```python
In [224]: # Initialize Otter Grader
import otter
grader = otter.Notebook()
```

Data X Berkeley

NAME: Talia Arauzo

STUDENT ID: 3031791660

Data-X Fall 2020: Homework 04

Linear regression and matplotlib.

Author: Ming-Yen Kao.

In this homework, you will do some exercises with regression and plotting.

Only submit your jupyter notebook to Gradescope

Question 1 - Data Pre-processing

Data Source of SP500.csv is Yahoo Finance; Data Source of Gold.csv is Gold.org; Data Source of 30YTBond.csv is Macrotrends.net. SP500 is a stock market index that measures the stock performance of 500 large companies listed on stock exchanges in the United States. The unit of Gold price is USD per oz. 30YTBond stands for US Treasury Bond future price with maturity of 30 years.

1a) Use pandas to load data from .csv to DataFrame. Please load 'SP500.csv' to df_SP500, load 'Gold.csv' to df_Gold, and load '30YTBond.csv' to df_30YTB.

```python
In [58]: # Load required modules
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

```python
In [112]: df_SP500 = pd.read_csv("SP500.csv")
df_Gold = pd.read_csv("Gold.csv")
df_30YTB = pd.read_csv("30YTBond.csv")
```

```python
In [60]: print("SP500:")
display(df_SP500.head())
print("Gold:")
display(df_Gold.head())
print("30YTBond:")
display(df_30YTB.head())
```

SP500:

	Date	Open	High	Low	Close	Adj Close	Volume
0	1927-12-01	17.660000	17.660000	17.660000	17.660000	17.660000	0
1	1928-01-01	17.760000	17.760000	17.260000	17.570000	17.570000	0
2	1928-02-01	17.530001	17.629999	16.950001	17.260000	17.260000	0
3	1928-03-01	17.299999	19.280001	17.299999	19.280001	19.280001	0
4	1928-04-01	18.910000	19.750000	18.910000	19.750000	19.750000	0

Gold:

	Date	US dollar	Euro	Pound sterling	Japanese yen
0	1969/12/31	35.20	22.24	14.66	12592.44
1	1970/1/30	34.99	23.20	14.54	12496.35
2	1970/2/27	35.02	23.26	14.55	12529.95
3	1970/3/31	35.30	23.42	14.67	12627.65
4	1970/4/30	35.85	23.79	14.93	12858.36

30YTBond:

	Date	value
0	1977/8/22	102.375
1	1977/8/23	102.781
2	1977/8/24	102.656
3	1977/8/25	103.094
4	1977/8/26	103.062

```python
In [61]: grader.check("q1a")
```

Out[61]: All tests passed!

1b) Only leave 'Date' and 'US dolloar price' in df_Gold and only leave 'Date' and 'closing price' in df_SP500. In other word 'Open', 'High', 'Low', 'Adj Close', and 'Volume' have to be dropped from df_SP500. 'Euro', 'Pound sterling', and 'Japanese yen' have to be dropped from df_Gold.

```python
In [113]: df_SP500 = df_SP500[["Date", "Close"]]
df_Gold = df_Gold[["Date", "US dollar"]]
```

```python
In [63]: print("SP500:")
display(df_SP500.head())
print("Gold:")
display(df_Gold.head())
```

SP500:

	Date	Close
0	1927-12-01	17.660000
1	1928-01-01	17.570000
2	1928-02-01	17.260000
3	1928-03-01	19.280001
4	1928-04-01	19.750000

Gold:

	Date	US dollar
0	1969/12/31	35.20
1	1970/1/30	34.99
2	1970/2/27	35.02
3	1970/3/31	35.30
4	1970/4/30	35.85

```python
In [64]: grader.check("q1b")
```

Out[64]: All tests passed!

1c) Sample the very first price of the asset in each month to represnet that month.

For example, 1999/1/2 price of bananan is 90; 1999/1/5 price of bananan is 85; 1999/1/6 price of bananan is 91. You should use 90 to represent the price of 1999/1. Please convert the date to Pandas datetime formate with day always = 1 ex: 1970-02-01 or 2000-11-01. Replace the original data frames of df_SP500, df_Gold, and df_30YTB with the sampled data frames. For example, we have [1999/1/2: 90, 1999/1/5: 85, 1999/1/6: 91, 1999/2/4: 70, 1999/2/7: 89]. After sampling, we should have [1999/1/1: 90, 1999/2/1: 70].

- Remember to keep the Date in the format of Pandas timestamp!

df_30YTB after sampling should be

