 19 \end{pmatrix}.$$

Da cui applicando la codifica dei caratteri all'indietro otteniamo:

$$\begin{pmatrix} 7 \\ 8 \end{pmatrix} \to \begin{pmatrix} h \\ i \end{pmatrix} \to \text{“hi”}, \begin{pmatrix} 0 \\ 19 \end{pmatrix} \to \begin{pmatrix} a \\ t \end{pmatrix} \to \text{“at”},$$

ottenendo la stringa cifrata "hiat". Si ricorda che le operazioni matrice per vettore sono state svolte **modulo 26**.

Quindi, data una chiave k sotto forma di lista di liste (definizione di matrice come vista in laboratorio) e una stringa t di lunghezza pari, restituire la sua cifratura secondo il cifrario di Hill, ad esempio:

```
>>> #matrice 2x2 scritta in formato lista di lista
>>> k = [[3,3],[2,5]]
>>> #stringa di lunghezza pari
>>> testo = "esempioditestoinchiaro"
>>> #vostro codice ...
>>> #risultato atteso
>>> cifrato = "ouwqrszrdhouveldbnyqpa"
```