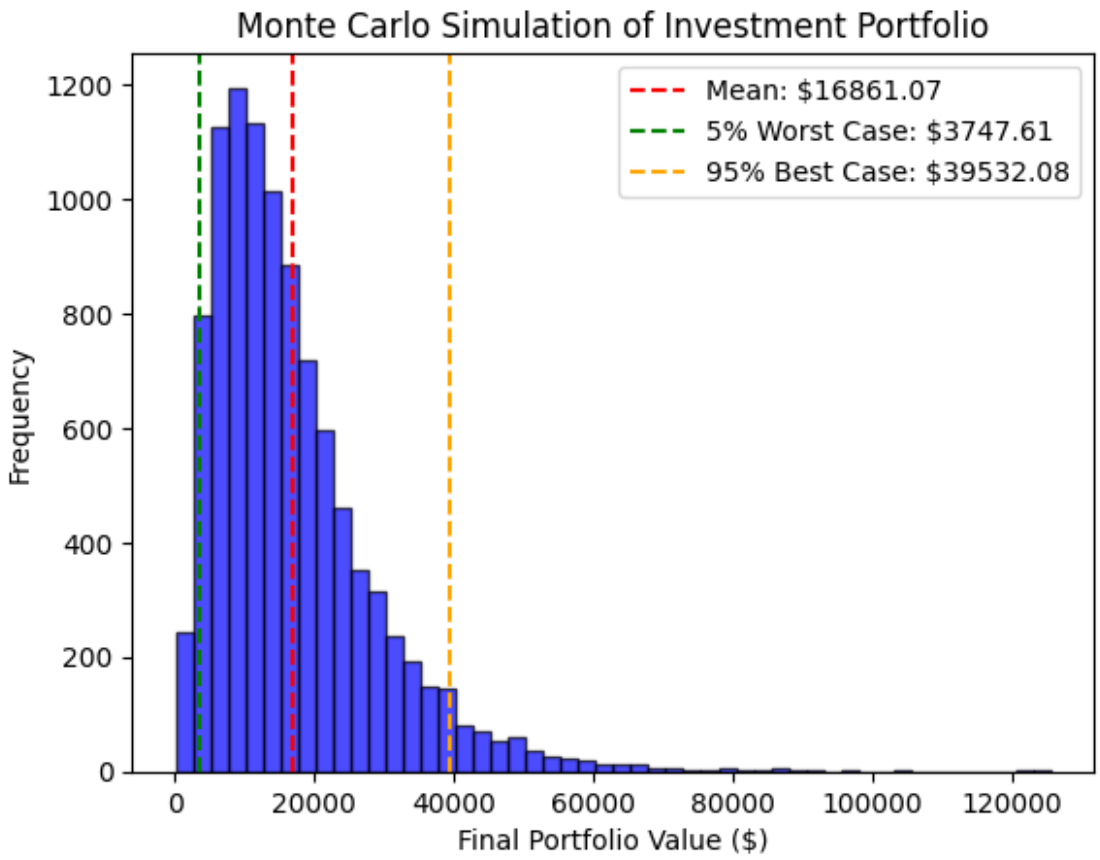
Normal Distribution: Short-term stock returns, risk modeling