```python
In [116]: # df_SP500
df_SP500["Date"] = pd.to_datetime(df_SP500["Date"])
df_SP500["year_month"] = pd.DatetimeIndex(df_SP500["Date"]).to_period('M')
df_SP500 = df_SP500[df_SP500["Date"].isin(df_SP500.groupby("year_month")["Date"].min())].drop(columns
= ["year_month"])
```

```python
In [118]: # df_Gold
df_Gold["Date"] = pd.to_datetime(df_Gold["Date"])
df_Gold["year_month"] = pd.DatetimeIndex(df_Gold["Date"]).to_period('M')
df_Gold = df_Gold[df_Gold["Date"].isin(df_Gold.groupby("year_month")["Date"].min())].drop(columns = [
"year_month"])
```

```python
In [114]: # df_30YTB
df_30YTB["Date"] = pd.to_datetime(df_30YTB["Date"])
df_30YTB["year_month"] = pd.DatetimeIndex(df_30YTB["Date"]).to_period('M')
df_30YTB = df_30YTB[df_30YTB["Date"].isin(df_30YTB.groupby("year_month")["Date"].min())].drop(columns
= ["year_month"])
```

```python
In [119]: grader.check("q1c")
```

Out[119]: All tests passed!

Question 2 - Data Visualization & Linear Regression

```python
In [120]: import matplotlib.pyplot as plt
```

2a) Oberve the example plot below and try to reproduce it. y-left is df_SP500['Date'] in linear scale and y-right is df_SP500['Date'] in log scale.

Specs:

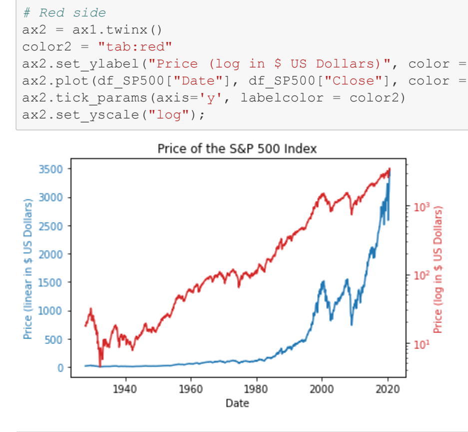
1. x-label and x-scale in black
2. y-left label, y-left scale, and the corresponding line-plot in blue
3. y-right label, y-right scale, and the corresponding line-plot in red
4. plot title is 'Price of the S&P 500 Index'
5. y-left label is 'Price (linear in $ US Dollars)'
6. y-right label is 'Price (log in $ US Dollars)'
7. x label is 'Date'

Some helpful functions: axis.set_xlabel(), axis.set_ylabel(), axis.tick_params(), and axis.twinx()

```python
In [146]: fig, ax1 = plt.subplots()
plt.title("Price of the S&P 500 Index")

# Blue side
color1 = "tab:blue"
ax1.set_xlabel("Date")
ax1.set_ylabel("Price (linear in $ US Dollars)", color = color1)
ax1.plot(df_SP500["Date"], df_SP500["Close"], color = color1)
ax1.tick_params(axis = 'y', labelcolor = color1)

# Red side
ax2 = ax1.twinx()
color2 = "tab:red"
ax2.set_ylabel("Price (log in $ US Dollars)", color = color2)
ax2.plot(df_SP500["Date"], df_SP500["Close"], color = color2)
ax2.tick_params(axis='y', labelcolor = color2)
ax2.set_yscale("log");
```




