Assignment 3

Write your name and email

- David Shen
- ds6870@nyu.edu

Exercises

We will start by redoing the dataset construction of Assignment 2. (Feel free to use your previous code!)

Start by importing pandas, numpy, maplotlib, and loading the data set.

The dataset has address

`url='https://github.com/amoreira2/Fin418/raw/refs/heads/main/assets/data/Retuns50stocks.xlsx'`

I strongly recommend you download first and look at the data set.

You should use `read_excel` to get the data that contains t0 stocks plus the market

See here:https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_excel.html .

Do the followings:

1. Import this dataframe as `df`
    - Use "skip_rows" to skip the initial rows before the data.
    - Figure out what is the code for missing value and change the option `na_values` appropriately
2. Change the name of the column with the date information to "Date"
3. Use `to_datetime` so python understand the column date as a datetime object (you will have to use the option format)
4. Set date as index
5. convert the date from the start of the month to end of the month.
6. Drop any date with a missing observations (it will be just one date)
7. Find the market in the columns and save the stocks in a data frame (df) and the market in a different data frame called df_market

```
# your code below
# this imports the relevant libraries
import numpy as np
```

```
import pandas as pd
import matplotlib.pyplot as plt
from pandas.tseries.offsets import MonthEnd

# this points to the location of the data
url='https://github.com/amoreira2/Fin418/raw/refs/heads/main/assets/data/Retuns50stocks.xlsx'


# Import the data
df_ind = pd.read_excel(
    url,
    sheet_name='Retuns50stocks',
    na_values=[],
    usecols="A:AZ"
)


# Rename the column with date information
df_ind.rename(columns={df_ind.columns[0]: 'date'}, inplace=True)

# Convert the date column to datetime
df_ind['date'] = pd.to_datetime(df_ind['date'], format='%Y%m') + MonthEnd(0)

# Set date as the index
df_ind.set_index('date', inplace=True)

df_ind = df_ind.dropna()

# Save market data
df_market = df_ind['Market']

# Check the dataframe
df_ind.info()
# your code below
df_ind.head()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 180 entries, 2000-01-31 to 2014-12-31
Data columns (total 51 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   CTL     180 non-null    float64
 1   T       180 non-null    float64
 2   CSCO    180 non-null    float64
 3   FCX     180 non-null    float64
 4   XL      180 non-null    float64
 5   IVZ     180 non-null    float64
 6   AMT     180 non-null    float64
 7   WHR     180 non-null    float64
 8   IR      180 non-null    float64
 9   WFT     180 non-null    float64
 10  YUM     180 non-null    float64
 11  CVS     180 non-null    float64
 12  DD      180 non-null    float64
 13  TYC     180 non-null    float64
 14  EL      180 non-null    float64
 15  MUR     180 non-null    float64
 22  DTE     180 non-null    float64
 23  S2H     180 non-null    float64
 24  PSA     180 non-null    float64
 25  EXC     180 non-null    float64
 26  TKR     180 non-null    float64
 27  CMA     180 non-null    float64
 28  ORCL    180 non-null    float64
 29  MS      180 non-null    float64
 32  AGN     180 non-null    float64
 33  MMM     180 non-null    float64
 34  ETFC    180 non-null    float64
 35  CAR     180 non-null    float64
 36  MDR     180 non-null    float64
 37  NOV     180 non-null    float64
 38  PCH     180 non-null    float64
 39  BAX     180 non-null    float64
 40  JCI     180 non-null    float64
 49  NU      180 non-null    float64
 50  Market  180 non-null    float64
dtypes: float64(51)
memory usage: 73.1 KB
```

Exercise 1. Portfolio Moments

Construct a portfolio that equal weights the first five stocks. Compute this portfolio variance in this sample.

```
# your code below

weights = np.ones(5,)/5
portfolio_return = df_ind.iloc[:, :5]@weights
portfolio_return.var()
```

```
0.0045359248764002841
```

