In [1]:
```
# Estimate pi using a monte_carlo technique
# that can be speed up with numba

# if you don't have the numba package
# run 'pip install numba' from command line

import numba
import random
import numpy
```

In [2]:
```
# Simulation without using numba
def monte_carlo_pi(nsamples: int):
    acc = 0
    for i in range(nsamples):
        x = random.random()
        y = random.random()
        if (x**2 + y**2) < 1.0:
            acc += 1
    return 4.0 * acc / nsamples
```

In [3]:
```
# Simulation using numba
# Note, you only have to add a single line decorator

@numba.jit
def monte_carlo_pi_numba(nsamples: int):
    acc = 0
    for i in range(nsamples):
        x = random.random()
        y = random.random()
        if (x**2 + y**2) < 1.0:
            acc += 1
    return 4.0 * acc / nsamples
```

In [4]:
```
 # use 1,000,000 samples to determine pi
num_samples = 1000000
```

In [5]:
```
%%timeit -o
monte_carlo_pi(num_samples)
```

```
805 ms ± 32.5 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Out[5]:
```
<TimeitResult : 805 ms ± 32.5 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)>
```

In [6]:
```
# save timeit result for no numba
result_no_numba = _
```

In [7]:
```
%%timeit -o
monte_carlo_pi_numba(num_samples)
```

```
16.3 ms ± 15.1 µs per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

Out[7]:
```
<TimeitResult : 16.3 ms ± 15.1 µs per loop (mean ± std. dev. of 7 runs, 1 loop each
```

In [8]:
```
# save timeit result with numba
result_with_numba = _
```

In [11]:
```
no_numba_time = np.mean(result_no_numba.timings)
with_numba_time = np.mean(result_with_numba.timings)

# speed up factor
speed_up = no_numba_time/with_numba_time

# change/original * 100%
percent_change = (no_numba_time - with_numba_time) / no_numba_time * 100
```

In [14]:
```
# Calculate speed up.

no_numba_time = np.mean(result_no_numba.timings)
with_numba_time = np.mean(result_with_numba.timings)
print(f'Including a single line of numba decorator code results in:\n'
      f'Speed up factor of: {speed_up} or \n'
      f'Percent change in time: {percent_change}')
```

```
Including a single line of numba decorator code results in:
Speed up factor of: 49.50854201967425 or
Percent change in time: 97.98014653793963
```

In [ ]:
```
```