Esercitazione 3

14 novembre 2017

Termine per la consegna dei lavori: **martedì 21 novembre 2017** ore **23.59**.

Istruzioni

I lavori dovranno essere salvati in file di testo per ognuno degli esercizi e consegnati tramite Moodle su sito `https://elearning.unipd.it/math`, sezione "**Matematica: Laurea Triennale**" corso di "**Laboratorio di Programmazione**" cliccando su "**Esecitazione3**". 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 *sexy* strettamente minori di n. Esempio:

```
>>> n = 45
>>> #vostro codice ...
>>> #risultato atteso
[(5,11), (7,13), (11, 17), (13, 19), (17, 23), (23, 29), (31, 37), (37,
    43)]
```

Per maggiori informazioni sui primi sexy `https://it.wikipedia.org/wiki/Numeri_primi_sexy`.

Esercizio 2

Sia $n > 0 \in \mathbb{N}$, un intero positivo, generare la lista dei primi n numeri di Lucas, la quale inizia con i valori $L_0 = 2$ e $L_1 = 1$. **Non** sono permessi cicli, **non** è permessa la ricorsione e **non** sono permesse le funzioni. Esempio:

```
>>> n = 17
>>> #vostro codice ...
>>> #risultato atteso
[2, 1, 3, 4, 7, 11, 18, 29, 47, 76, 123, 199, 322, 521, 843, 1364,
   2207]
```

Esercizio 3

Data una stringa di testo composta solo di caratteri alfabetici minuscoli e spazi, crearne l'"indice analitico" in questo modo: creare un dizionario contenente come chiavi tutte e sole le iniziali delle parole contenute nel testo e ad ognuna di queste dovrà essere associato come valore un altro dizionario contenente come chiavi le parole che iniziano con quella lettera e come valore una lista delle posizioni di tali parole nel testo.
Esempio:

```
>>> testo = "tre strette tazze dentro a tre tazze strette"
>>> #vostro codice ...
>>> #risultato atteso
>>> {'a': {'a': [4]},
>>>  's': {'strette': [1,7]},
>>>  't': {'tre': [0,5], 'tazze': [2,6]},
>>>  'd': {'dentro': [3]} }
```

(n.b.: la prima parola compare in posizione 0)

Esercizio 4

Dato un intero m generare la lista di tutti i numeri *one-step* palindromi minori o uguali a m. Un numero n è detto one-step palindromo se la somma con la sua "rappresentazione inversa" è palindroma.

Esempio: 56 è one-step palindromo perchè 56+65=121 che è palindromo. Nota che 65 è la rappresentazione inversa di 56.
Esempio:

```
>>> m = 1000
>>> #vostro codice ...
>>> #risultato atteso
[0, 1, 2, 3, 4, 10, 11, 12, 13, ... , 900, 902, 910, 920, 930, 940]
```

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"
```