Daily Coding Problem #14

Problem

This problem was asked by Google.

The area of a circle is defined as πr^2. Estimate π to 3 decimal places using a Monte Carlo method.

Hint: The basic equation of a circle is $x^2 + y^2 = r^2$.

Solution

Monte Carlo methods rely on random sampling. In this case, if we take a cartesian plane and inscribe a circle with radius `r` inside a square with lengths `2r`, then the area of the circle will be πr^2 while the area of the square will be $(2r)^2 = 4r^2$. Then, the ratio of the areas of the circle to the square is `π / 4`.

So, what we can do is the following:

- Set r to be 1 (the unit circle)
- Randomly generate points within the square with corners (-1, -1), (1, 1), (1, -1), (-1, 1)
- Keep track of the points that fall inside and outside the circle
    - You can check whether a point (x, y) is inside the circle if $x^2 + y^2 < r^2$, which is another way of representing a circle
- Divide the number of points that fall inside the circle to the total number of points -- that should give us an approximation of π / 4.

```
from random import uniform
from math import pow

def generate():
    return (uniform(-1, 1), uniform(-1, 1))

def is_in_circle(coords):
    return coords[0] * coords[0] + coords[1] * coords[1] < 1

def estimate():
    iterations = 10000000
    in_circle = 0
    for _ in range(iterations):
        if is_in_circle(generate()):
            in_circle += 1
    pi_over_four = in_circle / iterations
    return pi_over_four * 4
```

Note that this doesn't give a perfect approximation -- we need more iterations to get a closer estimate. We want the digits

of pi up to 3 decimal places. This translates to an error of < 10^(-3). The error scales with the square root of the number of guesses, which means we need 10^6 iterations to get to our desired precision. If we want more precision, we'll have to crank up the iterations.

This problem _is_ embarrassingly parallel. None of the estimations have any dependent computations, so we can parallelize this problem easily -- divide up the workload into `P` processes you have, and then add up all the points in the circle in the end. Extra credit: make this program multi-process.