Informatica

Appunti dal laboratorio 4

Esercizio

- Scrivere un programma Python che estragga in continuazione 6 numeri a caso compresi tra 0 e 9 (estremi inclusi) finché non si ottiene una sestina di numeri tutti uguali tra di loro. Il programma deve mostrare tutte le estrazioni su schermo ed eseguirne il conteggio*.

*In data 19 novembre 2015 è stato raggiunto l'insuperabile (letteralmente: questo record può essere solo eguagliato) risultato di avere sei volte "3" al primo colpo.

Numeri pseudocasuali

- In un computer, che per sua natura è completamente deterministico, non esistono numeri estratti a caso.
- Esistono, però, funzioni matematiche parametriche che sono in grado di generare sequenze di numeri che agli occhi dell'utente appaiono come casuali, ma che in realtà sono deterministici.
- Numeri di questo tipo si dicono "pseudocasuali"
- In particolare, un tipico parametro che concorre alla loro generazione è l'istante (con precisione al decimillesimo di secondo) in cui viene lanciato il programma (valore segnato dall'orologio del computer, chiamato "clock")

Random

- Per usare i numeri pseudocasuali in un programma bisogna importare il modulo “random”
- Importare un modulo vuol dire rendere disponibili tutti quei comandi precedentemente scritti da altri programmatori e inseriti nell’ambiente di programmazione Python
- Il fatto che non tutti i moduli siano resi disponibili in maniera automatica è dovuto a motivazioni di prestazione
- Le funzioni del modulo random si invocano con la dot notation
- In particolare usiamo la funzione randint(x,y) che fornisce un numero intero compreso tra x e y (estremi inclusi)

While True + break

- Quando si deve ripetere un'istruzione un numero non prevedibile di volte, ma dipendente da una condizione, la si inserisce tipicamente in ciclo while
- Potremmo pensare di scrivere un codice del genere

```
while (i 6 numeri estratti non sono tutti uguali):
    estrai 6 numeri a caso
    aumenta il contatore
    stampa i 6 numeri su schermo
```

- Questo modo di procedere ha un problema: la prima volta che la condizione del while viene controllata, non è stata fatta ancora alcuna estrazione, quindi la condizione non ha valore di verità
- Ovviamente un essere umano è in grado di capire che la prima volta il controllo va omesso, ma un computer no
- Ecco quindi che può venire utile il ciclo while True, ovvero un loop in cui si entra in maniera incondizionata, visto che la condizione ha valore di verità "vero"
- Dal loop, però, a un certo punto bisogna uscire, altrimenti il programma non terminerebbe mai
- L'esecuzione di un'istruzione break provoca l'uscita dal ciclo in cui tale istruzione è inserita

- Il codice diventerebbe quindi:

```
while True:
        estrai 6 numeri a caso
        aumenta il contatore
        stampa i 6 numeri su schermo
        if (i 6 numeri sono tutti uguali):
                break
```

- Si entra nel ciclo iterativo in ogni caso e vi si rimane finché non viene eseguita l'istruzione break
- L'istruzione break viene eseguita nel caso in cui è vero che i 6 numeri estratti sono tutti uguali

While con variabile booleana

- Un modo alternativo di procedere è quello di usare una variabile booleana (che assume solo i valori “True” e “False”) che possiamo chiamare “vinto” per indicare se sono stati estratti 6 numeri uguali o meno
- Il valore iniziale di tale variabile sarà naturalmente False:

```
vinto = False
while not vinto:
        estrai 6 numeri a caso
        aumenta il contatore
        stampa i 6 numeri su schermo
        if (i 6 numeri sono uguali)
                vinto = True
```

La condizione di uscita dal while

- Come si specifica in Python la condizione “i 6 numeri estratti sono uguali”?
- Un modo molto semplice è quello di fare una catena di uguaglianze
- Tale soluzione, però, non è scalabile: se anziché estrarre 6 numeri dovessimo estrarne 20, o 100, la condizione diventerebbe molto lunga da scrivere
- Per mantenere la scalabilità del programma anziché usare 6 variabili e una sequenza di uguaglianze per confrontarle, introduciamo una lista e un ciclo for per confrontare i valori al suo interno

Soluzione

```
from __future__ import print_function
import random
count = 0
vinto = False
while not vinto:
        giocata = []
        for i in range(0,6):
                giocata.append(random.randint(0,9))
        count = count + 1
        print("giocata no. ", count, ":", giocata)
        vinto = True
        for j in range(0,5):
                vinto = vinto and (giocata[j]==giocata[j+1])
```

La condizione di uscita dal while

- Dopo l'estrazione dei 6 numeri e la loro visualizzaione, la variabile "vinto" viene posta a True, non perché siamo sicuri che i 6 numeri estratti siano tutti uguali, ma appunto per controllare se questa condizione è vera
- "vinto" viene messa in congiunzione con la condizione giocata[j]==giocata[j+1], con j che varia da 0 a 4 (estremi inclusi)
- se tutte le coppie consecutive di valori in giocata sono uguali, "vinto" uscirà dal ciclo for ancora col valore True
- se anche una sola coppia è formata da valori diversi, "vinto" assume il valore False e tale rimarrà fino alla fine del ciclo (False in congiunzione con qualunque condizione dà False)

Esercizi

- Scrivere un programma Python che chieda in input all'utente una sestina di numeri compresi tra 0 e 9 e gli mostri quante estrazioni sono state necessarie per fare uscire tale sestina
- Scrivere un programma Python che chieda in input all'utente una sestina di numeri da giocare con le regole del Superenalotto, faccia un'estrazione di 6 numeri (tra 1 e 90, tutti diversi tra loro), e dia il punteggio ottenuto dalla giocata dell'utente