Monte Carlo method: computing `pi`.

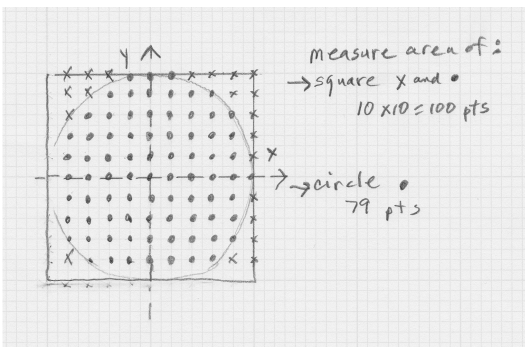
Problem setup:
For a circle, `a = pi * r`^{2} where `a` is the area of the circle, `r` is the radius of the circle.

Inscribe a unit circle (r=1) in a square (width = height = 2).
Mark grid points inside the square.

Measuring area:
The area of the square (`areaS`) is the number of grid points inside the square; it is also 2*2 = 4 as measured in length units (e.g., mm)

The area of the circle (`areaC`) is the number of grid points inside the circle;
it is (# points inside the circle / # points inside the square) * `areaS`.

Consider the picture.



For a 2 X 2 square, the number of points inside the square (count the x and the * points) is 100.
So 100 pts gives the area 4 * l^2 (where l is the unit length)

For the circle inscribed inside the 2 X 2 square, the number of points inside the circle (count the * points) is 79.
So the area of the circle is measured as 79 points,
which gives an area of (79/100) * 4 * l^2 = 3.16 * 4 * l^2.

In other words, we measure the `areaC` = the frequency of the circle points * the area of the square = (# circle points / # square points) * `areaS`.

Now, what is `pi`? For a circle `pi = areaC / r`^{2}

By measuring the area of the circle (`areaC`) as above (`areaC = (79 / 100) * areaS`), we can get a measurement of `pi: = (79/100) * areaS / r`^{2},

The square we measured above has `areaS = 4`, the circle we measured has `r = 1`,
→ then we estimate pi = `(79/100) * 4 = 3.16`

The better the measurements of area, the more accurate the measurement of pi.

Randomizing:
Recall that random number generators generate a floating point value in the range [0, 1).

So consider only the quadrant of the inscribed square x = [0, 1), y = [0, 1)

If we measure the area of the ¼ circle, we are obtaining the value of `¼ * areaC`.
So `4 * (¼ * areaC) = pi *` r^2.
Or, `pi = 4 * (¼ * areaC) /` r^2.

For the quarter circle inscribed in a 1 X 1 square, the measurement of the ¼ circle is (#points inside circle / # points inside square).
`¼ * areaC` = (#points inside quarter circle) / (# points inside quarter square) `* ¼ * areaS`

<u>Monte Carlo:</u>
Improve the measurement of pi by randomizing the location of the marker points (x and * in the picture above). Repeat this *many* times, aggregating the counts. Exit from the loop and do the multiplication.

Note: you are estimating `pi` by measuring areas.

Note: you can watch your estimate of `pi` improve as the number of iterations increases – and that, of course, means that the measurements of areas improve.

Pseudocode:

```
totalCount = 0;
circleCount = 0;
repeat many times {
      generate random x, random y;  // (x, y) within 1X1 square
      totalCount++;                 // increment area of square counter
      if (sqrt (x^2 + y^2) <= 1)
            circleCount++;          // increment area of circle counter
      }

return 4 * (circleCount / totalCount);    // this 4 is from measuring
                                          // ¼ of a circle radius 1
                                          // inside a square length = 1
```