Unit 5 - Financial Planning

In [115]:
```
# Initial imports
import os
import requests
import pandas as pd
from dotenv import load_dotenv
import alpaca_trade_api as tradeapi
from MCForecastTools import MCSimulation

%matplotlib inline
```

In [116]:
```
# Load .env enviroment variables
load_dotenv()
```

Out[116]: True

Part 1 - Personal Finance Planner

Collect Crypto Prices Using the `requests` Library

In [117]:
```
# Set current amount of crypto assets
my_btc = 1.2
my_eth = 5.3
```

In [118]:
```
# Crypto API URLs
btc_url = "https://api.alternative.me/v2/ticker/Bitcoin/?convert=CAD"
eth_url = "https://api.alternative.me/v2/ticker/Ethereum/?convert=CAD"
```

In [119]:
```
# Fetch current BTC price
current_BTC_price = requests.get(btc_url)


# Fetch current ETH price
current_ETH_price = requests.get(eth_url)
```

In [120]:
```
response = current_BTC_price.content
```

In [121]:
```
data =current_BTC_price.json()
```

In [122]:
```
import json
```

In [123]:
```
print(json.dumps(data, indent = 4))
```


```
{
    "data": {
        "1": {
            "id": 1,
            "name": "Bitcoin",
            "symbol": "BTC",
            "website_slug": "bitcoin",
            "rank": 1,
            "circulating_supply": 18526025,
            "total_supply": 18526025,
            "max_supply": 21000000,
            "quotes": {
                "USD": {
                    "price": 13079.22,
                    "volume_24h": 19464458164,
                    "market_cap": 242028495269,
                    "percentage_change_1h": 0.148588548656435,
                    "percentage_change_24h": 1.11933581712965,
                    "percentage_change_7d": 15.5500373870949,
                    "percent_change_1h": 0.148588548656435,
                    "percent_change_24h": 1.11933581712965,
                    "percent_change_7d": 15.5500373870949
                },
                "CAD": {
                    "price": 17170.400016,
                    "volume_24h": 25552940677.6992,
                    "market_cap": 317735008589.143,
                    "percent_change_1h": 0.148588548656435,
                    "percent_change_24h": 1.11933581712965,
                    "percent_change_7d": 15.5500373870949
                }
            },
            "last_updated": 1603580223
        }
    },
    "metadata": {
        "timestamp": 1603580223,
        "num_cryptocurrencies": 1429,
        "error": null
    }
}
```


In [124]:
```
current_price_BTC = data['data']['1']['quotes']['USD']['price']
current_price_BTC
```

Out[124]: 13079.22

In [125]:
```
 current_price_BTC = 13065.33
```

In [126]:
```
my_btc_value =my_btc*current_price_BTC
my_btc_value
```

Out[126]: 15678.395999999999

In [127]:
```
data_2 =current_ETH_price.json()
```

In [128]:
```
print(json.dumps(data_2, indent = 4))
```


```
{
    "data": {
        "1027": {
            "id": 1027,
            "name": "Ethereum",
            "symbol": "ETH",
            "website_slug": "ethereum",
            "rank": 2,
            "circulating_supply": 113143850,
            "total_supply": 113143850,
            "max_supply": 0,
            "quotes": {
                "USD": {
                    "price": 411.34,
                    "volume_24h": 9307173473,
                    "market_cap": 46477579333,
                    "percentage_change_1h": 0.339760406066261,
                    "percentage_change_24h": 0.60237503368535,
                    "percentage_change_7d": 12.3900581820997,
                    "percent_change_1h": 0.339760406066261,
                    "percent_change_24h": 0.60237503368535,
                    "percent_change_7d": 12.3900581820997
                },
                "CAD": {
                    "price": 540.007152,
                    "volume_24h": 12218457335.3544,
                    "market_cap": 61015766148.3624,
                    "percent_change_1h": 0.339760406066261,
                    "percent_change_24h": 0.60237503368535,
                    "percent_change_7d": 12.3900581820997
                }
            },
            "last_updated": 1603580355
        }
    },
    "metadata": {
        "timestamp": 1603580355,
        "num_cryptocurrencies": 1429,
        "error": null
    }
}
```


In [129]:
```
current_price_ETH = data_2['data']['1027']['quotes']['USD']['price']
current_price_ETH
```

