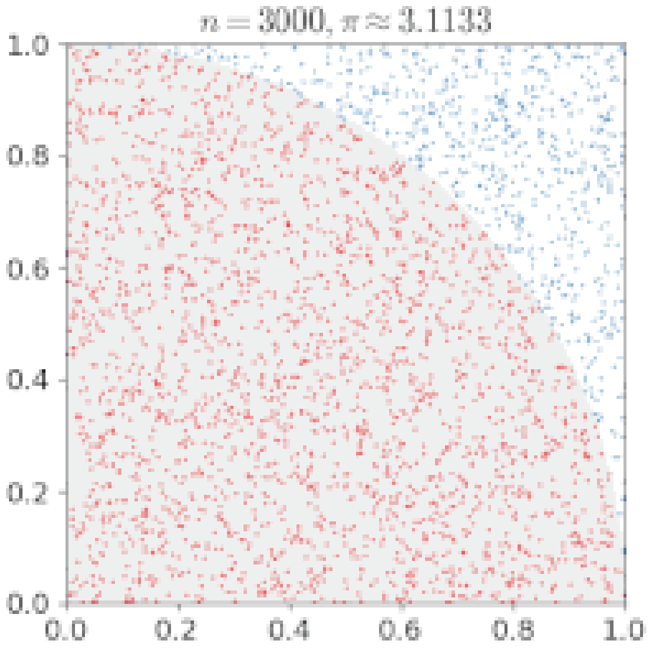
Lab Exercise

- **Monte Carlo Method**
- **Single Thread Version**

```
point_in_circle = 0
r = radius of circle
while i = 1~point_total
        x = random number between 0 and r
        y = random number between 0 and r
        if (x,y) within circle
                point_in_circle += 1
pi = 4 * point_in_circle / point_total
```

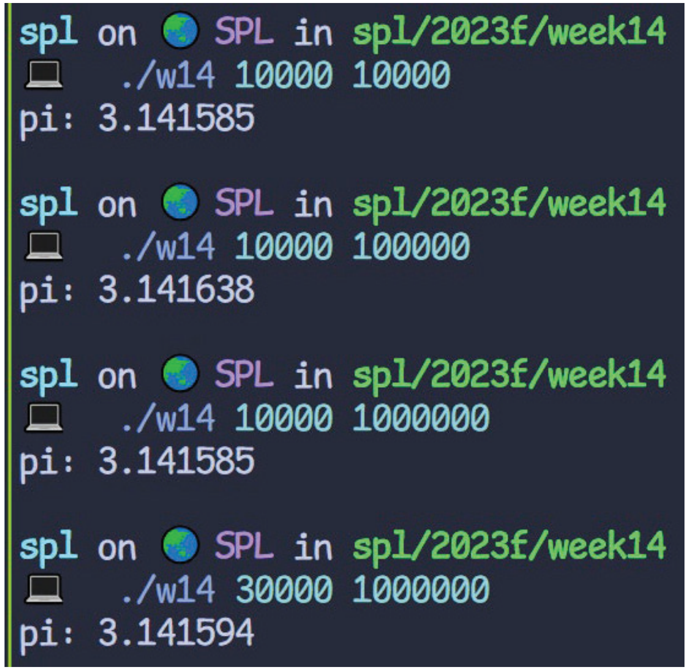


Lab Exercise

- **Making *parallel version* for Monte Carlo Method**
  - The parallel version get two command line arguments
    1) the number of threads
    2) the number of random points per thread
  - Each thread must create random points and count the number of points that are inside the circle
  - Each thread must add the number of points in the circle to the global variable “count”.
  - Once the threads have finished their work, the main thread uses the global variable to calculate the value of π.

Lab Exercise

- **Running example**



```
spl on 🌏 SPL in spl/2023f/week14
💻  ./w14 10000 10000
pi: 3.141585

spl on 🌏 SPL in spl/2023f/week14
💻  ./w14 10000 100000
pi: 3.141638

spl on 🌏 SPL in spl/2023f/week14
💻  ./w14 10000 1000000
pi: 3.141585

spl on 🌏 SPL in spl/2023f/week14
💻  ./w14 30000 1000000
pi: 3.141594
```

- **Exercise Hint**
  - Radius of the circle is always **1**
  - Generate random numbers between 0 and 1 using random_r()
    - » `struct random_data* random_state; // in data`
    - » `random_r(data->random_state, &x_int);`
    - » `x = (double)x_int / RAND_MAX;`

Exercise Submission

- **Submit your source code and Makefile**
  - The ***make*** command should generate a ***w14*** executable.
  - via iCampus

  - Bundle ***source code*** and ***Makefile*** with tar command
    - *tar.gz* format

    $ tar cvzf *[student_id].tar.gz week14*

  - We'll grade your submission with **make**
    - If compilation fails, your points for this exercise will be zero