# Module 5 Challenge

Start Assignment

Due 23 Sep by 23:59 Points 100 Submitting a text entry box or a website url

Submission
Not submitted!
Submission details
Grade 0/100 pts provided
Graded anonymously no
Comments
No comments


## Background

You've decided to start a fintech consulting firm that focuses on projects to benefit local communities. You just won your first contract with a large credit union. The project entails building a tool to help credit union members evaluate their financial health. Specifically, the credit union board wants the members to be able to do two things. First, they should be able to assess their monthly budgets. Second, they should be able to forecast a reasonably effective retirement plan based on their current holdings of cryptocurrencies, stocks, and bonds. *The chief technology officer (CTO) of the credit union wants you to develop a prototype application to present at its next assembly.*

## What You're Creating

You'll create two financial analysis tools with a single *Jupyter notebook*:

1. A financial planner for emergencies. The members will be able to use this tool to visualise their current savings. The members can then determine if they have enough reserves for an emergency fund.
2. A financial planner for retirement. This tool will forecast the performance of their retirement portfolio in 30 years. To do this, the tool will make an Alpaca API call via the Alpaca SDK to get historical price data for use in Monte Carlo simulations.

You'll use the information from the Monte Carlo simulation to answer questions about the portfolio in your Jupyter notebook.

## Files

Download the following files to help you get started:

Module 5 Challenge files

## Instructions

This Challenge breaks the instructions into two parts. In Part 1, you'll build the *financial planner for emergencies*. In Part 2, you'll build the financial planner for retirement.

### Part 1: Create a Financial Planner for Emergencies

In this section, you'll create a personal financial planner for emergencies. To develop the prototype, assume the following:

- The average monthly household income for each credit union member is $12,000.
- Each credit union member has a savings portfolio that consists of a cryptocurrency wallet, stocks, and bonds.

Use the starter code in `financial_planning_tools.ipynb` to complete the steps in the following subsections.

#### Evaluate the Cryptocurrency Wallet by Using the Requests Library

In this section, you'll determine the current value of a member's cryptocurrency wallet. You'll collect the current prices for the Bitcoin and Ethereum cryptocurrencies by using the Python Requests library. For the prototype, you'll assume that the member holds the 1.2 Bitcoins (BTC) and 5.3 Ethereum coins (ETH). To do all this, complete the following steps:

1. Create two variables called `btc_qty` and `eth_qty`. Set them equal to `1.2` and `5.3`, respectively.
2. Use the Requests library to get the current price (in Canadian dollars) of Bitcoin (BTC) and Ethereum (ETH) by using the API endpoints that the starter code supplied.
3. Navigate the JSON response object to access the current price of each coin, and store each in a variable.

> **NOTE**
>
> Note the specific identifier for each cryptocurrency in the API JSON response. The Bitcoin identifier is `1`, and the Ethereum identifier is `1027`.

4. Calculate the value, in Canadian dollars, of the current amount of each cryptocurrency and of the entire cryptocurrency wallet.

#### Evaluate the Stock and Bond Holdings by Using the Alpaca SDK

In this section, you'll determine the current value of a member's stock and bond holdings. You'll make an API call to Alpaca via the Alpaca SDK to get the current closing prices of the SPDR S&P 500 ETF Trust (ticker: SPY) and of the iShares Core US Aggregate Bond ETF (ticker: AGG). For the prototype, assume that the member holds 110 shares of SPY, which represents the stock portion of their portfolio, and 200 shares of AGG, which represents the bond portion. To do all this, complete the following steps:

> **IMPORTANT**
>
> Remember to create a `.env` file in your working directory to store the values of your Alpaca API key and Alpaca secret key.

1. Create two variables named `my_agg` and `my_spy` and set them equal to `200` and `110`, respectively.
2. Set the variables for the Alpaca API and secret keys. Using the Alpaca SDK, create the Alpaca `tradeapi.REST` object. In this object, include the parameters for the Alpaca API key, the secret key, and the version number.
3. Set the following parameters for the Alpaca API call:
   - `tickers`: Use the tickers for the member's stock and bond holdings.
   - `timeframe`: Use a time frame of one day.
   - `start_date` and `end_date`: Use the same date for these parameters, and format them with the date of the previous weekday (or `2020-08-07`). This is because you want the one closing price for the most-recent trading day.
4. Get the current closing prices for `SPY` and `AGG` by using the Alpaca `get_bars` function. Format the response as a Pandas DataFrame by including the `.df` property at the end of the `get_bars` function.
5. Navigating the Alpaca response DataFrame, select the `SPY` and `AGG` closing prices, and store them as variables.
6. Calculate the value, in dollars, of the current amount of shares in each of the stock and bond portions of the portfolio, and print the results.

