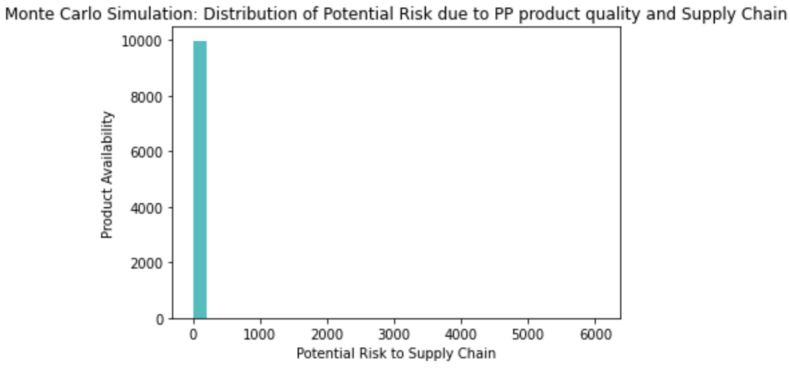
Monte Carlo Supply Chain Simulation

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

# Step 1: Forcast Data Preprocessing (Dummy values)
demand_data = [600, 710, 858, 681, 980, 890, 1001, 99, 1123, 900, 670, 809]  # monthly demand data

# Step 2: Defining quality parameters for Product Quality Issues (Dummy values)
product_quality_prob = 0.01  # Probability of product quality issues
product_quality_loss = 5000  # Average financial loss per quality issue in pounds
product_quality_std_dev = 1000  # Standard deviation for quality issues

# Step 3: Defining parameters for Supply Chain Disruptions/Inventory Cost Component (Dummy values)
service_level = 0.002  # Probability of supply chain disruptions leading to failed SL
supply_chain_reliability = 65000  # Average financial loss per disruption
supply_chain_std_dev = 13000  # Standard deviation for disruptions

# Step 4: Implementing the Monte Carlo simulation
num_iterations = 10000

# Step 5: Lists to store total losses for each iteration

total_losses = []

for _ in range(num_iterations):
    # Simulate Product Quality Issues
    product_quality_loss_sample = np.random.normal(product_quality_loss, product_quality_std_dev)
    product_quality_loss_total = product_quality_loss_sample if random.random() < service_level else 0

    # Simulate Supply Chain Disruptions
    supply_chain_loss_reliability = np.random.normal(supply_chain_reliability, supply_chain_std_dev)
    supply_chain_loss_total = supply_chain_loss_reliability if random.random() < service_level else 0
```

```
    # Calculate total loss for this Reliability iteration
    total_loss = product_quality_loss_total + supply_chain_loss_total * service_level
    total_losses.append(total_loss)

# Calculate statistics
mean_loss = np.mean(total_losses)
std_deviation = np.std(total_losses)
supplychain_reliability_10 = np.percentile(total_losses, 10)
supplychain_reliability_99 = np.percentile(total_losses, 99)

# Print results
print("Monte Carlo Simulation Results:")
print("Mean Potential Loss due to Quality: £", round(mean_loss, 2))
print("Standard Deviation: £", round(std_deviation, 2))
print("10th_supplychain_reliability_loss: £", round(supplychain_reliability_99, 2))
print("99th_supplychain_reliability_loss: £", round(supplychain_reliability_99, 2))

#Plotting a histogram of supply chain Reliability and potential financial loss

plt.hist(total_losses, bins=30, color='c')

# Print the result
plt.xlabel('Potential Risk to Supply Chain')
plt.ylabel('Product Availability')
plt.title('Monte Carlo Simulation: Distribution of Potential Risk due to PP product quality and Supply Chain')
plt.show()


quality_losses = [loss for loss in total_losses if loss < 10000]

# Plotting a histogram of potential risk to product quality
plt.xlabel('Potential Risk to Product Quality')
plt.ylabel('Probability Density')
plt.title('Distribution of Potential Risk to Product Quality')
```

```
Monte Carlo Simulation Results:
Mean Potential Loss due to Quality: £ 9.38
Standard Deviation: £ 214.16
10th_supplychain_reliability_loss: £ 0.0
99th_supplychain_reliability_loss: £ 0.0
```




```
Text(0.5, 1.0, 'Distribution of Potential Risk to Product Quality')
```