Exercise 2. Annualize it

Report this number in yearly units

```
portfolio_return.var()*12
```

```
0.054471540851680169
```

Exercise 3. Function 1

Construct a function that takes as input the number of stocks in the portfolio, lets call that parameter N, and outputs the (yearly) variance of the portfolio that equal weight the first N stocks

```
# your code below

def var_of_portfolio(N):
  weights = np.ones(N,)/N
  portfolio_return = df_ind.iloc[:, :N]@weights
  return portfolio_return.var()*12
```

Exercise 4. Function 2

Construct a function that takes as input the number of stocks in the portfolio and also, lets call that parameter N, and outputs the (yearly) variance of the portfolio that equal weight N randomly picked stocks

date	CTL	T	CSCO	FCX	XL	IVZ	AMT	WHR	IR	W
2000-										

```
# your code below
# print(df_ind.iloc[:, np.random.choice(df_ind.column)].info())

def rand_var_of_portfolio(N):
  weights = np.ones(N,)/N
  portfolio_return = df_ind.sample(n=N, axis=1)@weights
  return portfolio_return.var()*12
```

2005-05-31	10.3980	-0.2853	-17.8724	-4.5455	25.8793	-8.7719	-20.2685	-12.4338	-2.5672	6.00

5 rows × 51 columns

Exercise 5. A simulation

Construct a function that that takes as an input the number of stocks N and then use the function above (excercise 4) to simulate 100 different portfolios and output the average of these 100 portfolios

```
import numpy as np

# your code below
def simulate_var(N):
  variances = []
  for i in range(100):
    variances.append(rand_var_of_portfolio(N))
  return np.mean(variances)
```

