Esercitazione 4

17 novembre 2015

Termine per la consegna dei lavori: **martedì 24 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, d'ora in poi sarà **obbligatoriamente** un file **eseguibile** con estensione `.py` per ognuno degli esercizi. Controllate che l'esecuzione del comando:

```
python <nome_file>.py
```

per ognuno degli esercizi produca l'output desiderato.

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 consegna4
```

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 (riportati all'interno di una riga commento all'inizio del file, es: `#Mario Rossi 1234567`).

ATTENZIONE!

Gli unici moduli importabili ammessi in questa esercitazione sono i moduli `math` e `random`. Esercizi risolti utilizzando altri 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`.

Esercizio 1: Numeri perfetti

Un naturale $n \geq 2$ è detto *perfetto* se è uguale alla somma dei suoi divisori naturali, **escluso** se stesso.
Esempio: $6 = 1 + 2 + 3$ è il più piccolo numero perfetto.
Scrivere un programma che cerchi i primi 4 numeri perfetti visualizzandoli a video. Riuscite a trovare anche il quinto in tempi ragionevoli?

Per maggiori informazioni sui numeri perfetti: `https://en.wikipedia.org/wiki/Perfect_number`.

Esercizio 2: Triangolo di Tartaglia

Scrivere un programma che generi e visualizzi **ben formattate** (a triangolo), le prime 15 righe del triangolo di Tartaglia. *Esempio (prime 5 righe):*

```
      [1]
    [1, 1]
   [1, 2, 1]
 [1, 3, 3, 1]
[1, 4, 6, 4, 1]
```

Per maggiori informazioni sul triangolo di Tartaglia:
`http://it.wikipedia.org/wiki/Triangolo_di_Tartaglia`.

Esercizio 3: Monte Carlo π

Sia P un punto di coordinate $(x, y) \in [0, 1]^2$, con x, y scelti casualmente. Se il punto P è contenuto nel quarto di circonferenza con centro in (0, 0) e raggio 1, allora lo classificheremo come positivo (punti rossi in Figura 1), altrimenti come negativo (punti blu in Figura 1). Generare una serie di punti, come appena descritto, fino a che non vale:

$$\left| 4 \cdot \frac{\#(\oplus)}{\#(\oplus) + \#(\ominus)} - \pi \right| < 10^{-8},$$

dove $\#(\oplus)$ e $\#(\ominus)$ rappresentano rispettivamente il numero di punti positivi e negativi generati. Stampare a video l'approssimazione di π ottenuta (ovvero, $4 \cdot \frac{\#(\oplus)}{\#(\oplus)+\#(\ominus)}$) e il numero di punti generati per raggiungerla. Ripetendo l'esperimento il numero di generazioni cambia?

Per maggiori informazioni sul metodo Monte Carlo:
`https://it.wikipedia.org/wiki/Metodo_Monte_Carlo`.

Esercizio 4: Cifrario di Vigenère

Il cifrario di Vigenère è un semplice esempio di cifrario polialfabetico. Può essere considerato una generalizzazione del cifrario di Cesare (che è monoalfabetico), dove la chiave rappresentata da un valore intero viene sostituita da una parola. Vediamo un esempio: consideriamo il testo in chiaro "rapportoimmmediato", e la chiave "verme". Allineiamo

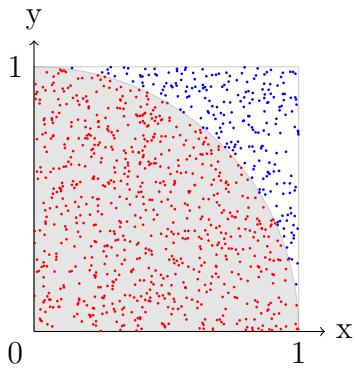

Figura 1: Monte Carlo π approximation: esempio con 1000 punti generati causalmente.

la chiave, **ripetuta**, al testo in chiaro, ottenendo:

testo	r	a	p	p	o	r	t	o	i	m	m	e	d	i	a	t	o
chiave	v	e	r	m	e	v	e	r	m	e	v	e	r	m	e	v	e

Ad ogni carattere in chiaro si applica il cifrario di Cesare (come visto in laboratorio) considerando come chiave il numero associato al carattere della chiave corrispondente. Ad esempio, il primo carattere in chiaro 'r' e il carattere della chiave associato 'v': 'r' corrisponde a 17, mentre 'v' corrisponde a 21, applicando Cesare si ottiene 'm' ($v = 21, r = 17 \rightarrow (17 + 21)\ mod\ 26 = 12 = m$). Si ricorda che come al solito si assume l'associazione $a = 0, b = 1, c = 2, \ldots, z = 25$.

Scrivere un programma che data un testo in chiaro e una chiave applica il cifrario di Vigenère. Il testo in chiaro si assume essere scritto con caratteri alfabetici minuscoli e **deve** contenere almeno uno spazio. Al momento della cifratura lo spazio deve essere preservato come tale (**non** deve essere cifrato). La chiave **non** deve contenere spazi ed è quindi composta da soli caratteri alfabetici minuscoli.
Esempio:

```
#chiave come stringa
k = "verme"
#testo in chiaro
testo = "rapporto immediato"
#vostro codice ...
#risultato atteso
cifrato = "megbsmxf uqhiuueos"
```

Per maggiori informazioni sul cifrario di Vigenère:
`https://it.wikipedia.org/wiki/Cifrario_di_Vigen%C3%A8re`.

Esercizio 5: Mastermind

Scrivere un programma che generi casualmente un codice composto da 4 cifre comprese tra 0 e 9, senza comunicare all'utente la sequenza ottenuta. A questo punto all'utente viene chiesto di fare dei tentativi per indovinare la sequenza generata (inserendo quindi

in input 4 numeri compresi tra 0 e 9, **ignorare** input non validi).
Dopo ogni tentativo, il programma fornisce degli aiuti comunicando:

1. il numero di cifre corrette al posto giusto, cioè le cifre del tentativo che sono effettivamente presenti nel codice al posto corretto.

2. il numero di cifre corrette al posto sbagliato, cioè le cifre del tentativo che sono effettivamente presenti nel codice, ma non nel posto corretto.

N.B. **Non** bisogna comunicare **quali** cifre sono giuste o sbagliate, ma solo **quante**.
L'utente vince la partita se riesce ad indovinare il codice entro un numero di tentativi predeterminati (solitamente 9, riuscite ad avvisare l'utente dei tentativi rimasti?).

Attenzione: come gestire valori uguali

Consideriamo i seguenti esempi che chiariscono come gestire alcuni casi ambigui:

- Numero generato: 1234, tentativo: 4407
  il programma dovrà restituire 0 valori giusti in posizione corretta e 1 (viene contato un solo 4) in posizione errata;

- Numero generato: 1244, tentativo: 4607
  il programma dovrà restituire 0 valori giusti in posizione corretta e 1 (viene contato un solo 4) in posizione errata;

- Numero generato: 1244, tentativo: 6074
  il programma dovrà restituire 1 valore giusto in posizione corretta e 0 (non viene contato di nuovo il 4) in posizione errata;

Per maggiori informazioni su Mastermind:
`http://it.wikipedia.org/wiki/Mastermind`.