Esercitazione 9

13 Gennaio 2015

Termine per la consegna dei lavori: **martedì 20 gennaio** 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. In questo caso, un file **eseguibile** con estensione `.py`. Controllate che l'esecuzione del comando:

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

produca l'output desiderato.
Per consegnare gli elaborati dovete raggiungere la cartella contenente il file da inviare in modalità terminale (`cd path_della_cartella`) e quindi eseguire il comando:

```
consegna consegna9
```

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!

Non ci sono particolari restrizioni sui moduli importabili. Per non subire penalizzazioni **i moduli importati dovranno essere motivati nei commenti**. Python contiene inoltre 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: Game of Life

Game of Life è un automa cellulare. Esso rappresenta un caso molto famoso di come regole molto semplici possono portare alla creazione di fenomeni complicati e inaspettati. In questo esercizio dovrete realizzare una (semplice) implementazione del gioco *Game of Life* utilizzando quanto visto durante il corso.

Si tratta in realtà di un gioco senza giocatori, intendendo che la sua evoluzione è determinata dal suo stato iniziale, senza necessità di alcun input da parte di giocatori umani. Si svolge su una griglia di caselle quadrate (celle) N per M. Questa griglia è detta mondo. Ogni cella ha 8 vicini, che sono le celle ad essa adiacenti, includendo quelle in senso diagonale. Ogni cella può trovarsi in due stati: viva o morta (o accesa e spenta, on e off). Lo stato della griglia evolve in intervalli di tempo discreti. Gli stati di tutte le celle in un dato istante sono usati per calcolare lo stato delle celle all'istante successivo. Tutte le celle del mondo vengono quindi aggiornate simultaneamente nel passaggio da un istante a quello successivo: passa così una generazione.

Le transizioni di stato dipendono unicamente dal numero di vicini vivi:

- Una cella morta con esattamente 3 vicini vivi nasce, diventando viva
- Una cella viva con 2 o 3 vicini vivi sopravvive; altrimenti muore (per isolamento o sovraffollamento)

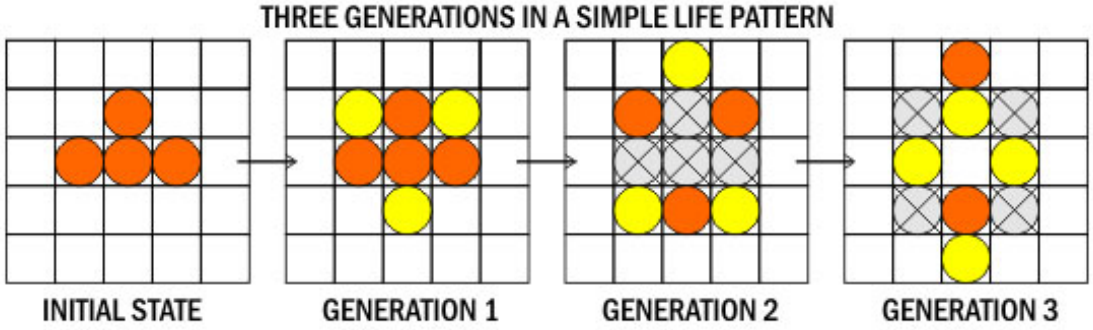

Figura 1: Esempio di evoluzione di alcune celle.

Una spiegazione dettagliata del gioco è presente in Wikipedia:

`http://it.wikipedia.org/wiki/Gioco_della_vita`.

Cosa vi è richiesto di fare:

- Creare uno script che simuli perfettamente *Game of Life* **e che avvii un esempio di gioco alla sua esecuzione** (eseguendo lo script deve avviarsi automaticamente una simulazione)
- Utilizzare almeno due classi, una che rappresenti l'oggetto cella e una che rappresenti il concetto di mondo in cui le celle si evolvono

- Per ogni generazione mostrare lo stato del mondo (stampando a schermo la griglia del mondo in modo tale che si capisca quali celle sono vive e quali morte). **Consiglio**: utilizzare la funzione `sleep(seconds)` del modulo `time` per bloccare per un certo lasso di tempo l'esecuzione del programma (e permettere quindi all'utente di vedere la stampa della griglia del mondo).

- Permettere la lettura della griglia della situazione iniziale del mondo da file

Il resto è a vostra discrezione.
Consiglio per testare il corretto funzionamento:

- Verificare l'evoluzione delle composizioni iniziali di celle vive più famose. Alcuni esempi:

`http://en.wikipedia.org/wiki/Conway's_Game_of_Life#Examples_of_patterns`

Figura 2: Esempio di possibile visualizzazione del mondo.