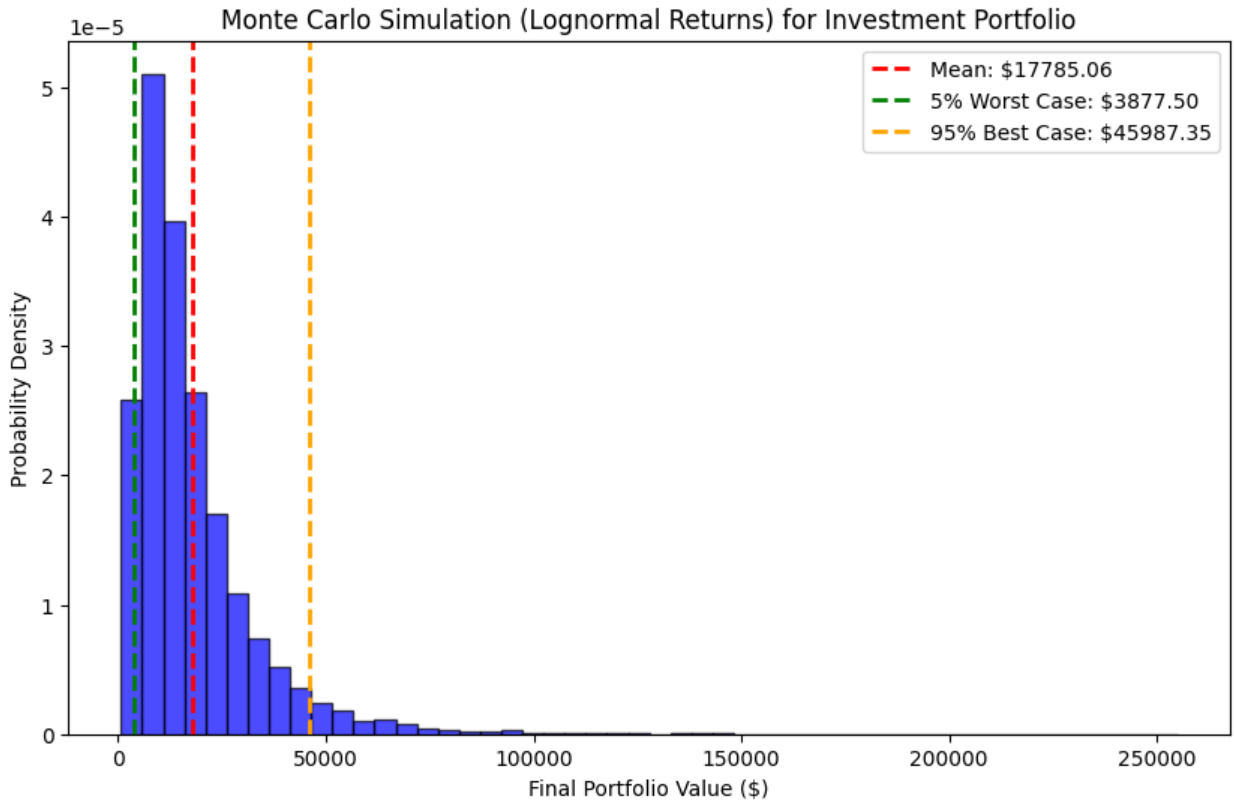
```
import numpy as np
import matplotlib.pyplot as plt

# 1. Simulation Parameters
np.random.seed(42)
num_simulations = 10000
initial_investment = 10000  # $10,000
expected_return = 0.07  # 7% mean return
volatility = 0.15  # 15% standard deviation (risk)
years = 5

# 2. Monte Carlo Simulation
simulated_returns = np.random.normal(expected_return, volatility,
num_simulations)
final_values = initial_investment * (1 + simulated_returns) ** years

# 3. Results Analysis
mean_value = np.mean(final_values)
percentile_5 = np.percentile(final_values, 5)  # Worst-case scenario
percentile_95 = np.percentile(final_values, 95)  # Best-case scenario

# 4. Plot Results
plt.hist(final_values, bins=50, alpha=0.7, color="blue",
edgecolor="black")
plt.axvline(mean_value, color="red", linestyle="dashed", label=f"Mean:
${mean_value:.2f}")
plt.axvline(percentile_5, color="green", linestyle="dashed",
label=f"5% Worst Case: ${percentile_5:.2f}")
plt.axvline(percentile_95, color="orange", linestyle="dashed",
label=f"95% Best Case: ${percentile_95:.2f}")
plt.xlabel("Final Portfolio Value ($)")
plt.ylabel("Frequency")
plt.title("Monte Carlo Simulation of Investment Portfolio")
plt.legend()
plt.show()
```



Lognormal Distribution: Portfolio values, investment returns, long-term modeling

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

# 1. Simulation Parameters
np.random.seed(42)
num_simulations = 10000  # Number of Monte Carlo trials
initial_investment = 10000  # Starting investment in USD
expected_annual_return = 0.07  # 7% mean return
volatility = 0.15  # 15% standard deviation (risk)
years = 5  # Investment duration

# 2. Convert Normal to Lognormal Distribution
mu = expected_annual_return - (0.5 * volatility**2)  # Adjusted mean
for lognormal
sigma = volatility  # Lognormal standard deviation