```python
In [147]: grader.check("q2a")
```

Out[147]: All tests passed!

2b) Linear regression of S&P 500 Index.

1. Convert the 'Date' in df_SP500 to numpy.array and convert the unit to day as 'x' and convert the 'Close' in df_SP500 to numpy.array as 'y'.
2. Shift the date so that the first element in array 'x' is 0.
3. Follow the linear regression procedure in the slides https://datax.berkeley.edu/wp-content/uploads/2020/09/slides-m100-linear-regression.pdf. Page 9.
4. Calculate the root mean square error and store it in 'error'.
5. Finally plot the data and the fitting line together as the example plot below.

Specs of plot:

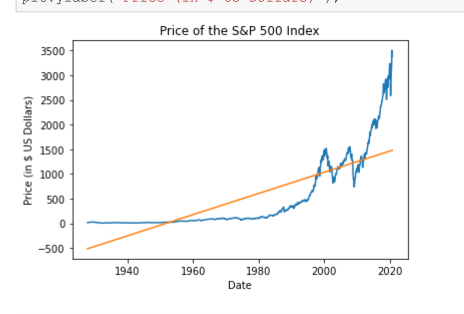
1. Color in default (please plot the data first so that the color of data will be blue)
2. plot title is 'Price of the S&P 500 Index'
3. x-label is 'Date'
4. y-label is 'Price (in $ US Dollars)'

```python
In [194]: # Linear reg data
x = df_SP500["Date"].to_numpy().astype("int") / 86400
x = x - x[0]
y = df_SP500["Close"].to_numpy()
```

```python
In [204]: # Linear reg
E_x = np.mean(x)
E_y = np.mean(y)
cov_xy = np.mean(x * y) - (E_x * E_y)
y_0 = E_y - ((cov_xy / np.var(x)) * E_x)
m = cov_xy / np.var(x)
y_pred = y_0 + (m * x)
error = np.sqrt(np.mean(np.square(y_pred - y)))
print ('Root Mean Square Error: ', error)
```

```
Root Mean Square Error:  434.79997247592337
```

```python
In [205]: # Linear reg plot
plt.plot(df_SP500["Date"], df_SP500["Close"])
plt.plot(df_SP500["Date"], y_pred)
plt.title("Price of the S&P 500 Index")
plt.xlabel("Date")
plt.ylabel("Price (in $ US Dollars)");
```




```python
In [206]: grader.check("q2b")
```

Out[206]: All tests passed!

2c) Exponential regression of S&P 500 Index.

1. Do the same thing as in 2b, but try to use exponential to fit the data this time.
2. Convert the 'Date' in df_SP500 to numpy.array and convert the unit to day as 'x' and convert the 'Close' in df_SP500 to numpy.array as 'y'.
3. Shift the date so that the first element in array 'x' is 0.
4. Use exponential function to fit the data. Calculate the root mean square error and store it in 'error'.
5. Finally plot the data and the fitting line together as the example below.

Hint: apply log to y value so that you can process the data as linear regression in 2(b). Remember to convert it back for the calculation for the 'error' and for the plot.

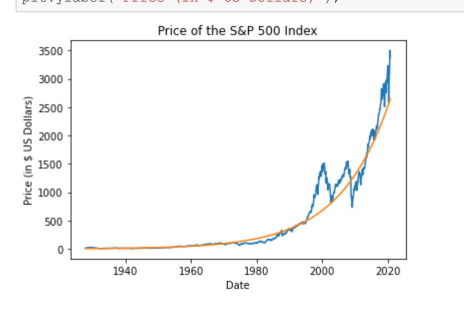
Specs of plot:

1. Color in default (please plot the data first so that the color of data will be blue)
2. plot title is 'Price of the S&P 500 Index'
3. x-label is 'Date'
4. y-label is 'Price (in $ US Dollars)'

```python
In [219]: # Exponential reg
y_log = np.log(y)
E_x = np.mean(x)
E_y = np.mean(y_log)
cov_xy = np.mean(x * y_log) - (E_x * E_y)
y_0 = E_y - ((cov_xy / np.var(x)) * E_x)
m = cov_xy / np.var(x)
y_pred = np.exp(y_0 + (m * x))
error = np.sqrt(np.mean(np.square(y_pred - y)))
print ('Root Mean Square Error: ', error)
```

