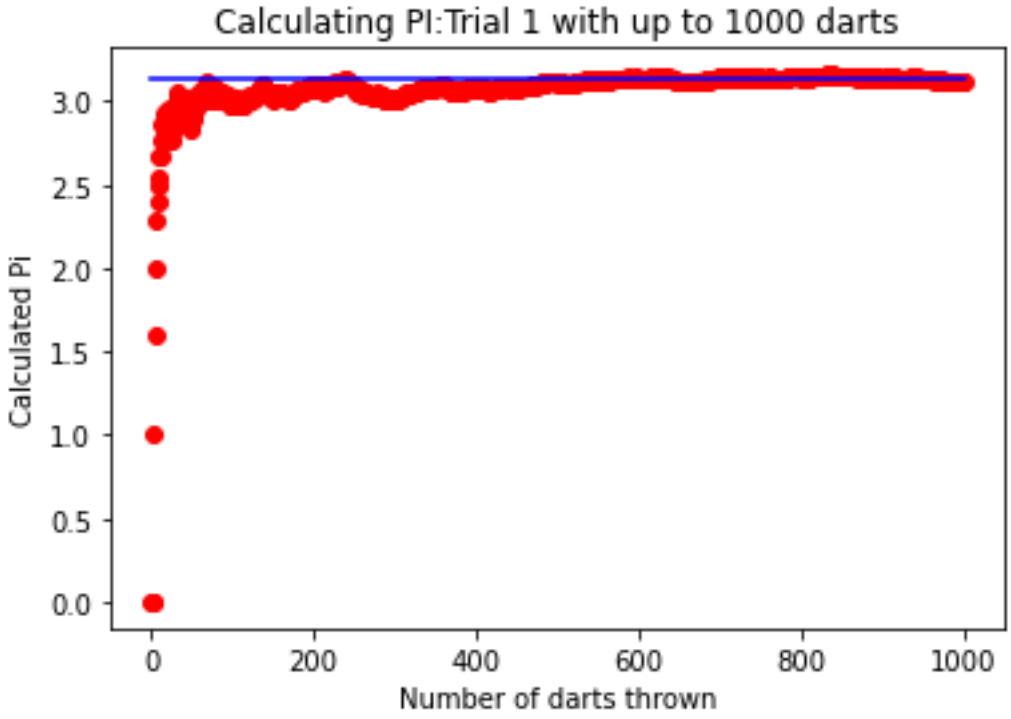
Statistics, Simulation, and Computation

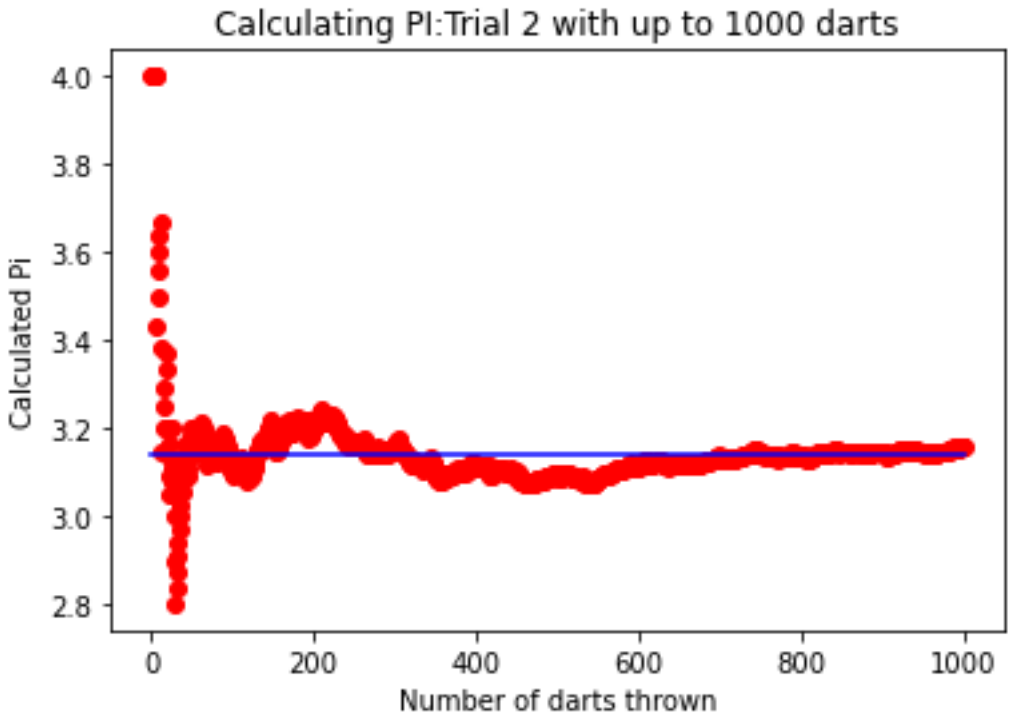
Lab 1

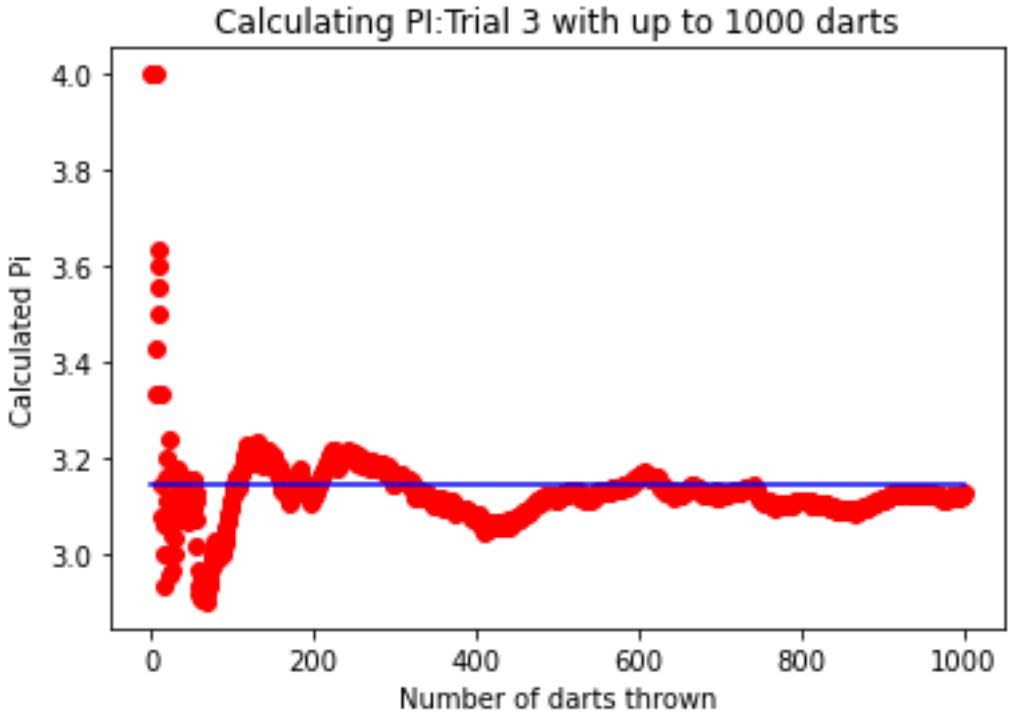
By: Kenneth Marenco

