AY 2024-25 SEM VII Subject: QPM

Experiment 7

Bhuvi Ghosh
60009210191

Aim: Perform CPPI and Drawdown Constraints on specified data source.

Objective:

- Learn about CPPI and drawdown constraints.
- Implement CPPI and drawdown constraints in Python.
- Test the CPPI and drawdown constraints on a specified data source.

Theory:

CPPI stands for constant proportion portfolio insurance. It is a risk management strategy that aims to protect the investor's capital while still allowing them to participate in market growth. CPPI works by allocating a fixed proportion of the investor's wealth to the risky assets and the remaining proportion to cash. The risky assets are rebalanced periodically, and the cash is used to buy more risky assets when the market declines.

The CPPI strategy is implemented as follows:

1. The investor specifies a target floor, which is the minimum value that the portfolio is allowed to fall to.
2. The investor also specifies a multiplier, which determines how much of the portfolio is allocated to risky assets.
3. The portfolio is rebalanced periodically, and the risky assets are rebalanced to a target value that is equal to the floor * multiplier.
4. If the market declines, the cash in the portfolio is used to buy more risky assets, so that the portfolio stays above the floor.
5. If the market rises, the risky assets are sold, and the proceeds are either reinvested in the risky assets or withdrawn from the portfolio.

The CPPI algorithm can be implemented with these simple steps: (ref: https://alpaca.markets/learn/cppi-1/)

1. Calculate the cushion C= CPPI−F. Here F is the pre-defined floor for CPPI.
2. Calculate the budget allocation towards the risky assets E′=C∗M and the safe assets B′=CPPI−E′ respectively.
3. Compute the new CPPI value during the time of rebalancing from the assets returns:

   CPPI=E+B = E′(1+Er) + B′(1+Br)

E: current risky assets value

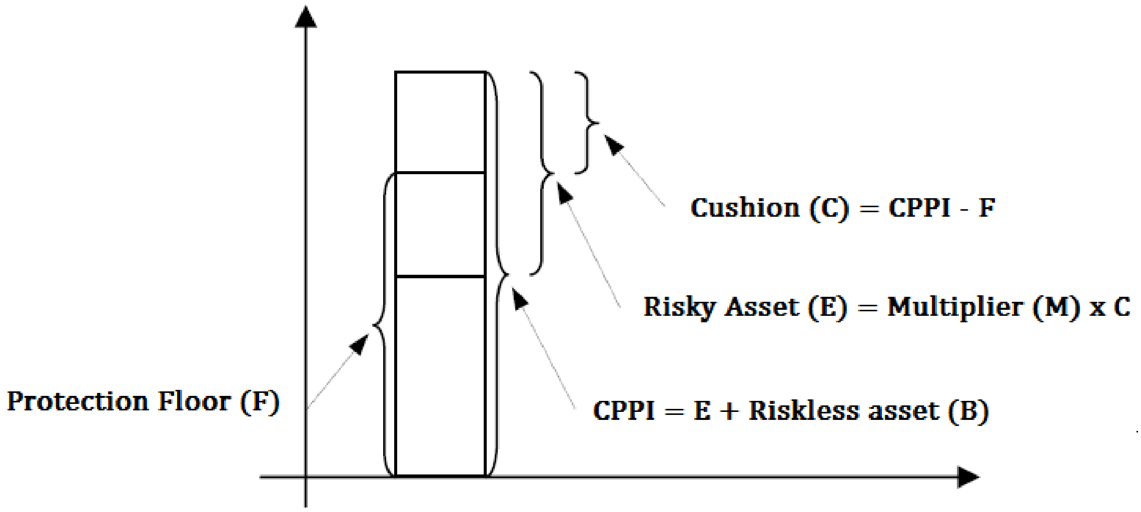
B: current safe assets value

Er: risky assets returns

Br: safe assets returns

4. Repeat step-1 & 2 with the new CPPI value for rebalancing.

*Initially CPPI = initial investment, then it will take the updated values


Cushion (C) = CPPI - F
Risky Asset (E) = Multiplier (M) x C
Protection Floor (F)
CPPI = E + Riskless asset (B)

Drawdown constraints are a way to limit the amount of loss that an investor can suffer. They work by preventing the investor's portfolio from falling below a certain level. Drawdown constraints can be implemented in a variety of ways, but they typically involve selling some of the risky assets in the portfolio when the market declines.

