Assignment 3

Ojas Phadake CH22B007

Question 1:

Introduction

What to do:

The question asks to employ the Monte-Carlo technique, to find the value of π. This involves scattering points randomly in a square and then finding out how many points lie inside a circle of predefined circle.

Method Of Solving

I imported the libraries **numpy** and **matplotlib.pyplot** at the start. Radius of the circle is taken as input from the user. After that, I used parameterization to plot the circle. I converted i to radians using np.pi*i/180, and made it go from 0 to 361 so that the 2π radians will get covered.

After that the coordinates of 20 random points were randomly obtained. I defined a function named **'detective'** which will check if a point is inside the circle or not.

The counters **inside** and **outside** are initialized to zero, and are incremented if they are inside or outside the circle respectively. The ratios of the two is found out and printed. For visual pleasing, I coloured the points inside and outside with different colours.

Plots

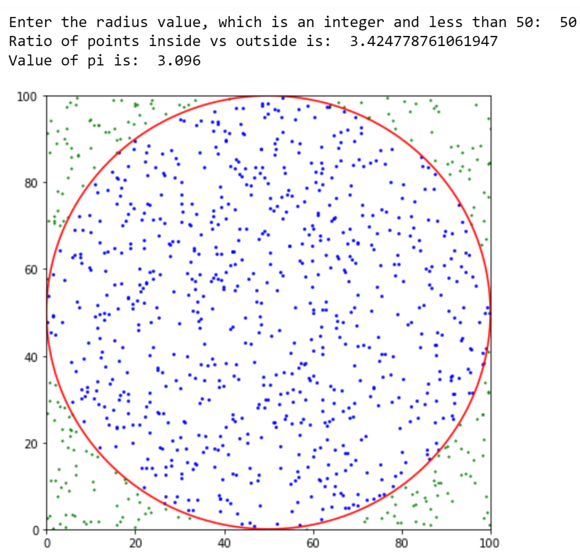
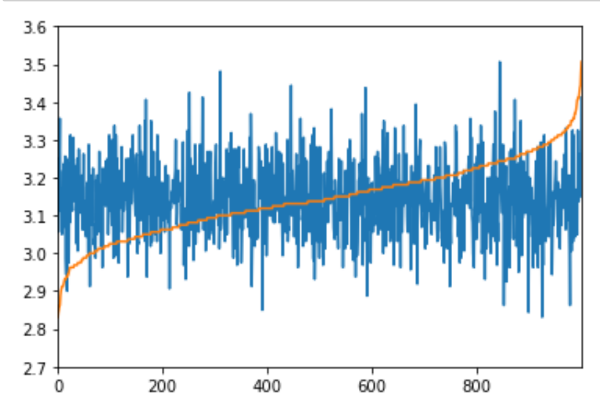


Figure 1: Obtained output

After that, I tried iterating the the same procedure 1000 times, and then taking out the mean, median, range of the graph. The results obtained were as follows:




```
The median is: 3.1375
The mean is: 3.14351875
The range is : [2.83125, 3.50625]
```

Figure 2: Something extra

The yellow plotted curve indicates the sorted values of π. It is very sharp at the edges as the values at the corners is of course going to be skewed due to relatively less number of data points.

Ways to improve accuracy

The original experiment, will be better done if more number of points are considered, which will improve the value by a few more decimal places.

$$Value\ of\ \pi = \left(\frac{100}{radius}\right)^2 * \frac{inside}{N} \tag{1}$$

We see that iterating the experiment will give us a far better value, as the average will lie closer to the true value. We also can see that if the number of points is very small, around 100, then the answer is very different. But, if number of points is about 1000 then a correct value upto 2 digits can be obtained. Also, choosing a smaller circle reduces the chance that the randomly scattered points will be in the accurate ratio, hence we must choose a circle which is in the range of 30-50 for better estimates.

Question 2

Introduction

What to do:

This question asks to plot the contour plot obtained after following a specific algorithm to colour code.

Method Of Solving

matplotlib.plt and **numpy** were imported. Two arrays named z and T were defined. As suggested by the question, the array z contains the coordinates of the points to be plotted. The array T will contain the values at each point z[x][y].

The initial values are entered in the array, like inside circle 100 and outside circle 0 are initialized.

I defined a function named **ChangingTimes** having 2 arguments, n representing the number of steps and lamb, which stands for λ. I repeated the algorithm for n times using a for loop.

The values of n and λ are taken as input from the user and then the final plot is displayed.

