LA PIOGGIA

Esempio di simulazione con Python

Installato Python e avviata la IDLE, ci si trova nella shell (ambiente interattivo in cui viene immediatamente eseguita una qualsiasi istruzione), ma ci si può spostare nella finestra di edit del codice (scegliendo nel menu della shell la voce file-nuovo) così da poter scrivere le istruzioni con molte agevolazioni per il programmatore, come ad esempio il diverso colore per i vari elementi del linguaggio assegnato in automatico.

Nell'applicazione Python che andiamo a costruire verranno utilizzate le seguenti librerie:
- *turtle* per la rappresentazione delle gocce scegliendo la forma circolare dell'oggetto grafico (circle)
- *random* per la generazione dei numeri casuali
- *math* per poter applicare la funzione che restituisce la radice quadrata nel calcolo della distanza tra una goccia e la successiva (sqrt)
- *time* per definire una pausa nell'esecuzione del codice così da simulare il tempo tra una goccia e la successiva (sleep)

Quindi, le prime istruzioni del codice Python devono essere proprio quelle necessarie per importare le suddette librerie:

```
import turtle
import random
import math
import time
```

Ora devono essere richiesti in input i parametri di simulazione:

```
print ('imposta i parametri caratteristici della pioggia che vuoi simulare')

# la dimensione massima della goccia
max = int(input('digita la dimensione massima della goccia '))

# la distanza minima tra le gocce
dist = int(input('digita la distanza minima tra le gocce '))

# il tempo massimo di attesa tra la caduta di una goccia e l'altra
```



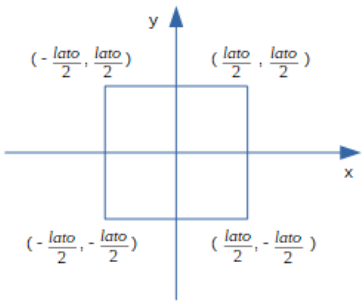
```
attesa = int(input("digita il tempo  massimo di attesa tra una goccia e l'altra (in sec) "))
```

NB
Le frasi che iniziano col simbolo # sono dei commenti per documentare il codice affinchè risulti più leggibile; la funzione *input* permette di acquisire un dato da tastiera che sarà richiesto dopo la frase che compare tra le parentesi della funzione stessa; la funzione *int* trasforma in numero intero quel dato. In alternativa a *int* è possibile utilizzare la funzione *float* per trasformare il dato in un numero reale.

Per rendere più generiche possibile le caratteristiche della pioggia da simulare, si può calcolare in modo casuale la dimensione minima della goccia, in funzione della dimensione massima acquisita in input; di seguito viene fissato il lato del quadrato che rappres
enterà la superficie di caduta delle gocce; anche tale valore potrebbe essere richiesto in input.

```
# calcolo in modo casuale la dimensione minima della goccia e la mostro
min = random.random()*max
print ("la dimensione minima della goccia sara' ",min)

# scelgo un valore per il lato del quadrato su cui far cadere la pioggia
lato = 300
```

Si procede quindi disegnando il quadrato che rappresenta il campo di osservazione della pioggia caduta, mediante opportune funzioni della libreria *turtle*; in questa applicazione, il quadrato sarà centrato rispetto all'origine del piano cartesiano su cui si possono muovere gli oggetti grafici della libreria *turtle*.




```
# disegno il quadrato
lato = lato /2       # per centrare il quadrato sull'origine degli assi
a = turtle.Turtle()  # definisco l'oggetto a di tipo turtle
a.hideturtle()       # nascondo a
a.penup()            # sollevo la penna perchè non lasci una traccia nel movimento
a.goto(-lato,-lato)  # la sposto nel vertice in basso a sinistra del quadrato
```