Drawdown constraints are implemented as follows:

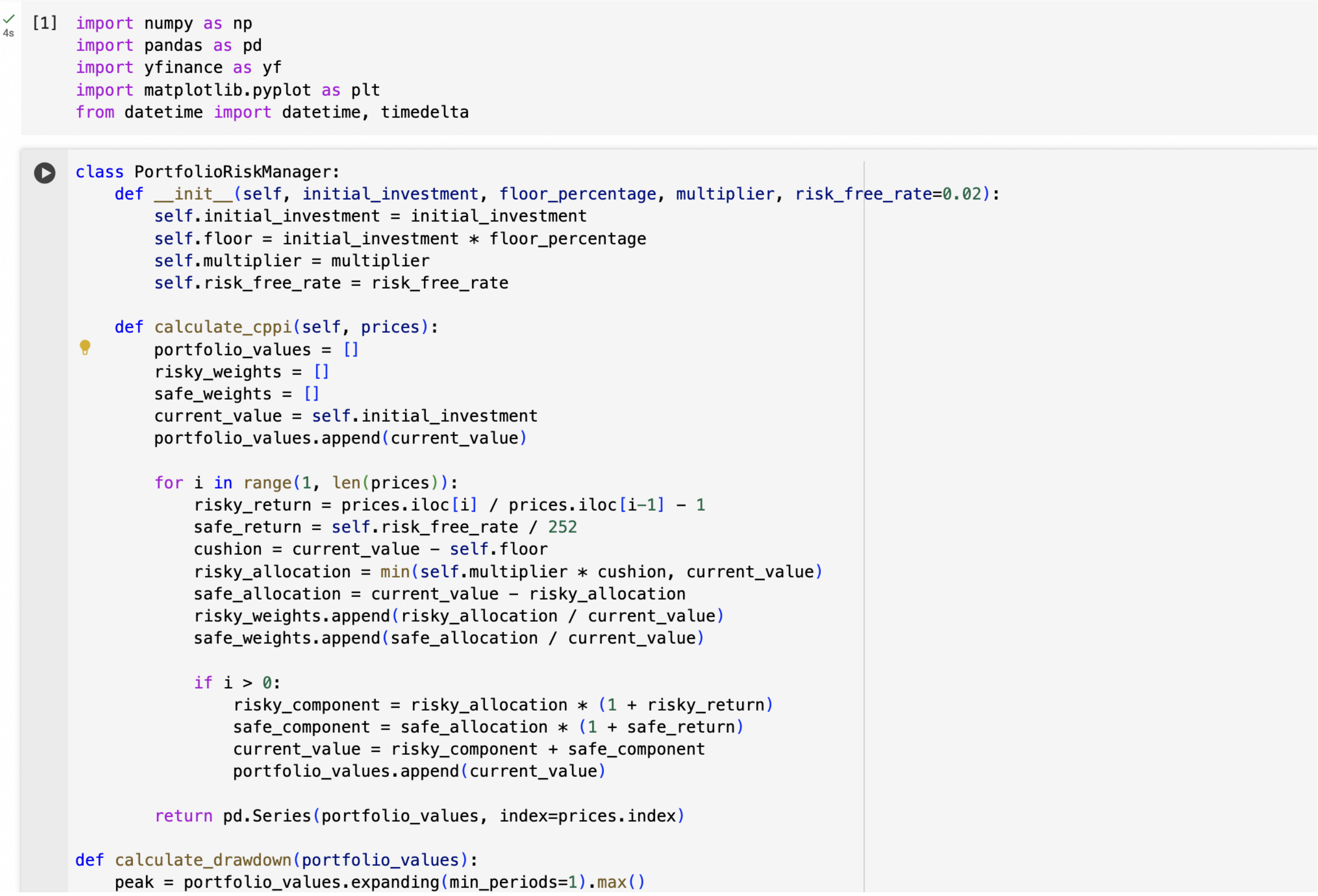
1. The investor specifies a maximum drawdown (MDD), which is the maximum percentage loss that the portfolio is allowed to suffer.
2. If the portfolio falls below the maximum drawdown, the investor sells some of the risky assets in the portfolio to bring the portfolio back above the maximum drawdown level.

