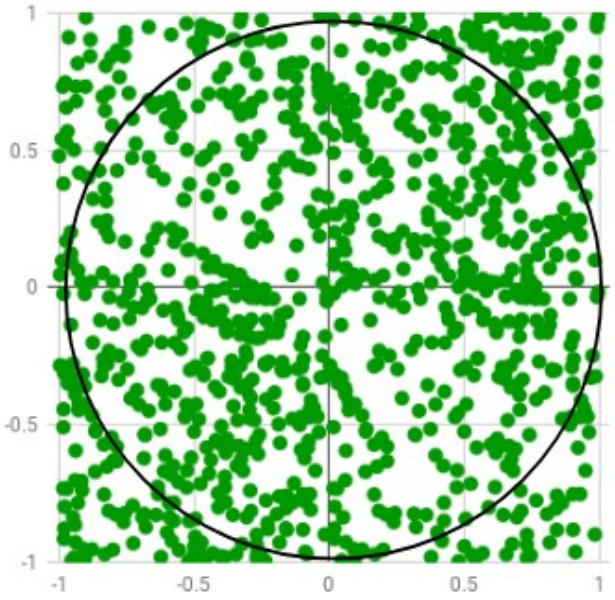
Estimating the value of Pi using Monte Carlo

Monte Carlo estimation

Monte Carlo methods are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. One of the basic examples of getting started with the Monte Carlo algorithm is the estimation of Pi.

Estimation of Pi

The idea is to simulate random (x, y) points in a 2-D plane with domain as a square of side 2r units centered on (0,0). Imagine a circle inside the same domain with same radius r and inscribed into the square. We then calculate the ratio of number points that lied inside the circle and total number of generated points. Refer to the image below:



$$2 \qquad 2$$

We know that area of the square is $4r$ unit sq while that of circle is πr .
The ratio of these two areas is as follows :

$$\frac{\text{area of the circle}}{\text{area of the square}} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4}$$

Now for a very large number of generated points,

$$\frac{\pi}{4} = \frac{\text{no. of points generated inside the circle}}{\text{total no. of points generated or no. of points generated inside the square}}$$

that is,

$$\pi = 4 * \frac{\text{no. of points generated inside the circle}}{\text{no. of points generated inside the square}}$$

The beauty of this algorithm is that we don't need any graphics or simulation to display the generated points. We simply generate random (x, y) pairs and then check if $x^2 + y^2 \leqslant 1$. If yes, we increment the number of points that appears inside the circle. In randomized and simulation algorithms like Monte Carlo, the more the number of iterations, the more accurate the result is. Thus, the title is "**Estimating** the value of Pi" and not "Calculating the value of Pi". Below is the algorithm for the method:

The Algorithm

1. Initialize circle_points, square_points and interval to 0.
2. Generate random point x.
3. Generate random point y.
4. Calculate d = x*x + y*y.
5. If d <= 1, increment circle_points.
6. Increment square_points.
7. Increment interval.
8. If increment < NO_OF_ITERATIONS, repeat from 2.
9. Calculate pi = 4*(circle_points/square_points).
10. Terminate.

The code doesn’t wait for any input via stdin as the macro INTERVAL could be changed as per the required number of iterations. Number of iterations are the square of INTERVAL. Also, I’ve paused the screen for first 10 iterations with getch() and outputs are displayed for every iteration with format given below. You can change or delete them as per requirement.

```
x y circle_points square_points - pi
```

Examples:

```
INTERVAL = 5
Output : Final Estimation of Pi = 2.56

INTERVAL = 10
Output : Final Estimation of Pi = 3.24

INTERVAL = 100
Output : Final Estimation of Pi = 3.0916
```

Python Code

```
import random

INTERVAL = 1000

circle_points = 0
square_points = 0

# Total Random numbers generated= possible x
# values* possible y values
    i in range(INTERVAL**2):

for

    # Randomly generated x and y values from a
    # uniform distribution
    # Range of x and y values is -1 to 1
    rand_x = random.uniform(-1, 1)
    rand_y = random.uniform(-1, 1)

    # Distance between (x, y) from the origin
    origin_dist = rand_x**2 + rand_y**2

    # Checking if (x, y) lies inside the circle
    if origin_dist <= 1:


        circle_points += 1

    square_points += 1

    # Estimating value of pi,
    # pi= 4*(no. of points generated inside the
    # circle)/ (no. of points generated inside the square)
    pi = 4 * circle_points / square_points

##    print(rand_x, rand_y, circle_points, square_points, "-", pi)
# print("\n")

print("Final Estimation of Pi=", pi)
```

Output:

Final Estimation of Pi = 3.16116

Online compiler:
https://www.programiz.com/python-programming/online-compiler/