Names: Seth Rasmussen and Chris Patania
Class: COSC 274
Project One: Ernesto Cesaro’s Theorem

In our project the true random number generator gave us a better estimate for pi using Ernesto Cesaro’s theorem. A true random number generator will always be a better way to generate numbers because the numbers are generated by factors the do not relate to the numbers themselves. We used random.org as our TRNG as it generates numbers based on a seed obtained from atmospheric radio interference. Unlike true random number generation the PRNG’s have a formula that can be reproducible given the same circumstances such as the same seed number.

Hypothesis

The experiment described below will ultimately result in three estimates of the value of Pi, which is found by using Ernesto Cesaro’s Theorem. Our hypothesis on which of those three results will be the closest, is that the TRNG’s results will be the closest estimate to the value of Pi.

Description

The project at hand is to use a programming language, we chose python, and create a program that uses the Theorem mentioned above to estimate the value of Pi. For our program we needed to test different random number generators to see which set of randomly generated numbers would come the closest to Pi and we used two PRNGs and one TRNG to do this.

Methodology

To do this we used python’s random module for two PRNGs, which was seeded with two user input numbers, and for the TRNG we used random.org to generate the list of numbers. Each random number generator generated a list of 1000 numbers between 1 and 1000. These lists were then used to find the probability of any random number pair’s GCD would be 1, from which that probability could be used in Ernesto’s Theorem to estimate Pi.

Presentation of Results

Below I have included the results of the program after it has been completed. As can be seen the first seed number entered by the user is 13 and the second is 324. The results are laid out in order starting with the first PRNG and ending with the TRNG.

```
[reno@eradLuin cosc274Project]$ python PI_thon_estimate.py
Please enter a seed number: 13
Please enter a second seed number: 324
PRNG with seed number one Pi estimate is:  3.08606699924
PRNG with seed number two Pi estimate is:  3.18896402072
TRNG using random.org Pi estimate is:  3.16491619017
```

Analysis of Results

As can be seen in the presentation of results above, the TRNG's results are the closest to the value of Pi at 3.1416. Coming in at a close second is the PRNG number 2, which is using python's random module with a seed number of 324.

- How does the range of the numbers used affect the accuracy of the estimate for Pi? Which of the PRNG/TRNG provides a better estimate?
  - The range of numbers used affects the percentage of the numbers having a GCD of 1, the lower the range, the higher percentage, which skews the results of estimating Pi. For example during the coding stage, there was a mistake in the code that lead the results to always produce a percentage of 100% of the numbers having a GCD of 1. This lead the estimate of Pi to result in 2.44, which is not very close to the actual value of Pi. We think that if the range of numbers was lower, say between 1 and 10, the percentage of GCD equaling 1 would be much higher, leading to the results being not as accurate compared to if the range was 1 and 1000 like done in this experiment.
- How does the number of samples used affect the accuracy of the estimate for Pi? Which of the PRNG/TRNG provides a better estimate?
  - The number of samples directly affects the accuracy in the sense of probability. For example if you are trying to prove that flipping a coin is truly a 50-50 chance, then the more times you flip the coin, the closer the probability gets to actually representing 50-50. This is because there are more data points to calculate and it makes sure that the outliers are not as influential on the results. The same idea is present in this experiment. The more numbers you have generated, the closer you get to the actual probability of number x and number y GCD being equal to 1.

Conclusion

As seen by the results our original hypothesis was correct, being that the TRNG was a more accurate representation in estimating Pi. The results of the two PRNGs ended up being not as close, but it all depended on the seed that was given to the generator. We found in playing with the program the the seed value of 324 was one of the closer estimates of the PRNGs. We found this due to an error in the original code that would round up and output only integers unless the decimal was very long. As such the only seed we managed to come across that output anything but the value 3, was 324. Another good test would have been to use the system time as a seed instead of using a given seed. This would have been much closer to a TRNG as it depends on something that is physically running on the system and the likelihood of it being the same is very slim.

Sources

The source code for this project can be found at https://github.com/reno777/PI_thon_estimate

The source code for the randomapi module used in this program can be found at the creators, Mitchell Cohen, Github page at https://github.com/mitchchn/randomapi