#### Evaluate the Emergency Fund

In this section, you'll use the valuations for the cryptocurrency wallet and for the stock and bond portions of the portfolio to determine if the credit union member has enough savings to build an emergency fund into their financial plan. To do this, complete the following steps:

1. Create a variable called `monthly_income` and set its value to `12000`.
2. To analyze savings health, create a DataFrame called `df_savings` with two rows. Store the total value in dollars of the crypto assets in the first row and the total value of the shares in the second row.

> **NOTE**
>
> The `df_savings` DataFrame should have one column named `amount` and two rows where `crypto` and `shares` are the index values:
>
> | | amount |
> |---|---|
> | crypto | 19385.586677 |
> | shares | 40616.500000 |

3. Use the `df_savings` DataFrame to plot a pie chart to visualize the composition of personal savings.
4. Use `if` conditional statements to validate if the current savings are enough for an emergency fund. An ideal emergency fund should be equal to three times your monthly income.
   - If total savings are greater than the emergency fund, display a message congratulating the person for having enough money in this fund.
   - If total savings are equal to the emergency fund, display a message congratulating the person on reaching this financial goal.
   - If total savings are less than the emergency fund, display a message showing how many dollars away the person is from reaching the goal.

### Part 2: Create a Financial Planner for Retirement

In this section, you'll use the Alpaca API to get historical closing prices for a retirement portfolio. You'll then run Monte Carlo simulations to forecast the portfolio performance 30 years from now. You'll use the simulated data to answer questions in your *Jupyter notebook* about the portfolio.

Use the starter code in `financial_planning_tools.ipynb` to complete the steps in the following subsections.

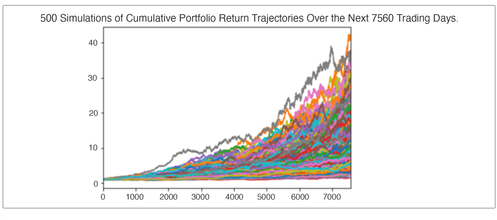
#### Create the Monte Carlo Simulation

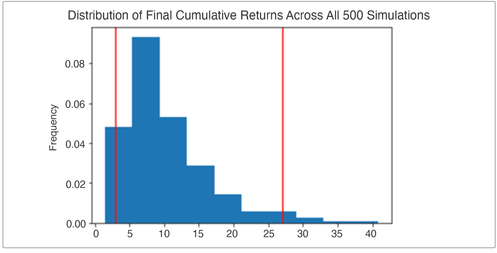
In this section, you'll use the MCForecastTools library to create a Monte Carlo simulation for the member's savings portfolio. To do this, complete the following steps:

1. Use the Alpaca API to fetch five years historical closing prices for a traditional `60-40` portfolio using the `SPY` and `AGG` tickers to represent the `60%` stocks (`SPY`) and `40%` bonds (`AGG`) composition of the portfolio. Make sure to convert the API output to a DataFrame and preview the output.

> **NOTE**
>
> In Monte-Carlo Simulation, getting data as far back as possible matters, because if we simulate using only small amounts of data during a recent time when markets are booming, or instead falling precipitously, a Monte-Carlo Analysis will inadvertently extrapolate this temporary market movement too far into the future. Getting data over a longer time period mitigates this effect.

2. Configure and execute a Monte Carlo Simulation of `500` runs and `30` years for the `60-40` portfolio.
3. Plot the simulation results and the probability distribution/confidence intervals.




#### Analyze the Retirement Portfolio Forecasts

1. Fetch the summary statistics from the Monte Carlo simulation results.
2. Given an initial investment of `$20,000`, calculate the expected portfolio return in dollars at the `95%` lower and upper confidence intervals.
3. Calculate the expected portfolio return at the `95%` lower and upper confidence intervals based on a `50%` increase in the initial investment.

#### Forecast Cumulative Returns in 10 Years

The CTO of the credit union is impressed with your work on these planning tools but wonders if 30 years is a long time to wait until retirement. So, your next task is to adjust the retirement portfolio and run a new Monte Carlo simulation to find out if the changes will allow members to retire earlier.

For this new Monte Carlo simulation, do the following:

- Forecast the cumulative returns for 10 years from now. Because of the shortened investment horizon (30 years to 10 years), the portfolio needs to invest more heavily in the riskier asset—that is, stock—to help accumulate wealth for retirement.
- Adjust the weights of the retirement portfolio so that the composition for the Monte Carlo simulation consists of 20% bonds and 80% stocks.
- Run the simulation over 500 samples, and use the same data that the API call to Alpaca generated.
- Based on the new Monte Carlo simulation, answer the following questions in your Jupyter notebook:
  - Using the current value of only the stock and bond portion of the member's portfolio and the summary statistics that you generated from the new Monte Carlo simulation, what are the lower and upper bounds for the expected value of the portfolio (with the new weights) with a 95% confidence interval?
  - Will weighting the portfolio more heavily toward stocks allow the credit union members to retire after only 10 years?

