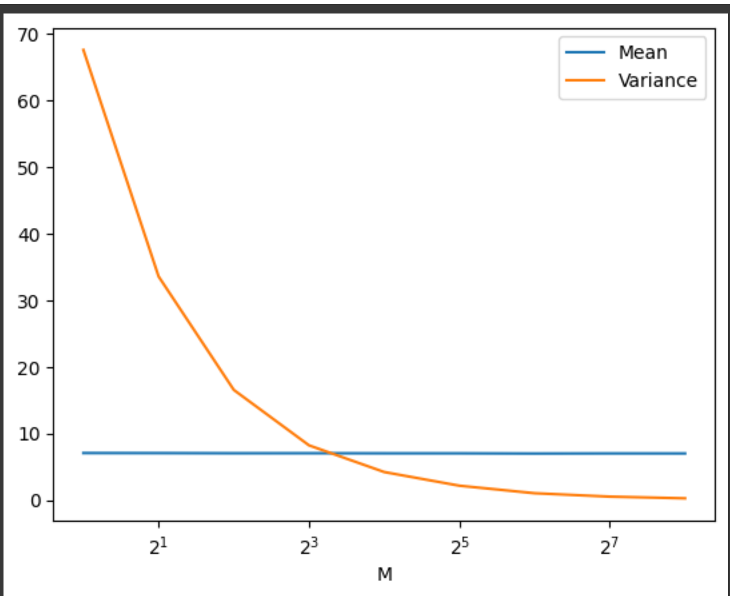
```python
import numpy as np

def monte_carlo_approximation(M):
    # Define the function f(x)
    def f(x):
        return 1 + 2*x + x**2

    # Generate M samples from the normal distribution with mean 1 and variance 3
    samples = np.random.normal(loc=1, scale=np.sqrt(3), size=M)

    # Evaluate f(x) for each sample
    evaluations = f(samples)

    # Take the average of all evaluations to approximate the expected value of f(X)
    expected_value_approximation = np.mean(evaluations)

    return expected_value_approximation
```

```python
import numpy as np
import matplotlib.pyplot as plt

def monte_carlo_simulation(N):
    # Define the function f(x)
    def f(x):
        return 1 + 2*x + x**2

    # Initialize lists to store the results
    M_values = []
    means = []
    variances = []

    # Iterate over values of M
    for M in [2**i for i in range(9)]:
        # Initialize a list to store the results of N simulations
        fM_values = []

        # Perform N simulations
        for _ in range(N):
            # Generate M samples from the normal distribution with mean 1 and variance 3
            samples = np.random.normal(loc=1, scale=np.sqrt(3), size=M)

            # Evaluate f(x) for each sample
            evaluations = f(samples)

            # Take the average of all evaluations to approximate the expected value of f(X)
            expected_value_approximation = np.mean(evaluations)

            # Store the result of this simulation
            fM_values.append(expected_value_approximation)

        # Calculate the mean and variance of fM over N rounds
        mean = np.mean(fM_values)
        variance = np.var(fM_values)

        # Store the results
        M_values.append(M)
        means.append(mean)
        variances.append(variance)

    # Visualize the results
    plt.plot(M_values, means, label="Mean")
    plt.plot(M_values, variances, label="Variance")
    plt.xscale("log", base=2)
    plt.xlabel("M")
    plt.legend()
    plt.show()

# Run the simulation with N=10000
monte_carlo_simulation(10000)
```


Mean
Variance
70
60
50
40
30
20
10
0
2^1
2^3
2^5
2^7
M