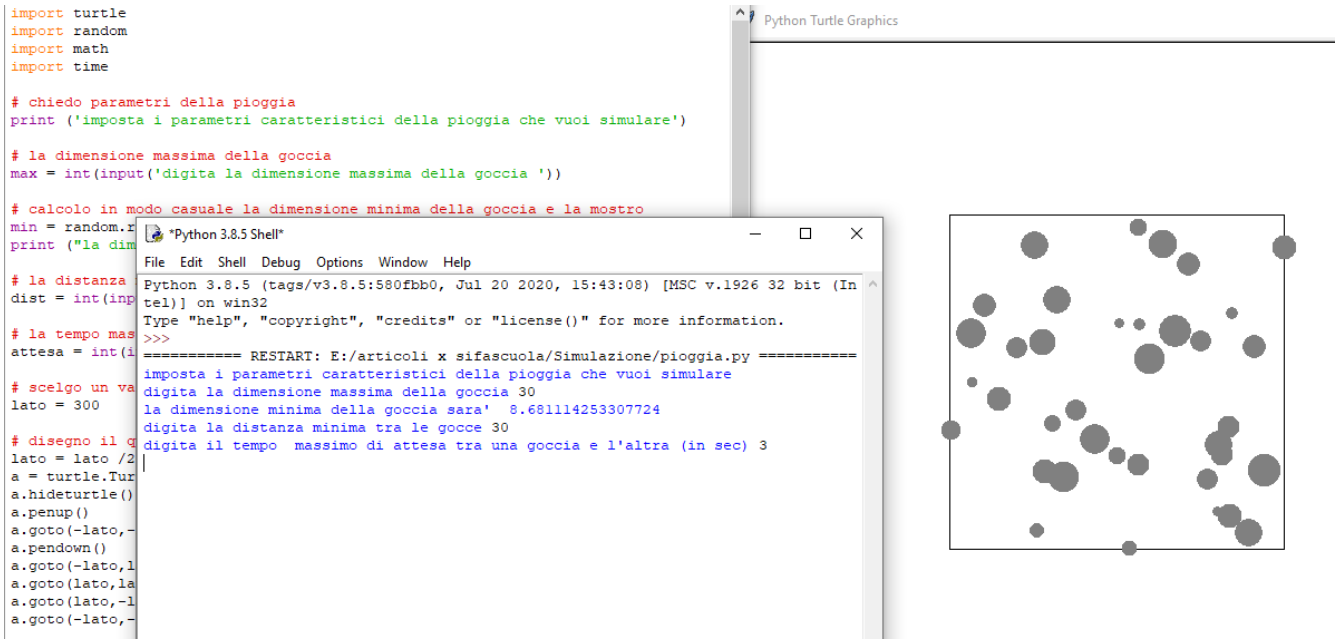
```
a.pendown()          # abbasso la penna
a.goto(-lato,lato)   # la sposto nel vertice in alto a sinistra del quadrato
a.goto(lato,lato)    # la sposto nel vertice in alto a destra del quadrato
a.goto(lato,-lato)   # la sposto nel vertice in basso a destra del quadrato
a.goto(-lato,-lato)  # la sposto nel vertice in basso a sinistra del quadrato
```