The formula for the drawdown constraint is as follows:

MDD = (Trough Value - Peak Value) / Peak Value

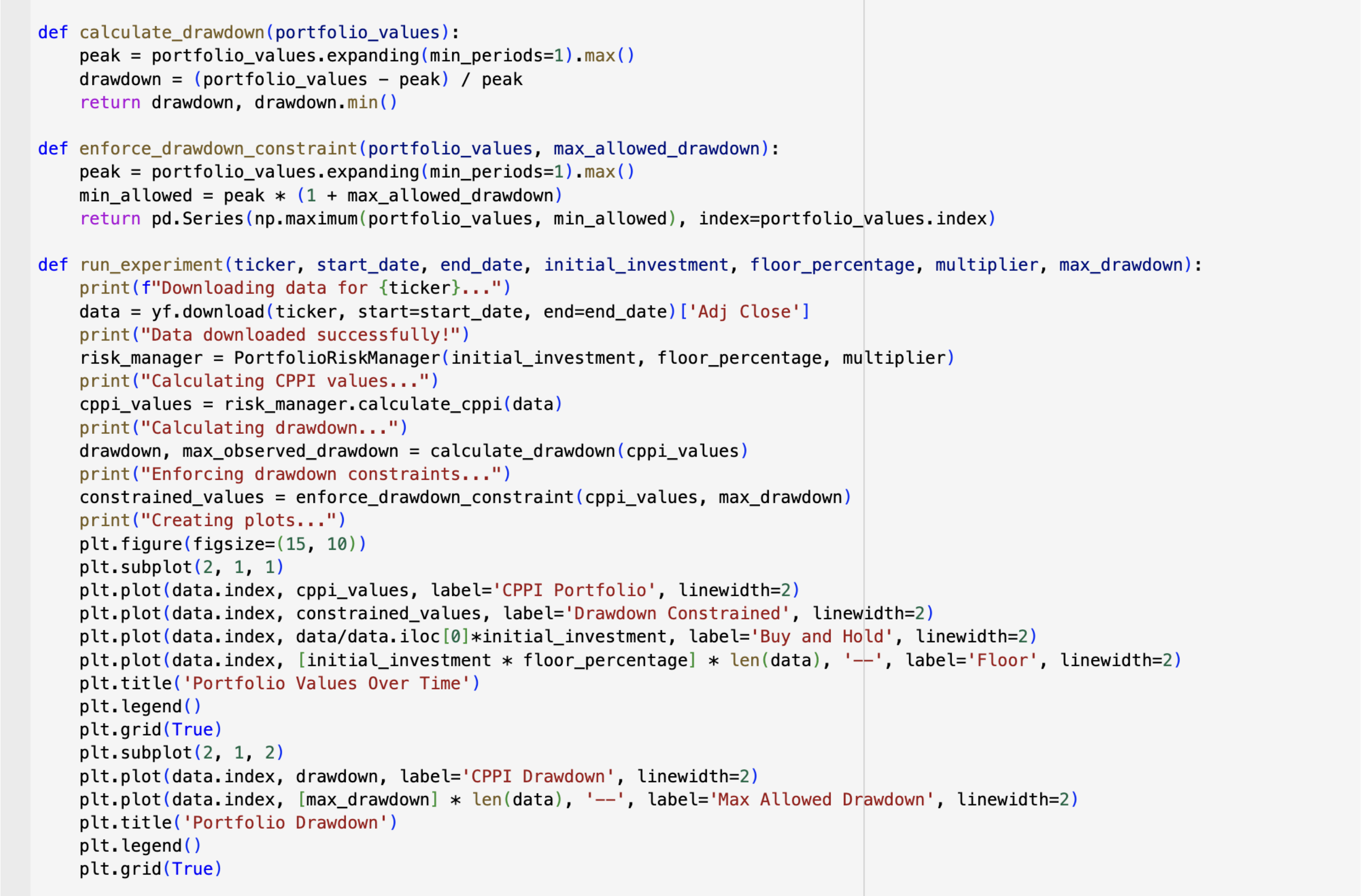
Lab Experiment to be done by students:

1. Download the specified data source.
2. Implement the CPPI and drawdown constraints in Python.
3. Test the CPPI and drawdown constraints on the specified data source.
4. Analyze the results.

