Q6

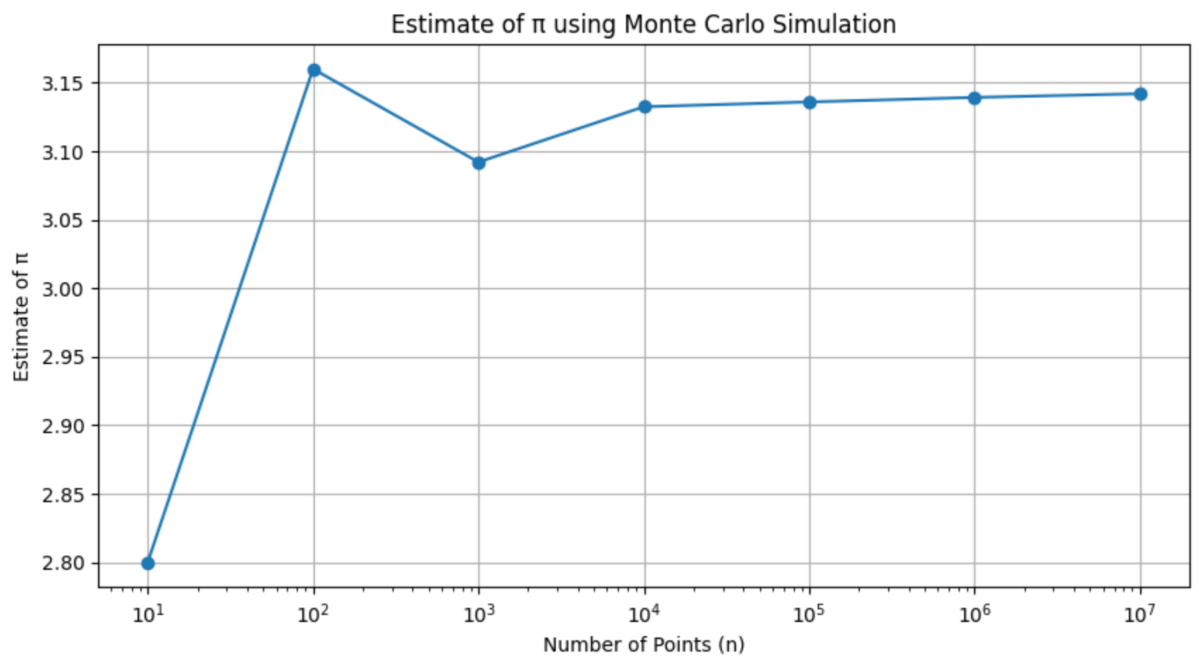
```
import random
import matplotlib.pyplot as plt

def estimate_pi(n):
    count = 0
    for _ in range(n):
        x, y = random.uniform(-1, 1), random.uniform(-1, 1)
        if x**2 + y**2 <= 1:
            count += 1
    return 4 * count / n

# Testing the function
ns = [10, 100, 1000, 10000, 100000, 1000000]  # Different values of n

pi_estimates = [estimate_pi(n) for n in ns]

# Plotting
plt.figure(figsize=(10, 5))
plt.plot(ns, pi_estimates, marker='o')
plt.xscale('log')
plt.xlabel('Number of Points (n)')
plt.ylabel('Estimate of π')
plt.title('Estimate of π using Monte Carlo Simulation')
plt.grid(True)
plt.show()
```



```
# Assuming use of a simple print statement for tabular data in Python:
print("n", "4*count/n")
for n, pi_estimate in zip(ns, pi_estimates):
    print(n, pi_estimate)
```

```
n 4*count/n
10 2.8
100 3.16
1000 3.092
10000 3.1324
100000 3.13592
1000000 3.13922
10000000 3.1418992
```

```
Observations
* As n increases, the estimates of π generally become more accurate and
consistent.
* The variability in the estimate decreases with larger n due to the law of
large numbers, which states that as a sample size grows, its mean gets closer
to the average of the whole population.
* Initially, for small n, the estimates might be highly erratic due to
insufficient sampling.

The simulation effectively demonstrates the power of random sampling to
estimate mathematical constants and the importance of large sample sizes in
achieving high precision in probabilistic simulations. It is a Monte Carlo
simulation.
```