## Requirements

### Evaluate the Cryptocurrency Wallet by Using the Requests Library (10 points)

To receive all points, you must:

- Create a variable named `monthly_income` and set the value to `12000`. (2 points)
- Use the Requests library to get the current price (in US dollars) of Bitcoin (BTC) and Ethereum (ETH). (2 points)
- Navigate the JSON response object and store each current coin price in its respective variable. (3 points)
- Calculate the value (in US dollars) of the current amount of each cryptocurrency. (3 points)

### Evaluate the Stock and Bond Holdings by Using the Alpaca SDK (10 points)

To receive all points, you must:

- Create an `.env` file to store the values of the Alpaca API key and the Alpaca secret key. (1 point)
- Create a `tradeapi.REST` object and parameters for the Alpaca API key, including the Alpaca secret key and version. (1 point)
- Set the parameters for the Alpaca API call: `tickers`, `timeframe`, `start_date`, and `end_date`. (2 points)
- Get the closing price for `SPY` and `AGG` by using the Alpaca `get_bars` function, and then use `.df` to format as a Pandas DataFrame. (2 points)
- Set the Alpaca response DataFrame for `SPY` and `AGG` as the variable. (2 points)
- Calculate the value (in US dollars) of each stock with the current amount of shares. (2 points)

### Evaluate the Emergency Fund (20 points)

To receive all points, you must:

- Create a Python list named `savings_data` containing two elements: the total value of the cryptocurrency wallet and the total value of stock and bond portions of the portfolios. (5 points)
- Use the `savings_data` list to create a Pandas DataFrame named `savings_df`. Include the following three parameters: `savings_data`, `columns`, and `index`. (5 points)
- Plot the `savings_df` DataFrame as a pie chart that visualizes the composition of each member's portfolio. (5 points)
- Use Python to determine if the current portfolio has enough funds to create an emergency fund that is three times the monthly income of the member. Display a print message that corresponds to the status of emergency funds available in the portfolio. (5 points)

### Create the Monte Carlo Simulation (20 points)

To receive all points, you must:

- Make an API call via the Alpaca SDK to get 10 years of historical closing prices for a 60/40 portfolio: 60% stocks (SPY) and 40% bonds (AGG). (5 points)
- Run a Monte Carlo simulation for 500 samples and 30 years for the 60/40 portfolio and then plot the results. (5 points)
- Plot the probability distribution and confidence interval. (5 points)
- Generate summary statistics for the Monte Carlo simulation. (5 points)

### Analyze the Retirement Portfolio Forecasts (5 points)

To receive all points, you must:

- Answer the following question in your Jupyter notebook:
  - What are the lower and upper bounds for the expected value of the portfolio with a 95% confidence interval? (5 points)

### Forecast Cumulative Returns in 10 Years (5 points)

To receive all points, you must:

- Answer the following questions in your Jupyter notebook:
  - Using the current value of the stock and bond portion of the member's portfolio, as well as the summary statistics that you generated from the new Monte Carlo simulation, what are the lower and upper bounds for the expected value of the portfolio (with the new weights) with a 95% confidence interval? (2 points)
  - Will weighting the portfolio more heavily toward stocks allow the credit union members to retire after only 10 years? (3 points)

### Coding Conventions and Formatting (10 points)

To receive all points, your code must:

- Place imports at the top of the file, just after any module comments and docstrings, and before module globals and constants. (3 points)
- Name functions and variables with lowercase characters, with words separated by underscores. (2 points)
- Follow DRY (Don't Repeat Yourself) principles, creating maintainable and reusable code. (3 points)
- Use concise logic and creative engineering where possible. (2 points)

### Deployment and Submission (10 points)

To receive all points, you must:

- Submit a link to a GitHub repository that's cloned to your local machine and that contains your files. (4 points)
- Use the command line to add your files to the repository. (3 points)
- Include appropriate commit messages in your files. (3 points)

### Comments (10 points)

To receive all points, your code must:

- Be well commented with concise, relevant notes that other developers can understand. (10 points)

## Submission

To submit your Challenge assignment, click Submit, and then provide the URL of your GitHub repository for grading.

> **NOTE**
>
> You are allowed to miss up to two Challenge assignments and still earn your certificate. If you complete all Challenge assignments, your lowest two grades will be dropped. If you wish to skip this assignment, click Next, and move on to the next module.

Comments are disabled for graded submissions in Bootcamp Spot. If you have questions about your feedback, please notify your instructional staff or your Student Success Manager. If you would like to resubmit your work for an additional review, you can use the Resubmit Assignment button to upload new links. You may resubmit up to three times for a total of four submissions.