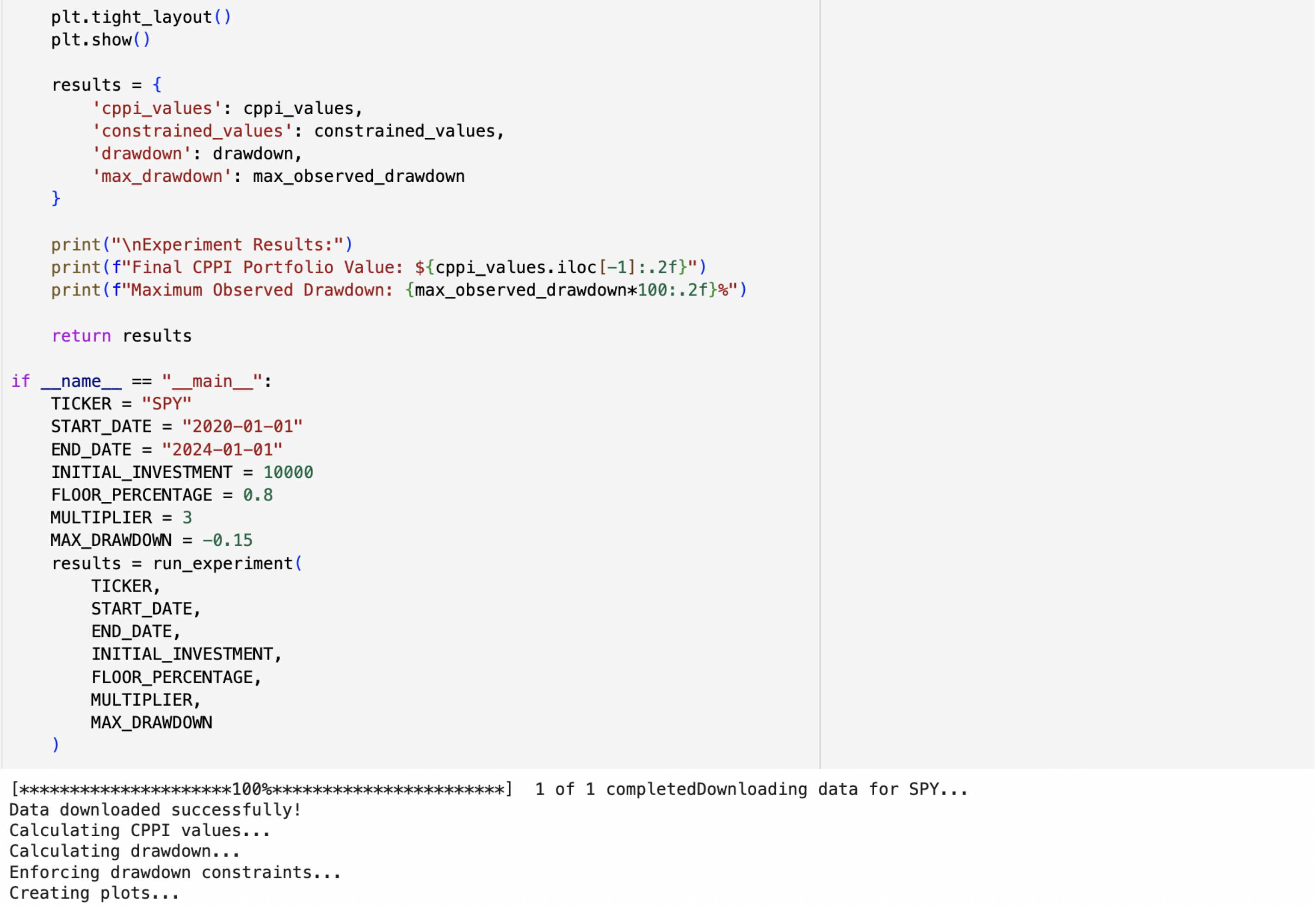


```
[1] import numpy as np
    import pandas as pd
    import yfinance as yf
    import matplotlib.pyplot as plt
    from datetime import datetime, timedelta
```

```
class PortfolioRiskManager:
    def __init__(self, initial_investment, floor_percentage, multiplier, risk_free_rate=0.02):
        self.initial_investment = initial_investment
        self.floor = initial_investment * floor_percentage
        self.multiplier = multiplier
        self.risk_free_rate = risk_free_rate

    def calculate_cppi(self, prices):
        portfolio_values = []
        risky_weights = []
        safe_weights = []
        current_value = self.initial_investment
        portfolio_values.append(current_value)

        for i in range(1, len(prices)):
            risky_return = prices.iloc[i] / prices.iloc[i-1] - 1
            safe_return = self.risk_free_rate / 252
            cushion = current_value - self.floor
            risky_allocation = min(self.multiplier * cushion, current_value)
            safe_allocation = current_value - risky_allocation
            risky_weights.append(risky_allocation / current_value)
            safe_weights.append(safe_allocation / current_value)

            if i > 0:
                risky_component = risky_allocation * (1 + risky_return)
                safe_component = safe_allocation * (1 + safe_return)
                current_value = risky_component + safe_component
                portfolio_values.append(current_value)

        return pd.Series(portfolio_values, index=prices.index)

def calculate_drawdown(portfolio_values):
    peak = portfolio_values.expanding(min_periods=1).max()
```

