Appunti di informatica

Lezione 11

anno accademico 2016-2017

Mario Verdicchio

Esercizio

- Scrivere un programma Python che simuli il destino di una persona che si gioca a dati tutto il patrimonio.
- Il programma chiede in input l'ammontare della cifra che la persona vuole giocare
- Poi inizia una sequenza di lanci di 2 dadi:
  - se il totale dei dadi è 7 il giocatore vince 4 euro che vengono aggiunti al suo ammontare
  - altrimenti il giocatore perde 1 euro che viene sottratto al suo ammontare
- I lanci di dado continuano finché il giocatore non perde tutto
- Il programma mostra l'esito di ogni lancio di dado, il numero di lanci che sono stati necessari per dilapidare il patrimonio del giocatore, e la massima cifra raggiunta dal suo ammontare durante la sequenza di lanci

Soluzione

```
from __future__ import print_function
import random

tot = input("quanti euro vuoi giocare a dadi?")
n = 0
max = tot

while (tot > 0):
    n = n + 1
    d1 = random.randint(1,6)
    d2 = random.randint(1,6)
    print("giocata no.", n, ":", "[", d1, "]", "[", d2, "]")
    if d1 + d2 == 7:
        tot = tot + 4
        print("hai vinto 4 euro e i tuoi soldi ammontano ora a:", tot, "euro")
        if tot > max:
            max = tot
    else:
        tot = tot - 1
        print("hai perso 1 euro e i tuoi soldi ammontano ora a:", tot, "euro")

print("avevi raggiunto la somma di", max, "euro ma dopo", n, "giocate hai perso tutto")
```

Esercizio

- Newton scoprì un algoritmo per approssimare la radice quadrata di un numero x
- Si inizia con una stima di tale valore inizializzata a 1
- Newton dimostrò che, data una stima della radice quadrata di x, una stima migliore è data dalla media tra la vecchia stima e il rapporto tra x e la vecchia stima
- Scrivere un programma in Python che, fissata una soglia di tolleranza (la massima differenza in valore assoluto tollerata tra il quadrato della stima della radice quadrata di x e x stesso), chiede all'utente il valore di x e ne calcola la radice quadrata con il metodo descritto, iterando il calcolo di una nuova stima finché non si scende sotto la soglia di tolleranza

Numeri interi vs. numeri con la virgola

- In Python 2.6 il trattamento dei numeri è automatico: se nell'espressione di calcolo ci sono solo valori interi, tutte le operazioni sono considerate tra numeri interi (come ad esempio la divisione), mentre se c'è anche un solo valore con la virgola coinvolto in un'operazione, essa viene considerata tra numeri razionali)
- In Python 3.5, invece, è il programmatore a specificare esplicitamente se desidera una divisione intera (//) oppure una divisione con i risultati con la virgola (/)

Soluzione

```
from __future__ import print_function

while True:
    x = input("inserisci un numero positivo")
    if x > 0:
        break

tol = 0.0001
sti = 1.0

while True:
    sti = (sti + x/sti)/2
    dif = x - sti * sti
    if dif < 0:
        dif = -dif
    if dif <= tol:
        break

print("la radice quadrata di", x, "e':", sti)
```

Esercizio

- Leibniz scoprì che π (pi greco) può essere approssimato nel seguente modo:

  $\pi/4 = 1 - 1/3 + 1/5 - 1/7 + 1/9 - 1/11 + 1/13\ldots$

- Scrivere un programma in Python che chiede all'utente con quanti termini vuole approssimare π e ne calcola il valore secondo l'uguaglianza scoperta da Leibniz

Soluzione

```
from __future__ import print_function


n = input("Quanti termini vuoi per approssimare pi greco?")
p = 0
for i in range (0,n):
    if i%2 == 0:
        p = p + 1.0/(2*i + 1)
    else:
        p = p - 1.0/(2*i + 1)

print(4*p)
```

Il modulo math

- Importare il modulo “math” permette di usare numerose funzioni matematiche, tra cui la potenza:

  math.pow(x,y)

  restituisce il valore x^y

Soluzione alternativa usando math

```
from __future__ import print_function
import math

n = input("Quante termini vuoi per approssimare pi greco?")
p = 0
for i in range (0,n):
        p = p + math.pow(-1,i)*1.0/(2*i + 1)

print(4*p)
```