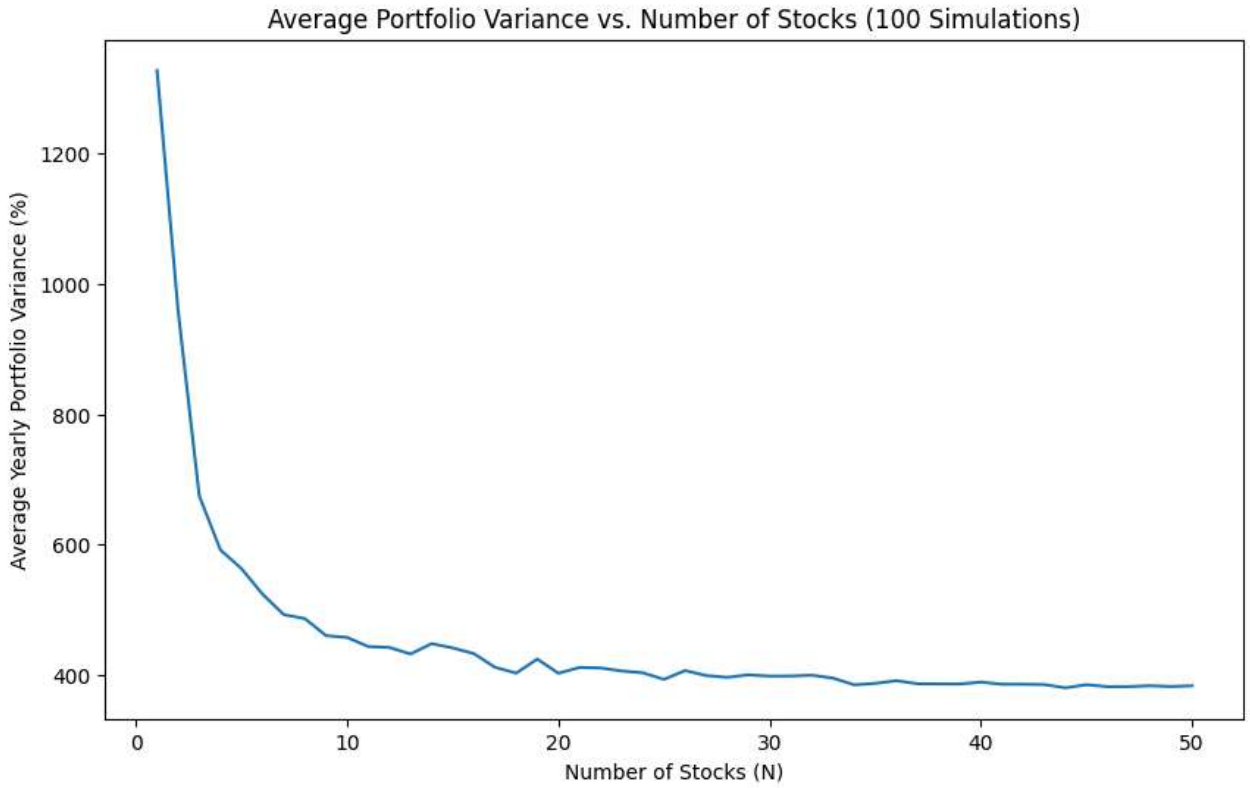
Exercise 6. A plot

Call the function above for N=1 to N=50 and plot the variance as function of N

```
# your code below


fig, ax = plt.subplots(figsize=(10, 6))
total_var = [simulate_var(N) for N in range(1, 51)]
ax.plot(range(1, 51), total_var, label="Total Variance")
ax.set_title('Average Portfolio Variance vs. Number of Stocks (100 Simulations)')
ax.set_xlabel('Number of Stocks (N)')
ax.set_ylabel('Average Yearly Portfolio Variance (%)')
```

```
Text(0, 0.5, 'Average Yearly Portfolio Variance (%)')
```