# 3. Simulate Lognormal Returns Over Multiple Trials
simulated_returns = np.random.lognormal(mu, sigma, num_simulations)
final_values = initial_investment * (simulated_returns ** years)  #
Apply growth over time
```

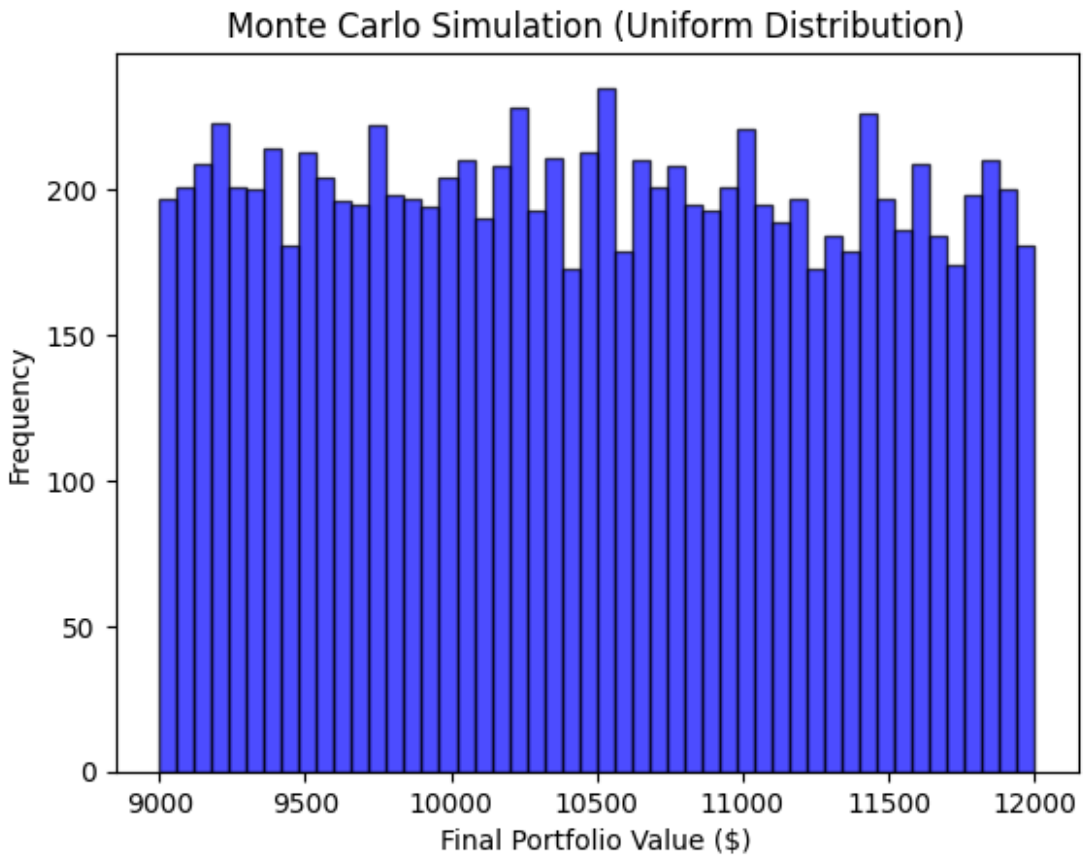
```
# 4. Analyze Results
mean_value = np.mean(final_values)  # Expected mean portfolio value
percentile_5 = np.percentile(final_values, 5)  # Worst 5% outcome
percentile_95 = np.percentile(final_values, 95)  # Best 5% outcome

# 5. Plot Results
plt.figure(figsize=(10, 6))
plt.hist(final_values, bins=50, alpha=0.7, color="blue",
edgecolor="black", density=True)
plt.axvline(mean_value, color="red", linestyle="dashed", linewidth=2,
label=f"Mean: ${mean_value:.2f}")
plt.axvline(percentile_5, color="green", linestyle="dashed",
linewidth=2, label=f"5% Worst Case: ${percentile_5:.2f}")
plt.axvline(percentile_95, color="orange", linestyle="dashed",
linewidth=2, label=f"95% Best Case: ${percentile_95:.2f}")
plt.xlabel("Final Portfolio Value ($)")
plt.ylabel("Probability Density")
plt.title("Monte Carlo Simulation (Lognormal Returns) for Investment
Portfolio")
plt.legend()
plt.show()