Plots

The obtained plots are as follows:

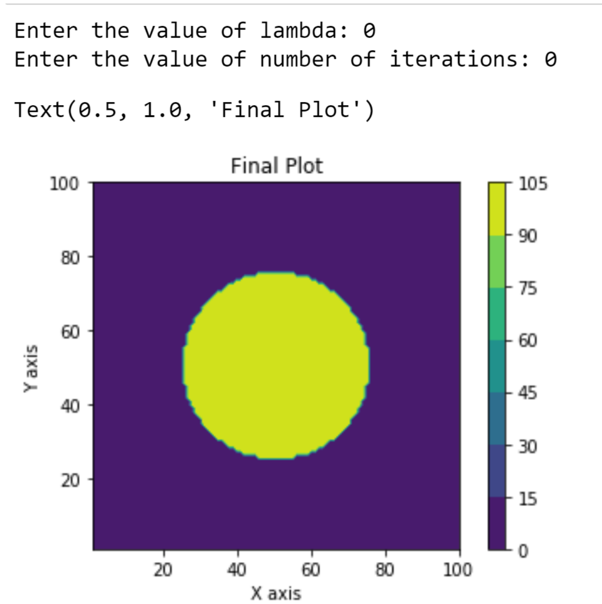


Figure 3: Initial plot

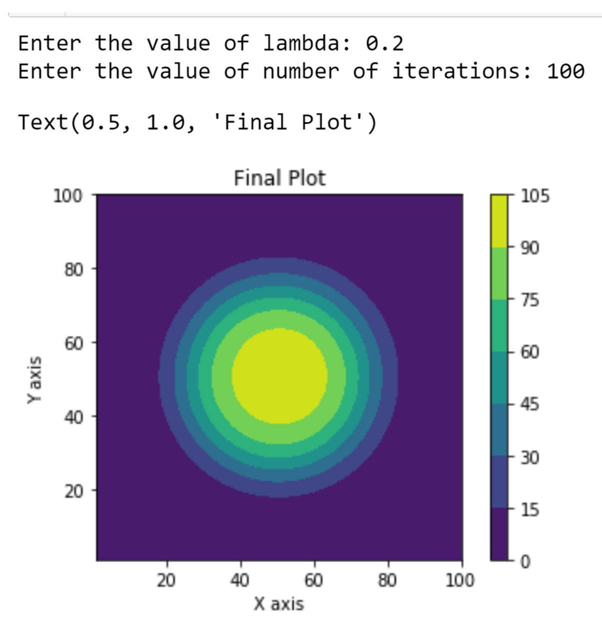


Figure 4: N = 100 and $\lambda = 0.2$

```
Enter the value of lambda: 0.5
Enter the value of number of iterations: 7

Text(0.5, 1.0, 'Final Plot')
```

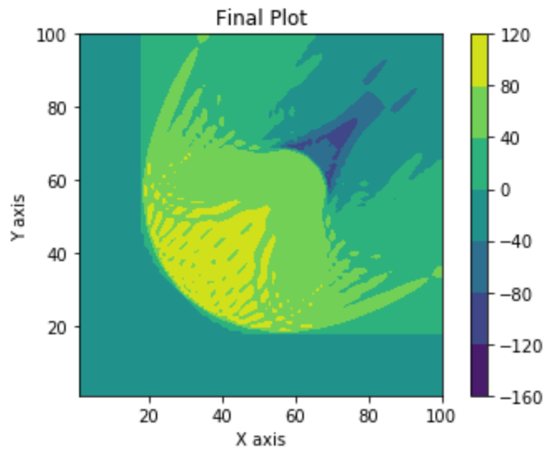


Figure 5: N = 7 and $\lambda = 0.5$

```
Enter the value of lambda: 0.1
Enter the value of number of iterations: 1000

Text(0.5, 1.0, 'Final Plot')
```

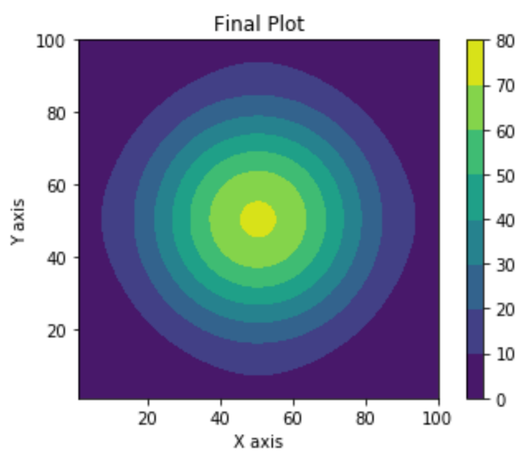


Figure 6: N = 1000 and $\lambda = 0.1$

```
Enter the value of lambda: 0.6
Enter the value of number of iterations: 30

Text(0.5, 1.0, 'Final Plot')
```

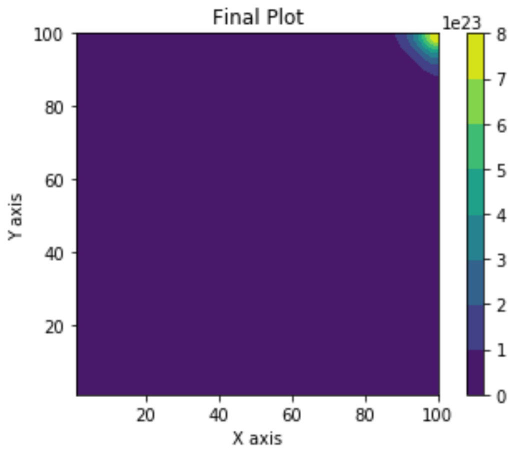


Figure 7: N = 30 and $\lambda = 0.6$

1 Conclusion

We find that as the value of lambda goes above 0.5, the circular plot starts to move to the right top corner. Uptil then the circles stay, and only after 0.5-0.6 do they start to digress.