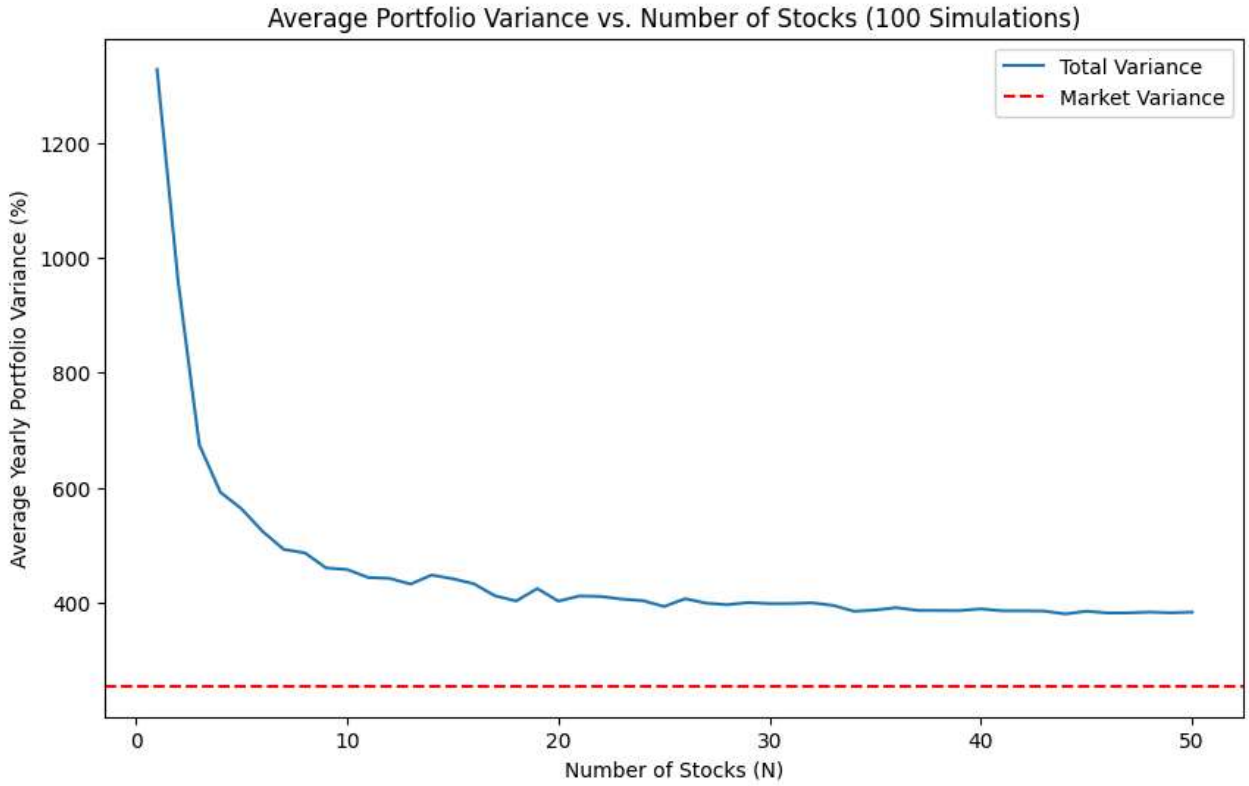
Exercise 7. the market

Add the market variance to the plot as an horizontal line to the plot you made above

```
# your code below
market_variance = df_market.var() * 12
print(f"Market Variance: {market_variance}")
print(f"Max Variance: {max(total_var)}")

ax.axhline(market_variance, color='r', linestyle='--', label='Market Variance')
ax.legend()
fig
```

```
Market Variance: 254.90037959357852
Max Variance: 1327.4777987636146
```



Exercise 8. Interpretation

Explain what you see in the plot. Try to articulate what you think is happening

The variance is very high when we have few stocks. For example, when we have N of 1, we have annual variance of around 1300%. For each stock that we add, the variance decreases. But there are diminishing returns, and the rate that the variance decreases is slowing. They are all above the market variance because there is idiosyncratic risk. I think all of these variance numbers are very high.

Exercise 9. A decomposition

We now want to build a function that outputs only the terms due to the variance terms

That is, instead of computing $Var(WR) = W@Var(R)@W = \sum_i \sum_j W_i W_j Cov(R_i, R_j)$ which gives you the variance of a portfolio with weight W,

I want you do compute $\sum_i W_i^2 Var(R_i)$, this component of the portfolio variance that comes only from the variance terms

Then subtract this term from the total variance term

Show the three lines in a plot as functions of N (1. the total variance (and also the market like in exercise 7), 2. the component coming only from the variance terms, and 3. the residual (the component due to the covariance terms))