# 6. Print Key Results
print(f"Mean Portfolio Value after {years} years: ${mean_value:.2f}")
print(f"5th Percentile (Worst-Case Scenario): ${percentile_5:.2f}")
print(f"95th Percentile (Best-Case Scenario): ${percentile_95:.2f}")
```




```
Mean Portfolio Value after 5 years: $17785.06
5th Percentile (Worst-Case Scenario): $3877.50
95th Percentile (Best-Case Scenario): $45987.35
```

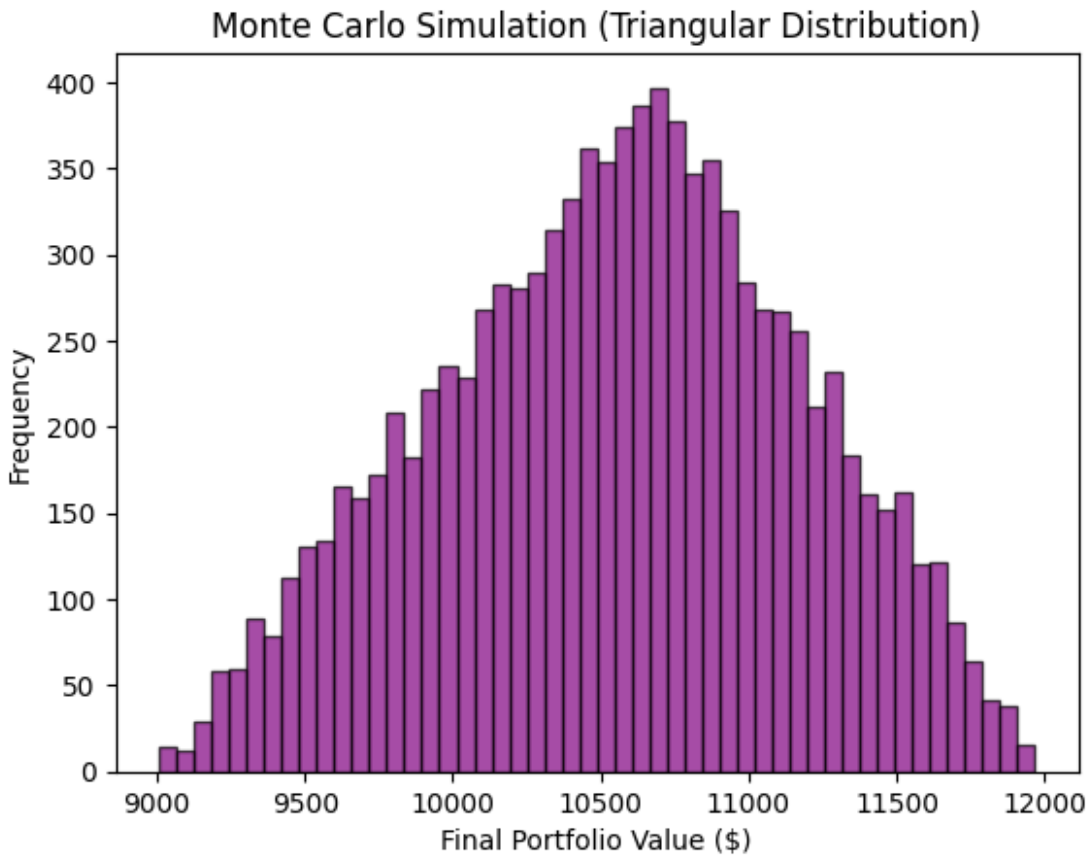
Uniform Distribution: Equal likelihood events, unknown risks

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

np.random.seed(42)
num_simulations = 10000
initial_investment = 10000

# Random returns uniformly between -10% and +20%
simulated_returns = np.random.uniform(-0.10, 0.20, num_simulations)

final_values = initial_investment * (1 + simulated_returns)

plt.hist(final_values, bins=50, color="blue", edgecolor="black",
alpha=0.7)
plt.title("Monte Carlo Simulation (Uniform Distribution)")
plt.xlabel("Final Portfolio Value ($)")
plt.ylabel("Frequency")
plt.show()
```