Out[129]: 411.34

In [130]:
```
current_price_ETH = 409.94
```

In [131]:
```
my_eth_value = my_eth * current_price_ETH
my_eth_value
```

Out[131]: 2172.682

In [132]:
```
print(f"The current value of your {my_btc} BTC is ${my_btc_value:0.2f}")
print(f"The current value of your {my_eth} ETH is ${my_eth_value:0.2f}")
```

```
The current value of your 1.2 BTC is $15678.40
The current value of your 5.3 ETH is $2172.68
```

In [133]:
```
## Collect Investments Data Using Alpaca: `SPY` (stocks) and `AGG` (bonds)
```

In [134]:
```
# Current amount of shares
my_agg = 200
my_spy = 50
```

In [135]:
```
# Set Alpaca API key and secret
# Create the Alpaca API object
alpaca_api_key = "PKEQGB76S8NG5EI0B7DH"
alpaca_secret_key = "oBLGewBGWEzdTLGJLj5TTpJjC7QVsc1Qyhxoo8PL"

api = tradeapi.REST(
    alpaca_api_key,
    alpaca_secret_key,
    api_version = "v2"
)
```

In [138]:
```
# Format current date as ISO format
today = pd.Timestamp("2020-10-23", tz="America/New_York").isoformat()

# Set the tickers
tickers = ["AGG", "SPY"]

# Set timeframe to '1D' for Alpaca API
timeframe = "1D"

# Get current closing prices for SPY and AGG
df_portfolio = api.get_barset(
    tickers,
    timeframe,
    start = today,
    end = today
).df

# Preview DataFrame
df_portfolio
```

Out[138]:

	AGG					SPY				
	open	high	low	close	volume	open	high	low	close	volume
2020-10-23 00:00:00-04:00	117.3	117.52	117.3	117.47	3482671	345.93	345.99	343.13	345.76	38718140

In [140]:
```
# Pick AGG and SPY close prices
agg_close_price = 117.47
spy_close_price = 345.76
# Print AGG and SPY close prices
print(f"Current AGG closing price: ${agg_close_price}")
print(f"Current SPY closing price: ${spy_close_price}")
```

```
Current AGG closing price: $117.47
Current SPY closing price: $345.76
```

In [141]:
```
# Compute the current value of shares
my_agg_value = my_agg*agg_close_price
my_spy_value = my_spy*spy_close_price

# Print current value of share
print(f"The current value of your {my_spy} SPY shares is ${my_spy_value:0.2f}"
)
print(f"The current value of your {my_agg} AGG shares is ${my_agg_value:0.2f}"
)
```

```
The current value of your 50 SPY shares is $17288.00
The current value of your 200 AGG shares is $23494.00
```

Savings Health Analysis

In [203]:
```
# Set monthly household income
monthly_income = 12000

# Create savings DataFrame
crypto_value = my_btc_value + my_eth_value
shares_value = my_agg_value + my_spy_value

print(crypto_value)
print(shares_value)
```

```
17851.077999999998
40782.0
```

In [204]:
```
data_01 = { "amount": [17851.077999999998, 40782.0] }
```

In [205]:
```
value_data = ["crypto", "shares"]
```

In [206]:
```
df_value  = pd.DataFrame(data_01, index =value_data)
```

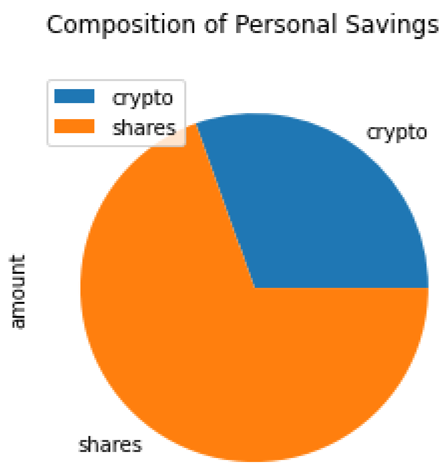
In [207]:
```
df_value
```

Out[207]:

	amount
crypto	17851.078
shares	40782.000

In [208]:
```
# Plot savings pie chart
df_value.plot.pie(subplots=True,title= "Composition of Personal Savings")
```

Out[208]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x000001FA27178108>],
      dtype=object)
```



In [211]:
```
# Set ideal emergency fund
emergency_fund = monthly_income * 3

# Calculate total amount of savings
total_amount_of_saving = crypto_value + shares_value

# Validate saving health
if emergency_fund < total_amount_of_saving:
    print("congratulation! You have enough money in your fund.")
```