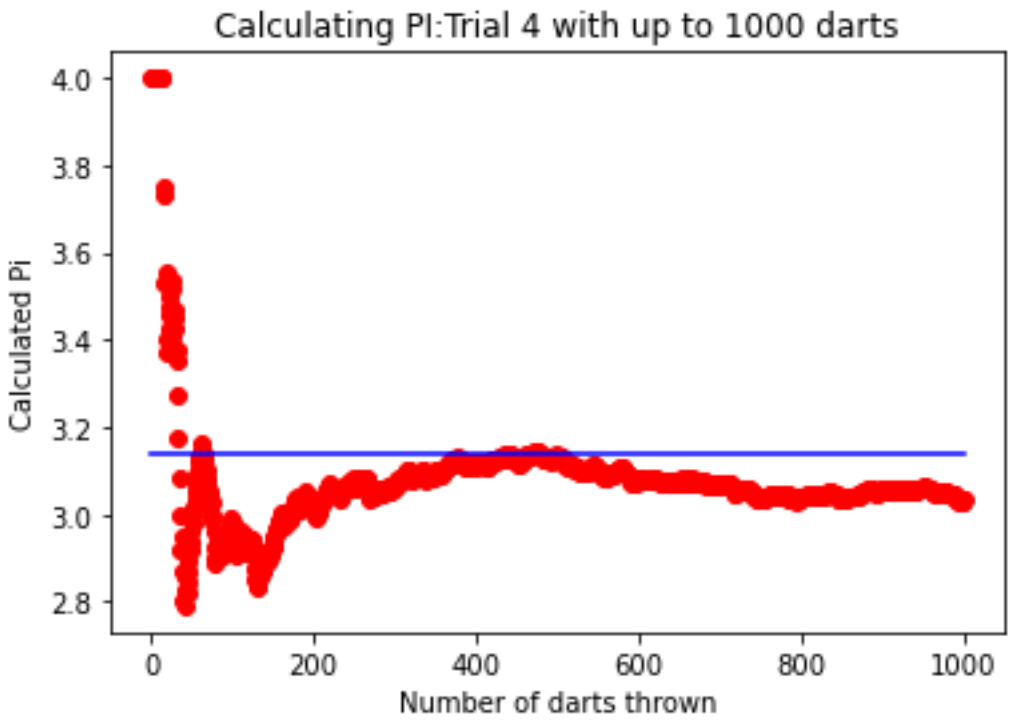
1. The Monte Carlo code that experimentally evaluates the value of pi based on the number of darts thrown uniformly in a unit circle is included separately.