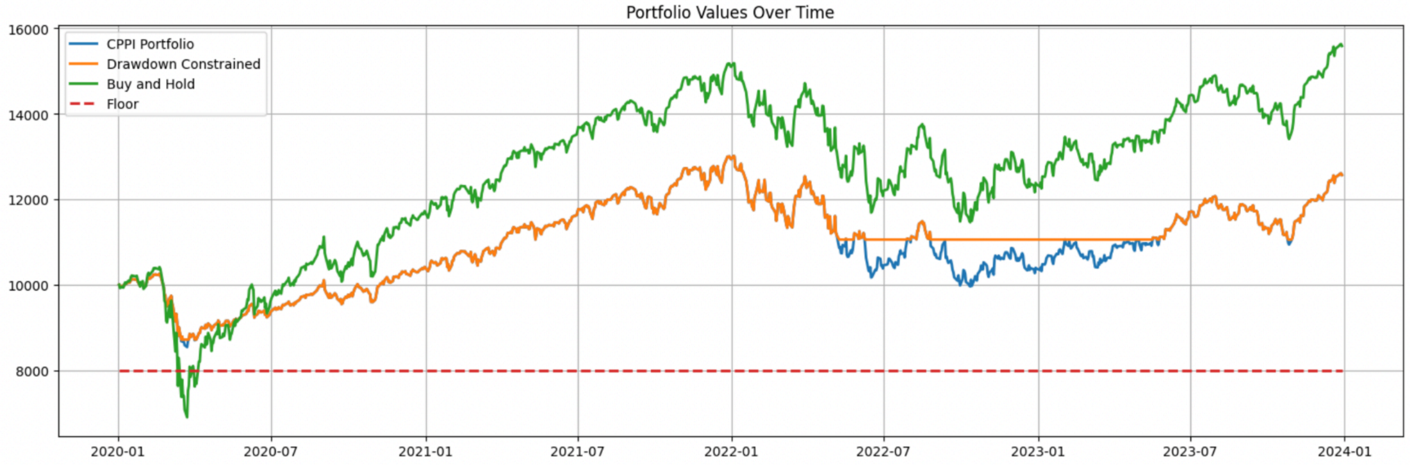


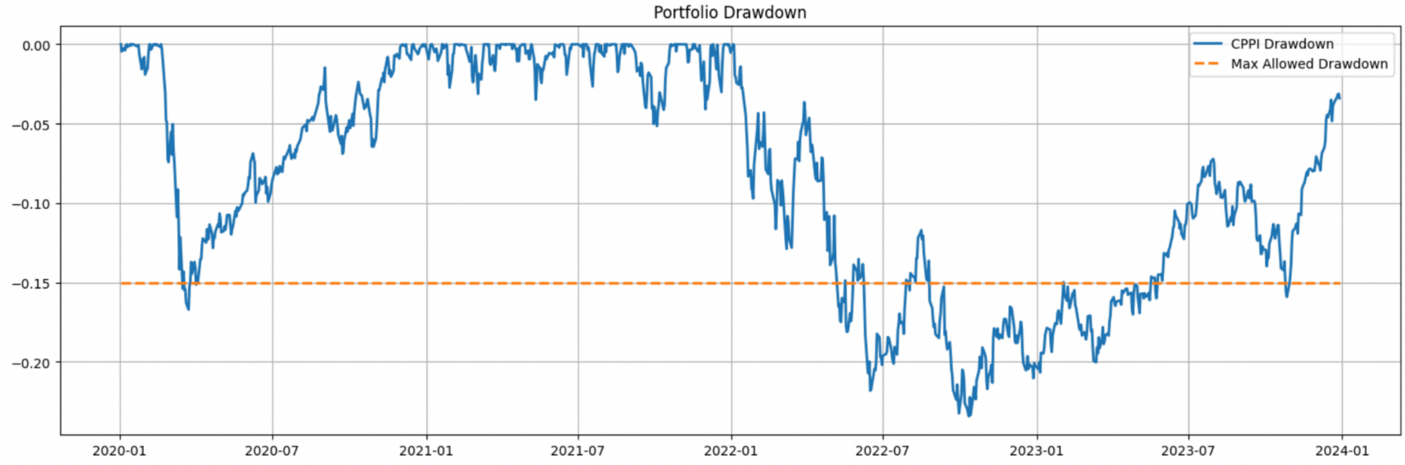
```
def calculate_drawdown(portfolio_values):
    peak = portfolio_values.expanding(min_periods=1).max()
    drawdown = (portfolio_values - peak) / peak
    return drawdown, drawdown.min()

def enforce_drawdown_constraint(portfolio_values, max_allowed_drawdown):
    peak = portfolio_values.expanding(min_periods=1).max()
    min_allowed = peak * (1 + max_allowed_drawdown)
    return pd.Series(np.maximum(portfolio_values, min_allowed), index=portfolio_values.index)

def run_experiment(ticker, start_date, end_date, initial_investment, floor_percentage, multiplier, max_drawdown):
    print(f"Downloading data for {ticker}...")
    data = yf.download(ticker, start=start_date, end=end_date)['Adj Close']
    print("Data downloaded successfully!")
    risk_manager = PortfolioRiskManager(initial_investment, floor_percentage, multiplier)
    print("Calculating CPPI values...")
    cppi_values = risk_manager.calculate_cppi(data)
    print("Calculating drawdown...")
    drawdown, max_observed_drawdown = calculate_drawdown(cppi_values)
    print("Enforcing drawdown constraints...")
    constrained_values = enforce_drawdown_constraint(cppi_values, max_drawdown)
    print("Creating plots...")
    plt.figure(figsize=(15, 10))
    plt.subplot(2, 1, 1)
    plt.plot(data.index, cppi_values, label='CPPI Portfolio', linewidth=2)
    plt.plot(data.index, constrained_values, label='Drawdown Constrained', linewidth=2)
    plt.plot(data.index, data/data.iloc[0]*initial_investment, label='Buy and Hold', linewidth=2)