```
congratulation! You have enough money in your fund.
```

Part 2 - Retirement Planning

Monte Carlo Simulation

In [212]:
```
# Set start and end dates of five years back from today.
# Sample results may vary from the solution based on the time frame chosen
start_date = pd.Timestamp('2015-08-07', tz='America/New_York').isoformat()
end_date = pd.Timestamp('2020-08-07', tz='America/New_York').isoformat()
```

In [213]:
```
# Get 5 years' worth of historical data for SPY and AGG
df_portfolio = api.get_barset(
    tickers,
    timeframe,
    start = start_date,
    end = end_date
).df

# Display sample data
df_portfolio.head()
```

Out[213]:

	AGG					SPY				
	open	high	low	close	volume	open	high	low	close	volume
2015-08-07 00:00:00-04:00	109.14	109.2750	109.035	109.21	2041167.0	208.16	208.34	206.87	207.93	8766978
2015-08-10 00:00:00-04:00	109.15	109.1700	108.920	109.06	1149778.0	209.28	210.67	209.28	210.58	6675589
2015-08-11 00:00:00-04:00	109.42	109.5765	109.284	109.42	1420907.0	208.98	209.47	207.76	208.63	8842455
2015-08-12 00:00:00-04:00	109.55	109.7100	109.350	109.36	1468979.0	207.11	209.14	205.36	208.89	13617145
2015-08-13 00:00:00-04:00	109.36	109.3651	109.110	109.15	1465173.0	208.73	209.55	208.01	208.63	7719779

In [214]:

```
# Configuring a Monte Carlo simulation to forecast 30 years cumulative returns
# Configure a Monte Carlo simulation to forecast five years cumulative returns
MC_even_dist = MCSimulation(
    portfolio_data = df_portfolio,
    weights = [.40, .60],
    num_simulation = 1000,
    num_trading_days = 252*30
)

# Print the simulation input data
MC_even_dist.portfolio_data.head()
```

Out[214]:

	AGG						SPY			
	open	high	low	close	volume	daily_return	open	high	low	clos
2015-08-07 00:00:00-04:00	109.14	109.2750	109.035	109.21	2041167.0	NaN	208.16	208.34	206.87	207
2015-08-10 00:00:00-04:00	109.15	109.1700	108.920	109.06	1149778.0	-0.001374	209.28	210.67	209.28	210
2015-08-11 00:00:00-04:00	109.42	109.5765	109.284	109.42	1420907.0	0.003301	208.98	209.47	207.76	208
2015-08-12 00:00:00-04:00	109.55	109.7100	109.350	109.36	1468979.0	-0.000548	207.11	209.14	205.36	208
2015-08-13 00:00:00-04:00	109.36	109.3651	109.110	109.15	1465173.0	-0.001920	208.73	209.55	208.01	208

In [215]:
```
# Running a Monte Carlo simulation to forecast 30 years cumulative returns
MC_even_dist.calc_cumulative_return()
```

```
Running Monte Carlo simulation number 0.
Running Monte Carlo simulation number 10.
Running Monte Carlo simulation number 20.
Running Monte Carlo simulation number 30.
Running Monte Carlo simulation number 40.
Running Monte Carlo simulation number 50.
Running Monte Carlo simulation number 60.
Running Monte Carlo simulation number 70.
Running Monte Carlo simulation number 80.
Running Monte Carlo simulation number 90.
Running Monte Carlo simulation number 100.
Running Monte Carlo simulation number 110.
Running Monte Carlo simulation number 120.
Running Monte Carlo simulation number 130.
Running Monte Carlo simulation number 140.
Running Monte Carlo simulation number 150.
Running Monte Carlo simulation number 160.
Running Monte Carlo simulation number 170.
Running Monte Carlo simulation number 180.
Running Monte Carlo simulation number 190.
Running Monte Carlo simulation number 200.
Running Monte Carlo simulation number 210.
Running Monte Carlo simulation number 220.
Running Monte Carlo simulation number 230.
Running Monte Carlo simulation number 240.
Running Monte Carlo simulation number 250.
Running Monte Carlo simulation number 260.
Running Monte Carlo simulation number 270.
Running Monte Carlo simulation number 280.
Running Monte Carlo simulation number 290.
Running Monte Carlo simulation number 300.
Running Monte Carlo simulation number 310.
Running Monte Carlo simulation number 320.
Running Monte Carlo simulation number 330.
Running Monte Carlo simulation number 340.
Running Monte Carlo simulation number 350.
Running Monte Carlo simulation number 360.
Running Monte Carlo simulation number 370.
Running Monte Carlo simulation number 380.
Running Monte Carlo simulation number 390.
Running Monte Carlo simulation number 400.
Running Monte Carlo simulation number 410.
Running Monte Carlo simulation number 420.
Running Monte Carlo simulation number 430.
Running Monte Carlo simulation number 440.
Running Monte Carlo simulation number 450.
Running Monte Carlo simulation number 460.
Running Monte Carlo simulation number 470.
Running Monte Carlo simulation number 480.
Running Monte Carlo simulation number 490.
Running Monte Carlo simulation number 500.
Running Monte Carlo simulation number 510.
Running Monte Carlo simulation number 520.
Running Monte Carlo simulation number 530.
Running Monte Carlo simulation number 540.
Running Monte Carlo simulation number 550.
Running Monte Carlo simulation number 560.
```