2. Below is the output of running the code. Each red dot is the experimentally calculated value for pi using the number of total darts thrown. The Blue line is numpy's value of pi.



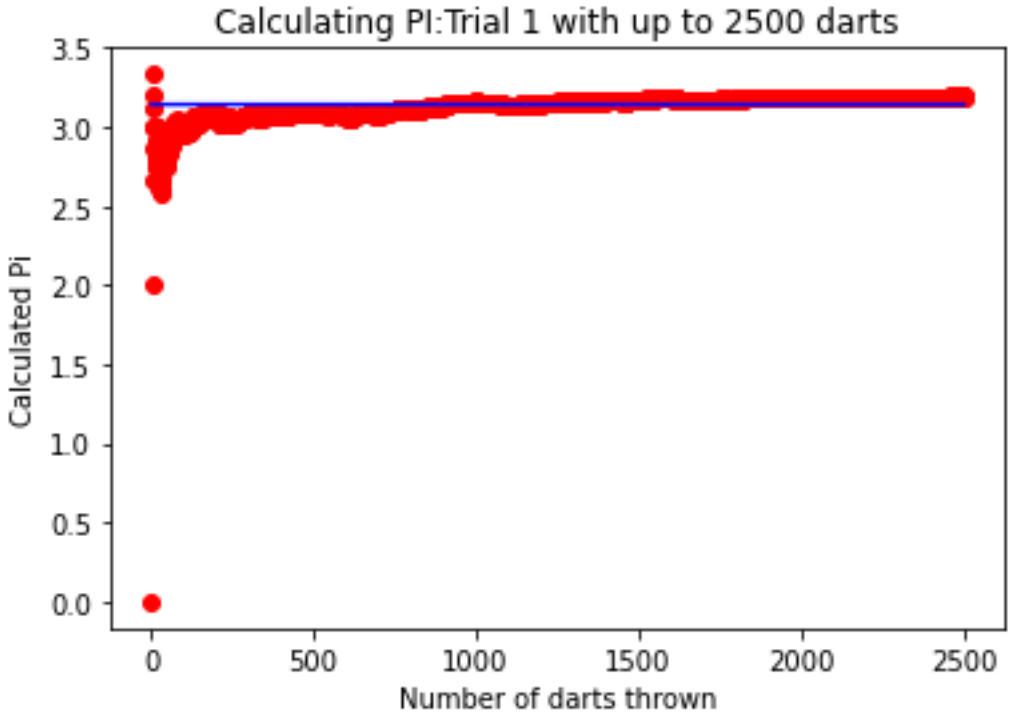

Calculating PI:Trial 2 with up to 1000 darts
Calculated Pi
4.0
3.8
3.6
3.4
3.2
3.0
2.8
0
200
400
600
800
1000
Number of darts thrown