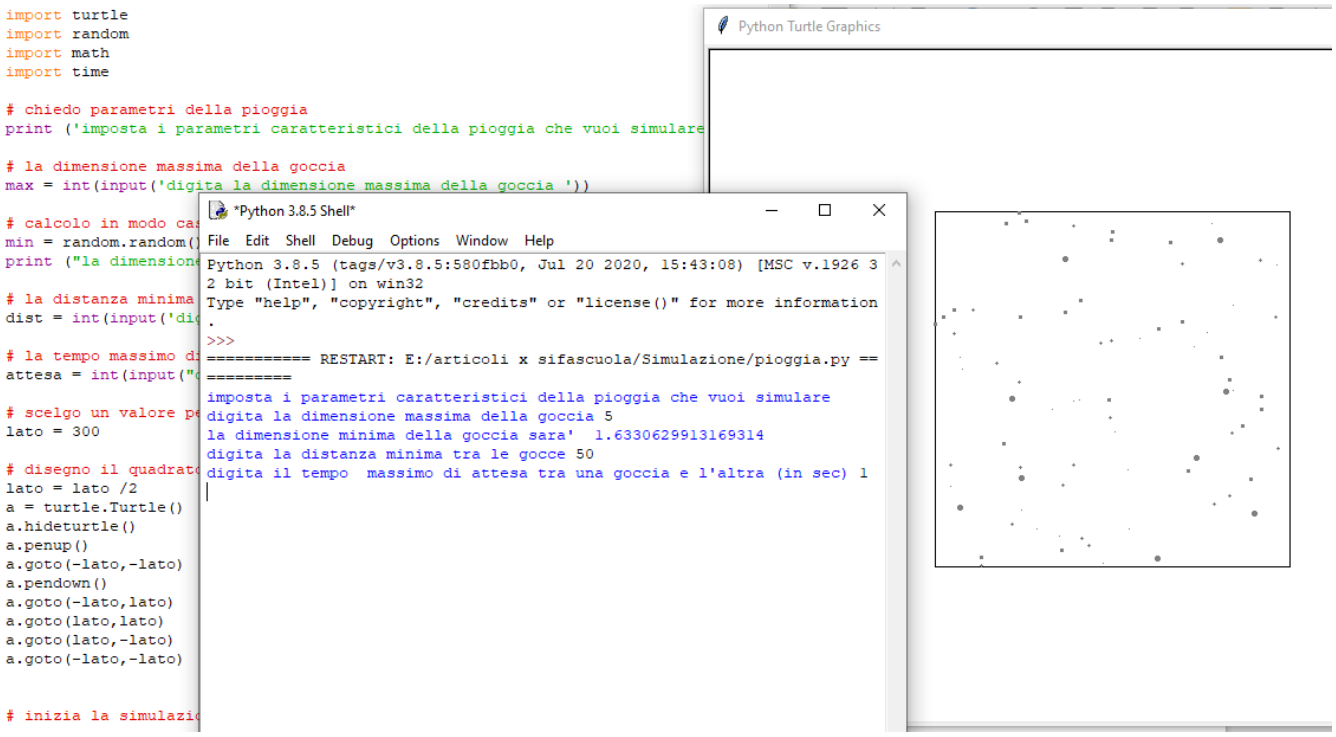
Ora inizia la parte di codice che realizza la vera e propria simulazione:

```
a.penup()
# creo un clone della turtle a, denominandolo b
b = a.clone()
# calcolo le coordinate casuali di b
x = random.random()*2*lato-lato
y = random.random()*2*lato-lato
# sposto b nel punto di coordinate (x,y)
b.goto(x, y)
# b 'timbra' un cerchio di colore grigio di dimensione casuale tra min e max
b.dot(min + random.random()*(max-min),'gray')

# faccio iniziare un ciclo infinito per le gocce successive

while True:
    # calcolo la posizione casuale della nuova goccia
    x1 = random.random()*2*lato-lato
    y1 = random.random()*2*lato-lato
    # se "abbastanza" distante dalla precedente faccio cadere la nuova goccia
    if math.sqrt((x-x1)**2+(y-y1)**2) > dist:
       b = a.clone()
       b.goto(x1, y1)
       # aspetto un tempo casuale prima di farla cadere
       time.sleep(random.random()*attesa)
       b.dot(min + random.random()*(max-min),'gray')
       # riporto nelle variabili x e y le coordinate di questa goccia
       x = x1
       y = y1
```

NB gli spazi che precedono le righe interne al ciclo *while* e alla struttura *if* sono essenziali e realizzano la cosiddetta *indentazione* la quale consente di individuare senza ombra di dubbio le linee di codice incluse in quella particolare struttura. La scelta di utilizzare un ciclo infinito offre la possibilità all'utente di questa applicazione di decidere autonomamente quando interrompere la simulazione e quindi di effettuare le proprie personali osservazioni.

Seguono due prove di simulazione della pioggia con diverse scelte dei parametri; si distinguono chiaramente la shell di Python che mostra l'interfaccia utente costruita nel nostro codice per la richiesta dei parametri iniziali, la finestra grafica con la rappresentazione degli oggetti turtle (gocce di pioggia realizzate con punti grigi), e la finestra di edit del codice che, come detto, colora in automatico con colori diversi i vari elementi del linguaggio per facilitare il compito del programmatore nella scrittura e nella successiva eventuale manutenzione del codice.




import turtle
import random
import math
import time
# chiedo parametri della pioggia
print ('imposta i parametri caratteristici della pioggia che vuoi simulare
# la dimensione massima della goccia
max = int(input('digita la dimensione massima della goccia '))
# calcolo in modo ca
min = random.random()
print ("la dimensione
# la distanza minima
dist = int(input('di
# la tempo massimo di
attesa = int(input("
# scelgo un valore pe
lato = 300
# disegno il quadrat
lato = lato /2
a = turtle.Turtle()
a.hideturtle()
a.penup()
a.goto(-lato,-lato)
a.pendown()
a.goto(-lato,lato)
a.goto(lato,lato)
a.goto(lato,-lato)
a.goto(-lato,-lato)
# inizia la simulazi
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 3
2 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information
.
>>>
========== RESTART: E:/articoli x sifascuola/Simulazione/pioggia.py ==
=========
imposta i parametri caratteristici della pioggia che vuoi simulare
digita la dimensione massima della goccia 5
la dimensione minima della goccia sara' 1.6330629913169314
digita la distanza minima tra le gocce 50
digita il tempo massimo di attesa tra una goccia e l'altra (in sec) 1
Python Turtle Graphics