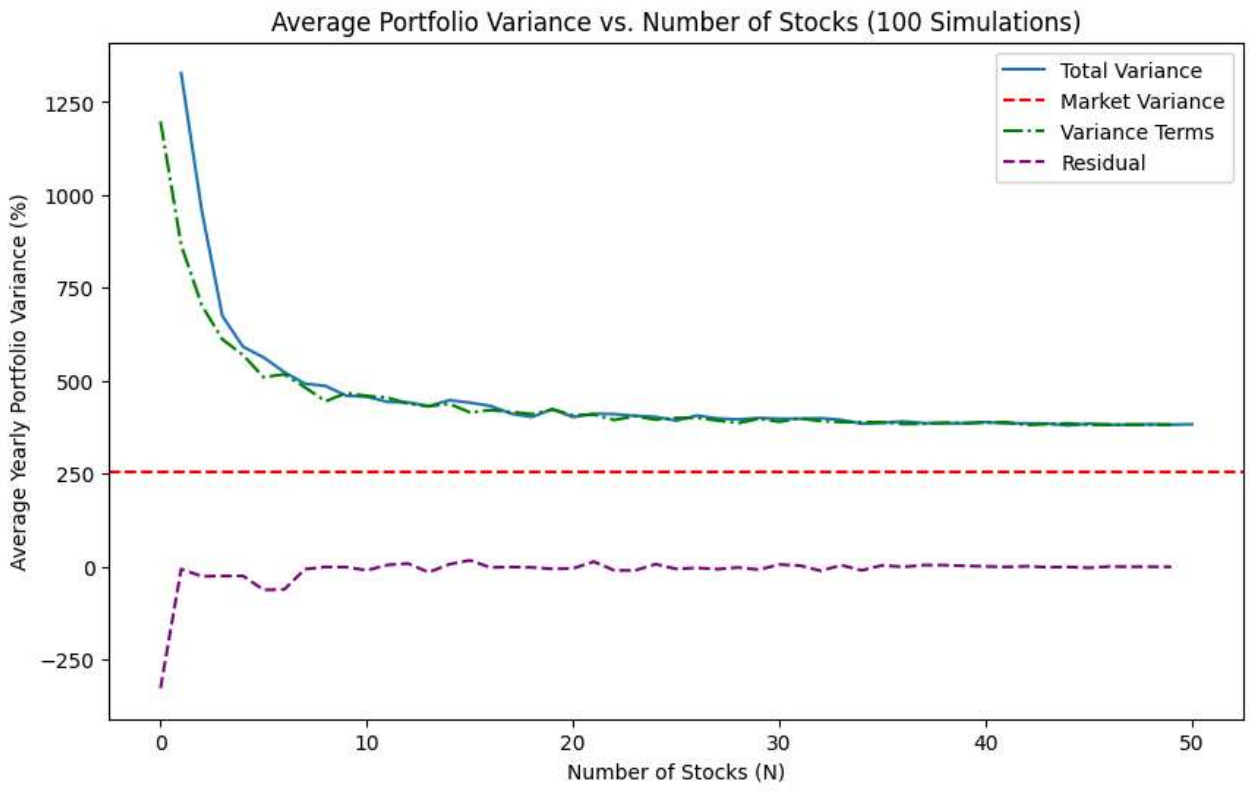
```
# your code below


def ann_var_terms(N):
  weights = np.ones(N,)/N
  sum = np.sum(weights**2 * df_ind.sample(n=N, axis=1))
  return sum*12


def simulate_var_terms(N):
  variances = []
  for i in range(100):
    variances.append(rand_var_of_portfolio(N))
  return np.mean(variances)
```

```
var_terms = [simulate_var_terms(N) for N in range(1, 51)]
residual = [simulate_var(N) - simulate_var_terms(N) for N in range(1, 51)]

ax.plot(var_terms, label='Variance Terms', color='green', linestyle='-.')
ax.plot(residual, label='Residual', color='purple', linestyle='--')
ax.legend()
fig
```



Exercise 10. Interpretation 2

looking at this new plot, what do you learn? What does that imply about portfolio construction?

Adding more stocks lowers the total variance of the portfolio but the residual from covariance terms does not decrease. So systematic risk from the co movement of the stocks with the market is not

diversifiable.

Exercise 11. Factor betas

Randomly select 10 stocks from `df` and draw a figure that plots the returns of these industries along with the market return.

Just by looking, which seems seems to have the highest beta?