```
Running Monte Carlo simulation number 570.
Running Monte Carlo simulation number 580.
Running Monte Carlo simulation number 590.
Running Monte Carlo simulation number 600.
Running Monte Carlo simulation number 610.
Running Monte Carlo simulation number 620.
Running Monte Carlo simulation number 630.
Running Monte Carlo simulation number 640.
Running Monte Carlo simulation number 650.
Running Monte Carlo simulation number 660.
Running Monte Carlo simulation number 670.
Running Monte Carlo simulation number 680.
Running Monte Carlo simulation number 690.
Running Monte Carlo simulation number 700.
Running Monte Carlo simulation number 710.
Running Monte Carlo simulation number 720.
Running Monte Carlo simulation number 730.
Running Monte Carlo simulation number 740.
Running Monte Carlo simulation number 750.
Running Monte Carlo simulation number 760.
Running Monte Carlo simulation number 770.
Running Monte Carlo simulation number 780.
Running Monte Carlo simulation number 790.
Running Monte Carlo simulation number 800.
Running Monte Carlo simulation number 810.
Running Monte Carlo simulation number 820.
Running Monte Carlo simulation number 830.
Running Monte Carlo simulation number 840.
Running Monte Carlo simulation number 850.
Running Monte Carlo simulation number 860.
Running Monte Carlo simulation number 870.
Running Monte Carlo simulation number 880.
Running Monte Carlo simulation number 890.
Running Monte Carlo simulation number 900.
Running Monte Carlo simulation number 910.
Running Monte Carlo simulation number 920.
Running Monte Carlo simulation number 930.
Running Monte Carlo simulation number 940.
Running Monte Carlo simulation number 950.
Running Monte Carlo simulation number 960.
Running Monte Carlo simulation number 970.
Running Monte Carlo simulation number 980.
Running Monte Carlo simulation number 990.
```

Out[215]:

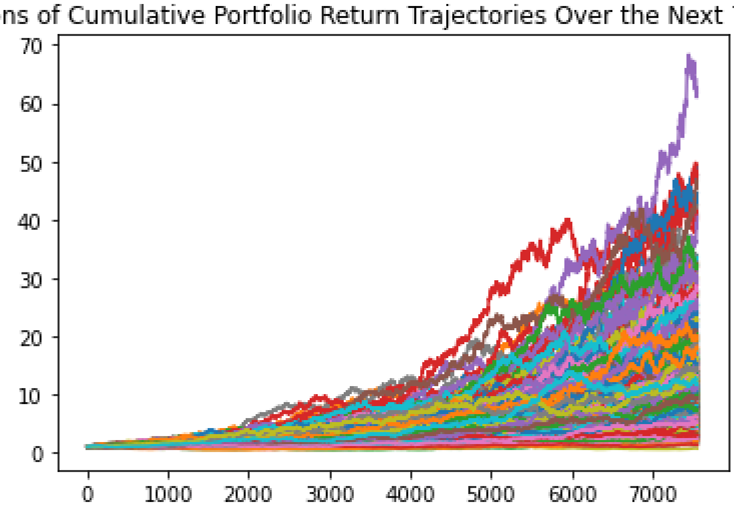
	0	1	2	3	4	5	6	7	8
0	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000
1	1.001645	0.992703	1.007377	1.000388	0.989756	1.003218	1.004473	1.002463	1.00948
2	0.992594	0.985863	0.992831	1.011940	0.983664	0.999882	1.014108	0.995428	1.02062
3	0.983252	0.999238	0.982704	1.011276	0.985572	0.995000	1.009633	0.993150	1.01729
4	0.975628	0.998319	0.990267	1.020897	1.002405	0.995821	1.008684	0.991639	1.00885
...	...	...	...	...	...	...	...	...	.
7556	4.055030	10.561309	7.138171	7.981635	7.990880	6.624006	31.415342	6.145570	3.88904
7557	4.069674	10.516493	7.108238	7.978051	7.926337	6.581671	31.456940	6.105986	3.86515
7558	4.067665	10.662554	7.201493	7.984972	7.893107	6.576487	31.345215	6.102212	3.86257
7559	4.071070	10.613486	7.172516	8.021627	7.880077	6.603729	31.484276	6.095828	3.90343
7560	4.044567	10.680181	7.166546	7.952109	7.849102	6.569362	31.368471	6.177812	3.92860

7561 rows × 1000 columns

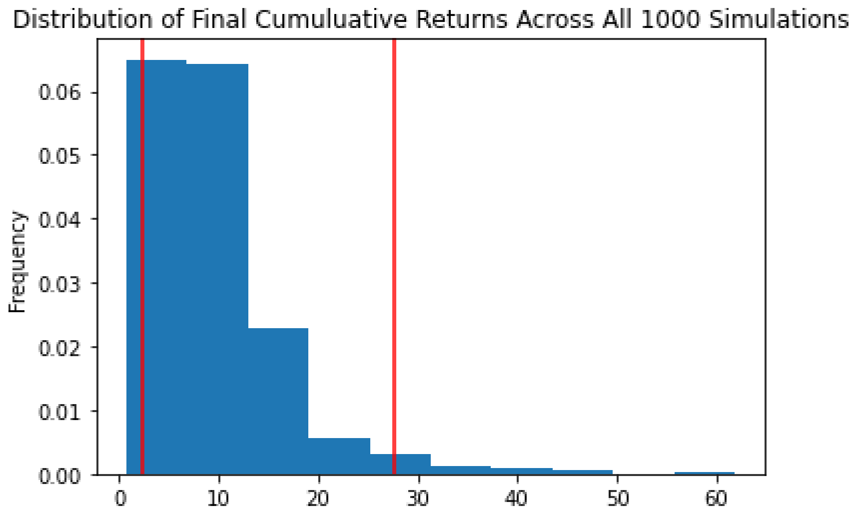
In [216]:

```
# Plot simulation outcomes
line_plot =MC_even_dist.plot_simulation()
```

1000 Simulations of Cumulative Portfolio Return Trajectories Over the Next 7560 Trading Days.



In [217]:
```
# Plot probability distribution and confidence intervals
dist_plot = MC_even_dist.plot_distribution()
```



Retirement Analysis

In [219]:
```
# Fetch summary statistics from the Monte Carlo simulation results
even_tbl = MC_even_dist.summarize_cumulative_return()
# Print summary statistics
print(even_tbl)
```

```
count           1000.000000
mean               9.667317
std                6.717147
min                0.724897
25%                5.404488
50%                8.045266
75%               12.142292
max               61.832012
95% CI Lower       2.287197
95% CI Upper      27.620726
Name: 7560, dtype: float64
```

Calculate the expected portfolio return at the 95% lower and upper confidence intervals based on a `$20,000` initial investment.

In [221]:
```
# Set initial investment
initial_investment = 20000

# Use the lower and upper `95%` confidence intervals to calculate the range of the possible outcomes of our $20,000
ci_lower = round(even_tbl[8]*20000,2)
ci_upper = round(even_tbl[9]*20000,2)


# Print results
print(f"There is a 95% chance that an initial investment of ${initial_investment} in the portfolio"
      f" over the next 30 years will end within in the range of"
      f" ${ci_lower} and ${ci_upper}")
```

```
There is a 95% chance that an initial investment of $20000 in the portfolio over the next 30 years will end within in the range of $45743.94 and $552414.52
```

Calculate the expected portfolio return at the `95%` lower and upper confidence intervals based on a `50%` increase in the initial investment.

In [222]:
```
# Set initial investment
initial_investment = 20000 * 1.5

# Use the lower and upper `95%` confidence intervals to calculate the range of the possible outcomes of our $30,000
ci_lower = round(even_tbl[8]*30000,2)
ci_upper = round(even_tbl[9]*30000,2)

# Print results
print(f"There is a 95% chance that an initial investment of ${initial_investment} in the portfolio"
      f" over the next 30 years will end within in the range of"
      f" ${ci_lower} and ${ci_upper}")
```

```
There is a 95% chance that an initial investment of $30000.0 in the portfolio over the next 30 years will end within in the range of $68615.91 and $828621.78
```