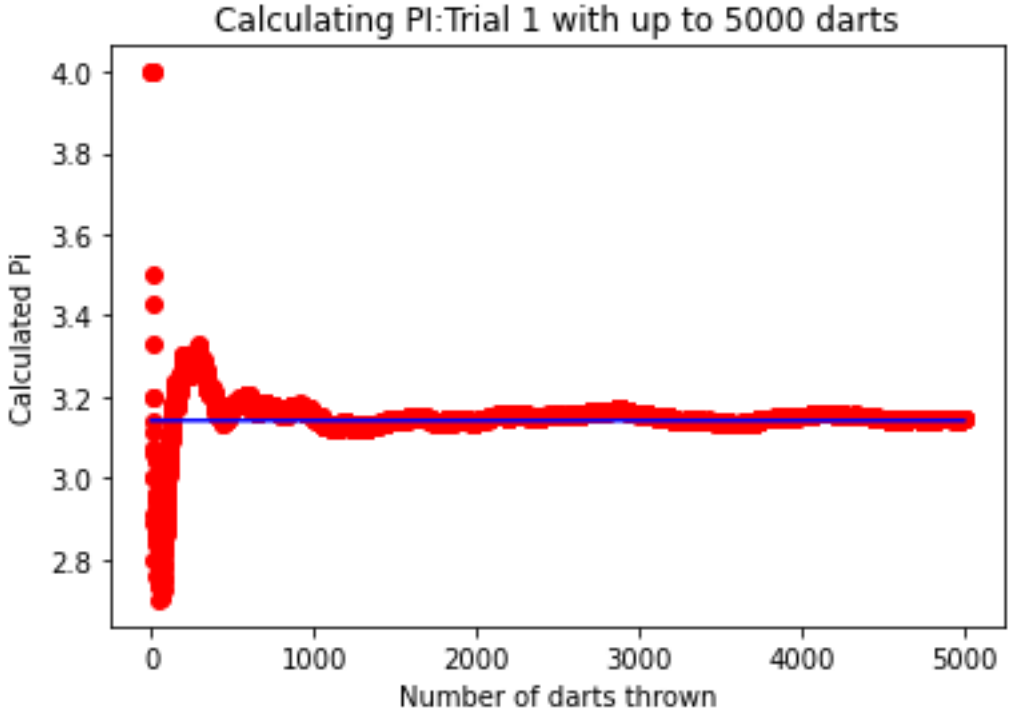

Calculating PI:Trial 3 with up to 1000 darts
Calculated Pi
4.0
3.8
3.6
3.4
3.2
3.0
0
200
400
600
800
1000
Number of darts thrown