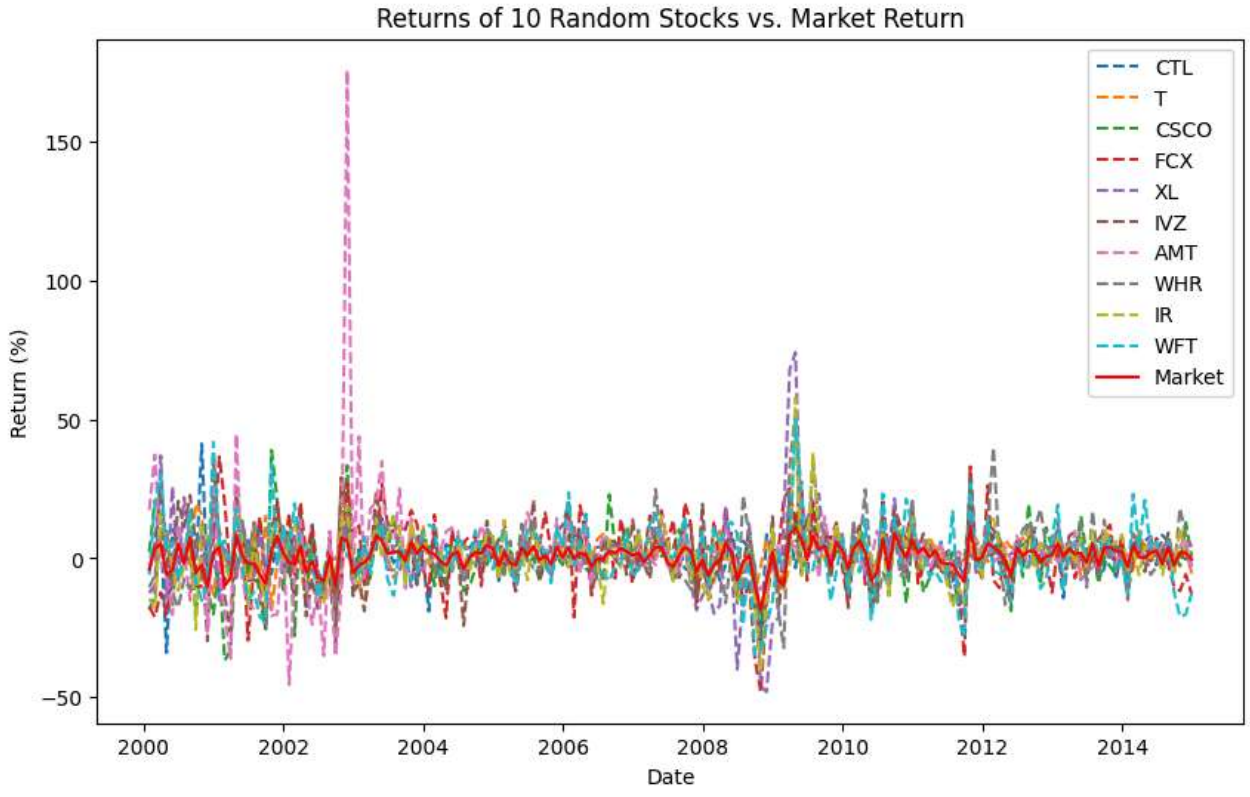
Why is it important to measure beta correctly? i.e. explain how you can use beta to improve a trade that you have in one of these companies

```
# Your code below
plot, ax = plt.subplots(figsize=(10, 6))

# stocks = df_ind.sample(n=10, axis=1)
stocks = df_ind.iloc[:, :10]

ax.plot(stocks, linestyle='--')
ax.plot(df_market, label='Market', color='red', linestyle='-')
ax.legend(stocks.columns.tolist() + ['Market'])
ax.set_title('Returns of 10 Random Stocks vs. Market Return')
ax.set_xlabel('Date')
ax.set_ylabel('Return (%)')
```

```
Text(0, 0.5, 'Return (%)')
```



It's hard to see, but I think WFT has the highest beta. You must measure beta correctly because it is the basis of how much the stock is moving with the market. You can use betas to see how much of a

position you have in a market. For example, if the portfolio is at 1.5 beta, you can short a company with 1 beta to be at 0.5 beta.

Exercise 12. Factor betas 2

Run regressions of all stocks in `df` on the market return you saved in `df_market`. Include intercepts in the regressions.

Just follow the code below to run regressions.

The coefficients to the market return are the betas to each industry.

Now suppose you are a fund manager and you have a mandate to keep your beta equal to 0.5.

Provide portfolio weights based on the regressions above to hit the mandate beta.