    plt.plot(data.index, [initial_investment * floor_percentage] * len(data), '--', label='Floor', linewidth=2)
    plt.title('Portfolio Values Over Time')
    plt.legend()
    plt.grid(True)
    plt.subplot(2, 1, 2)
    plt.plot(data.index, drawdown, label='CPPI Drawdown', linewidth=2)
    plt.plot(data.index, [max_drawdown] * len(data), '--', label='Max Allowed Drawdown', linewidth=2)
    plt.title('Portfolio Drawdown')
    plt.legend()
    plt.grid(True)
```



```
    plt.tight_layout()
    plt.show()

    results = {
        'cppi_values': cppi_values,
        'constrained_values': constrained_values,
        'drawdown': drawdown,
        'max_drawdown': max_observed_drawdown
    }

    print("\nExperiment Results:")
    print(f"Final CPPI Portfolio Value: ${cppi_values.iloc[-1]:.2f}")
    print(f"Maximum Observed Drawdown: {max_observed_drawdown*100:.2f}%")

    return results

if __name__ == "__main__":
    TICKER = "SPY"
    START_DATE = "2020-01-01"
    END_DATE = "2024-01-01"
    INITIAL_INVESTMENT = 10000
    FLOOR_PERCENTAGE = 0.8
    MULTIPLIER = 3
    MAX_DRAWDOWN = -0.15
    results = run_experiment(
        TICKER,
        START_DATE,
        END_DATE,
        INITIAL_INVESTMENT,
        FLOOR_PERCENTAGE,
        MULTIPLIER,
        MAX_DRAWDOWN
    )
```

```
[*********************100%***********************]  1 of 1 completedDownloading data for SPY...
Data downloaded successfully!
Calculating CPPI values...
Calculating drawdown...
Enforcing drawdown constraints...
Creating plots...
```





```
Experiment Results:
Final CPPI Portfolio Value: $12567.24
Maximum Observed Drawdown: -23.42%
```