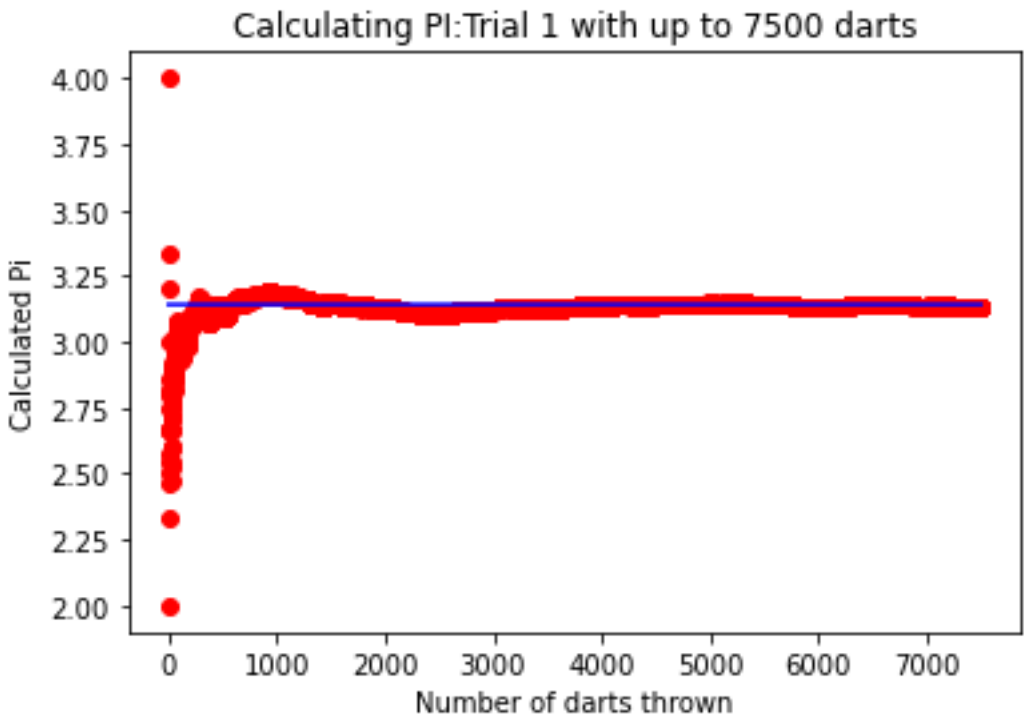


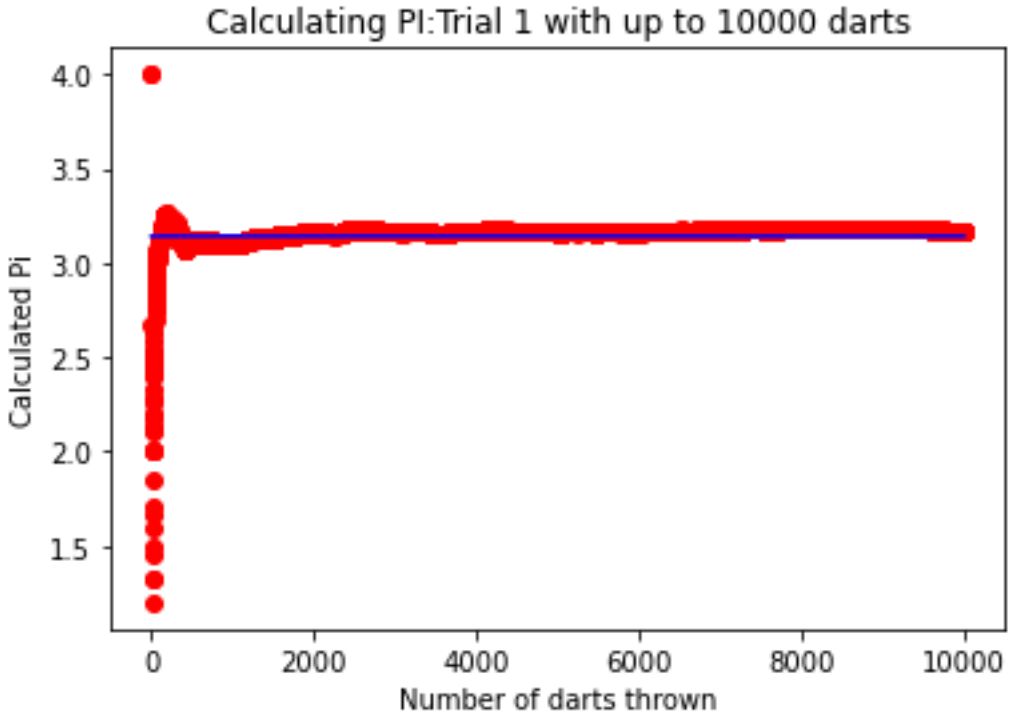
3. The number of Monte Carlo samples needed in order to arrive at a “good” estimate of pi is dependent on what level of deviation is acceptable. From the four trials seen above, some instances are very close where after 1000 throws the estimated value is 3.12. Other times, the value (such as in trial 4) is closer to 3.05. The ideal approximation would be to have an infinite number of trials to reach ever closer to pi. I believe that if the goal is to achieve pi with an approximate variation of 0.015, a sample size of 10,000 throws gives a “good” estimate.

4. Below are similar to the plots above but use a different number of thrown darts. This


Calculating PI:Trial 1 with up to 2500 darts
Calculated Pi
Number of darts thrown
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
500
1000
1500
2000
2500


Calculating PI:Trial 1 with up to 5000 darts
Calculated Pi
Number of darts thrown
4.0
3.8
3.6
3.4
3.2
3.0
2.8
0
1000
2000
3000
4000
5000





5. My conclusion is that roughly 5000 Monte Carlo samples are needed to acquire a “good” estimate of pi. This conclusion is drawn from the limited offset of the last calculated value of pi. From the earlier questions concerning the 1000 darts thrown, there was a substantial difference between the value of pi numpy provided and that calculated through simulation (a substantial difference will be noted when the blue line is almost tangent to the red circle or when the line

does not cross the last circle at all). Even at 2500 total darts thrown, there were several plots similar to the one shown above where the two values are substantially different.

6. An addition that would have improved the quality of the estimated value of pi would have been to define the parameters for what it means to have a “good” estimate of pi. The term good is too generic and does not provide a quantifiable means to determine whether the simulation has accomplished good estimates. This along with an addition of tabulated information would be much more useful in conveying those new parameters. Currently, the only basis for claiming good estimates are plots (which are useful for understanding the behavior of the simulation and how it operates) but it fails to display what the final estimates of pi are. This also fails to see the amount of fluctuation for the estimate per trial. A difference that may have helped would be to draw attention to what determines uniformly random samples. The methodology of numpy’s numpy.random.uniform is likely different from how other libraries create uniformly random data points which may require discrepencies.