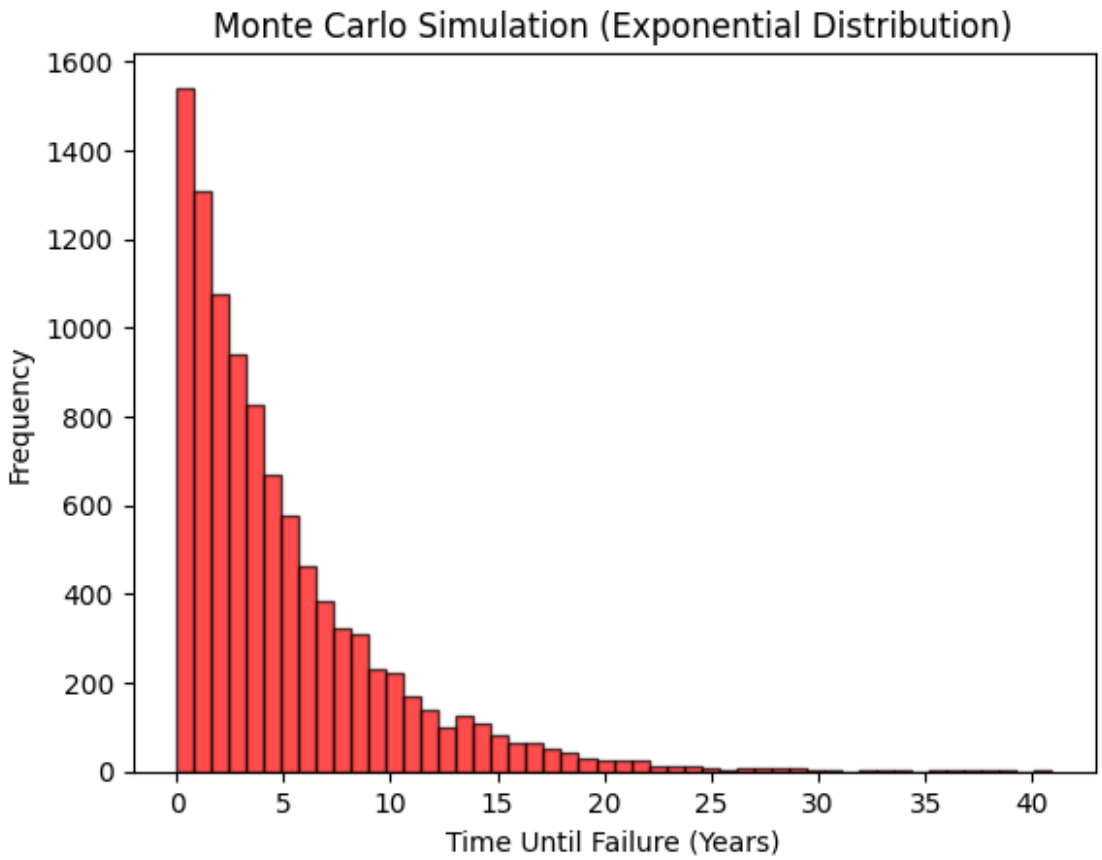
** Triangular Distribution:** Project costs, realistic risk estimates

```
from scipy.stats import triang

np.random.seed(42)
num_simulations = 10000
initial_investment = 10000

# Min = -10%, Most Likely = 7%, Max = 20%
min_return, mode_return, max_return = -0.10, 0.07, 0.20
c = (mode_return - min_return) / (max_return - min_return)

simulated_returns = triang.rvs(c, loc=min_return, scale=(max_return -
min_return), size=num_simulations)

final_values = initial_investment * (1 + simulated_returns)

plt.hist(final_values, bins=50, color="purple", edgecolor="black",
alpha=0.7)
plt.title("Monte Carlo Simulation (Triangular Distribution)")
plt.xlabel("Final Portfolio Value ($)")
plt.ylabel("Frequency")
plt.show()
```



