# Esercitazione 4

29 novembre 2016

Termine per la consegna dei lavori: **martedì 6 dicembre** 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: Anti-Primi

Un naturale $n \geq 1$ è detto *anti-primo* (o "fortemente composto") se il numero dei suoi divisori è maggiore del numero di divisori di ogni numero più piccolo ($\geq 1$).
*Esempio*: i divisori del 6 sono 1,2,3 e 6 ovvero sono 4 che è maggiore del numero di divisori di tutti i numeri dall'1 al 5.

Scrivere un programma che cerchi, e stampi a video, tutti gli anti-primi $\leq 10000$.

## Esercizio 2: La Congettura di Collatz

Sia $n \geq 1$ un intero positivo. Si consideri la funzione:

$$f(n) = \begin{cases} n/2 & \text{se } n \text{ pari} \\ 3n+1 & \text{altrimenti} \end{cases}$$

e si crei la sequenza

$$a_i = \begin{cases} n & \text{se } i = 0 \\ f(a_{i-1}) & \text{altrimenti} \end{cases}$$

La congettura di Collatz afferma che:

$$\forall n \in \mathbb{N}_+ \; \exists i \in \mathbb{N} \mid a_0 = n \Rightarrow a_i = 1.$$

*Esempio*: iniziando con n = 6, otteniamo la successione 6, 3, 10, 5, 16, 8, 4, 2, 1.

Scrivere un programma che ritorni il numero $\leq 100000$ che produce la successione più lunga.

Per maggiori informazioni sulla congettura di Collatz:
`https://it.wikipedia.org/wiki/Congettura_di_Collatz`.

## Esercizio 3: Il quadrato nascosto

Creare un programma che ritorni tutti e soli gli interi positivi tali che il loro quadrato sia della forma 1_2_3_4_5_6 dove _ indica una qualsiasi singola cifra.
*Esempio*: $135254 \Rightarrow 135254^2 = \mathbf{1}8\mathbf{2}9\mathbf{3}6\mathbf{4}4\mathbf{5}\mathbf{1}\mathbf{6}$.

## Esercizio 4: L'ago di Buffon

Supponiamo di avere un motivo a strisce parallele (per esempio un pavimento in parquet o un tappeto a strisce), tutte della stessa larghezza, e facciamoci cadere casualmente un ago. Qual è la probabilità che l'ago si trovi su una linea fra le due strisce?

La soluzione evidenzia una dipendenza da $\pi$ e quindi può essere usato come metodo Monte Carlo per calcolare pi greco.

In particolare, chiamiamo $L$ la lunghezza dell'ago e $D$ la larghezza delle strisce con $L \leq D$ allora vale che:

$$\pi = \frac{2L}{D}\frac{1}{p(+)},$$

dove $p(+)$ indica la probabilità che l'ago sia in una linea fra le strisce.

E' possibile stimare $p(+)$ eseguendo una simulazione Monte Carlo, ovvero si simula la caduta dell'ago per $N$ volte e chiamando $n$ il numero di volte che l'ago si trova su una linea tra le strisce possiamo approssimare $p(+)$ con:

$$p(+) \approx \frac{n}{N}.$$

Di conseguenza possiamo sperimentalmente approssimare pi greco con:

$$\pi \approx \frac{2L}{D}\frac{N}{n}.$$

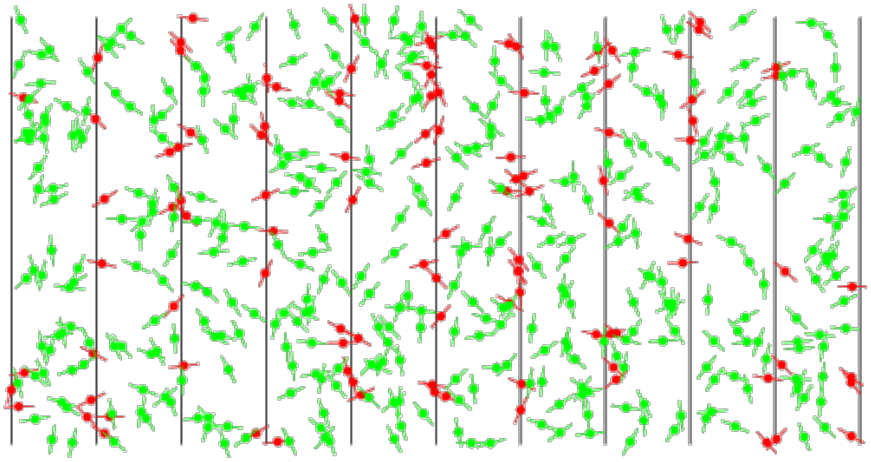

Figura 1: Esempio di simulazione Monte Carlo del problema dell'ago di Buffon. In rosso sono evidenziati gli aghi a cavallo di due strisce.

Scrivere un programma che esegue una simulazione Monte Carlo dell'ago di Buffon con $N \geq 10000$ iterazioni. Osservare come aumentando $N$ l'approssimazione sia tendenzialmente sempre più accurata.

Per maggiori informazioni sul problema dell'ago di Buffon:
`https://it.wikipedia.org/wiki/Ago_di_Buffon`.

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

## 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`.