Give at least *five* such portfolios that satisfy that restriction. At least one of these portfolios has to play in more than one stock at a time.

```
import statsmodels.api as sm

X = df_market
X = sm.add_constant(X)  # Adds a constant term to the predictor
y = df_ind['ACAS']
model = sm.OLS(y, X).fit(dropna=True)
print(model.summary())

# Your code below
df_alphas = pd.DataFrame()
df_betas = pd.DataFrame()
for stock in df_ind.columns:
  model = sm.OLS(df_ind[stock], X).fit(dropna=True)
  df_alphas.loc['alpha', stock] = model.params['const']
  df_betas.loc['beta', stock] = model.params['Market']
  # print(model.params['Market'])

df_betas
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   ACAS   R-squared:                       0.226
Model:                            OLS   Adj. R-squared:                  0.221
Method:                 Least Squares   F-statistic:                     51.87
Date:                Wed, 18 Feb 2026   Prob (F-statistic):           1.61e-11
Time:                        17:17:34   Log-Likelihood:                -708.16
No. Observations:                 180   AIC:                             1420.
Df Residuals:                     178   BIC:                             1427.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.8277      0.932      0.888      0.376      -1.012       2.668
Market         1.4529      0.202      7.202      0.000       1.055       1.851
==============================================================================
Omnibus:                       43.383   Durbin-Watson:                   2.135
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              432.197
Skew:                           0.466   Prob(JB):                     1.41e-94
Kurtosis:                      10.534   Cond. No.                         4.65
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

	CTL	T	CSCO	FCX	XL	IVZ	AMT	WHR	IR	
beta	0.726946	0.529153	1.530779	1.569894	1.438681	2.036247	1.352478	1.442238	1.46115	1.51

1 rows × 51 columns

```
# Find five portfolios

target_beta = 0.5

portfolios = [df_betas.sample(n=2, axis=1) for _ in range(5)]

weights = []

# balance weights
for p in portfolios:
  betas = p.values.flatten()
  A = np.array([
      [1., 1.],
      [betas[0], betas[1]]
  ])
  b = np.array([1., target_beta])
  w = np.linalg.solve(A, b)
  weights.append(pd.Series(w, index=p.columns))

# print
print("Five portfolios:")
for i, w in enumerate(weights):
  print(f"Portfolio {i+1}: \n{w}")
  print(f"Sum of weights: {w.sum()}")
  print(f"Beta: {portfolios[i].loc['beta'].values@w}")
  print()
```

```
Five portfolios:
Portfolio 1:
CTL   -8.068546
YUM    9.068546
dtype: float64
Sum of weights: 1.0
Beta: 0.5

Portfolio 2:
ACAS    0.1741
EXC     0.8259
dtype: float64
Sum of weights: 1.0
Beta: 0.5

Portfolio 3:
ACAS    1.501523
CAR    -0.501523
dtype: float64
Sum of weights: 0.9999999999999999
Beta: 0.5

Portfolio 4:
CTAS    1.417129
CCI    -0.417129
dtype: float64
Sum of weights: 1.0
Beta: 0.5000000000000002

Portfolio 5:
CTAS    1.539373
TKR    -0.539373
dtype: float64
Sum of weights: 1.0
Beta: 0.5
```

Exercise 13. Factor risk

Find beta-hedged portfolio returns using the five mandate portfolios you provided in Exercise 12.

Draw a plot drawing these portfolio returns together.(i.e. spy in the x-axis, hedge portfolio in the y)

Do they show any co-movement? Can you say they are all "risk free"? In what sense they are free of risk and what sense they are not?

```
# Your code below

portfolio_returns = []
for i, p_df in enumerate(portfolios):
  tickers = p_df.columns.tolist()
  returns = df_ind[tickers]
  portfolio_returns.append(returns@weights[i])

plot, ax = plt.subplots(figsize=(10, 6))

for i, pr in enumerate(portfolio_returns):
    tickers = portfolios[i].columns.tolist()
    ax.plot(df_market, pr, '.', label=f'Portfolio {i+1} Returns [{tickers[0]} + {tickers[1]}]
```

```
ax.set_title('Portfolio Returns vs. Market Returns (Beta-Hedged)')
ax.set_xlabel('Market Returns (%)')
ax.set_ylabel('Portfolio Returns (%)')
ax.legend()
ax.grid(True)
plot.show()
```