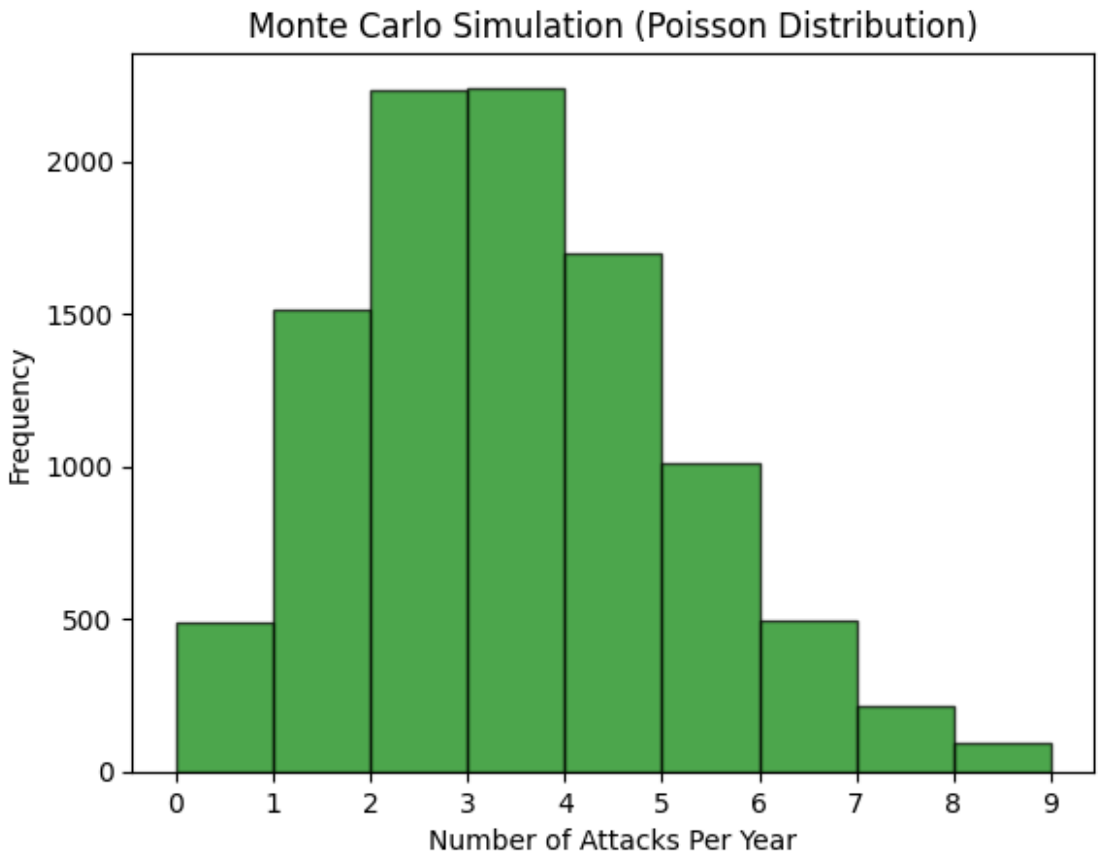
Exponential Distribution: Failures, rare events, system downtimes

```
from scipy.stats import expon

np.random.seed(42)
num_simulations = 10000
failure_rate = 1 / 5  # Failure occurs every 5 years on average

simulated_failures = expon.rvs(scale=1/failure_rate,
size=num_simulations)

plt.hist(simulated_failures, bins=50, color="red", edgecolor="black",
alpha=0.7)
plt.title("Monte Carlo Simulation (Exponential Distribution)")
plt.xlabel("Time Until Failure (Years)")
plt.ylabel("Frequency")
plt.show()
```



Poisson Distribution: Counting risk events per time period

```
from scipy.stats import poisson

np.random.seed(42)
num_simulations = 10000
lambda_value = 3  # Expect 3 cyber attacks per year

simulated_attacks = poisson.rvs(lambda_value, size=num_simulations)

plt.hist(simulated_attacks, bins=range(10), color="green",
edgecolor="black", alpha=0.7)
plt.title("Monte Carlo Simulation (Poisson Distribution)")
plt.xlabel("Number of Attacks Per Year")
plt.ylabel("Frequency")
plt.xticks(range(10))
plt.show()
```


Monte Carlo Simulation (Poisson Distribution)
Frequency
0
500
1000
1500
2000
0
1
2
3
4
5
6
7
8
9
Number of Attacks Per Year