```
Root Mean Square Error:  184.58637172480607
```

```python
In [222]: # Linear reg plot
plt.plot(df_SP500["Date"], df_SP500["Close"])
plt.plot(df_SP500["Date"], y_pred)
plt.title("Price of the S&P 500 Index")
plt.xlabel("Date")
plt.ylabel("Price (in $ US Dollars)");
```




```python
In [223]: grader.check("q2c")
```

Out[223]: All tests passed!

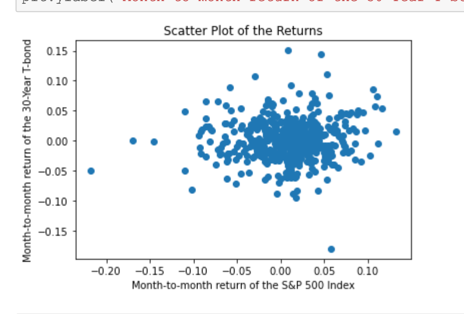
2d) Scatter plot of month-to-month return of S&P 500 Index and 30 Year Treasury Bond.

1. Compute the month-to-month returns on S&P 500 and 30 Year Treasury Bond between 1977-08-01 (inclusive) and 2019-12-31 (inclusive). The month-to-month returns should be in the net-increment. For example, price on 2014-1-1 is 100 and price on 2014-2-1 is 105. The month-to-month return from 2014-1-1 to 2014-2-1 should be 0.05. Another example, if the array of the price of S&P 500 is [100, 105, 100, 105], the month-to-month return array should be [0.05, -0.04761904762, 0.05].
2. Save the month-to-month returns in the format of numpy arrary in ‘x3’ and ‘y3’, respectively. Note that if the dimension of the array of price is n, the dimension of month-to-month return array should be n-1.
3. Finally plot ‘x3’ and ‘y3’ as scatter plot. Try to reproduce the plot below.

Specs of plot:

1. Color in default
2. plot title is 'Scatter Plot of the Returns'
3. x-label is 'Month-to-month return of the S&P 500 Index'
4. y-label is 'Month-to-month return of the 30-Year T-bond'

```python
In [293]: mask_SP500 = (df_SP500["Date"] >= pd.Timestamp(1977, 8, 1)) & (df_SP500["Date"] <= pd.Timestamp(2019,
12, 31))
x3 = df_SP500.sort_values(by = "Date").loc[mask_SP500]["Close"].values
x3 = np.array([(x3[idx + 1] - val)/val for idx, val in enumerate(x3) if idx < len(x3) - 1])
mask_30YTB = (df_30YTB["Date"] >= pd.Timestamp(1977, 8, 1)) & (df_30YTB["Date"] <= pd.Timestamp(2019,
12, 31))
y3 = df_30YTB.sort_values(by = "Date").loc[mask_30YTB]["value"].values
y3 = np.array([(y3[idx + 1] - val)/val for idx, val in enumerate(y3) if idx < len(y3) - 1])
plt.scatter(x3, y3)
plt.title("Scatter Plot of the Returns")
plt.xlabel("Month-to-month return of the S&P 500 Index")
plt.ylabel("Month-to-month return of the 30-Year T-bond");
```




```python
In [294]: print (x3, y3)
```

```
[-2.48008688e-03 -4.34062265e-02  2.69656282e-02  2.84715801e-03
 -6.15141758e-02 -2.47618936e-02  2.49310429e-02  8.54164677e-02
  4.23418353e-03 -1.75853459e-02  5.39097776e-02  2.59237286e-02
 -7.26110943e-03 -9.15740092e-02  1.66397742e-02  1.48891663e-02
  3.97461134e-02 -3.65255779e-02  5.51516105e-02  1.67345218e-03
 -2.63364775e-02  3.86556512e-02  8.74544714e-03  5.30777585e-02
  0.00000000e+00 -6.86059276e-02  4.26242781e-02  1.67671245e-02
  5.76246237e-02 -4.37981765e-03 -1.01794893e-01  4.11402210e-02
  4.65706741e-02  2.69687168e-02  6.50385165e-02  5.83544844e-03
  2.51675280e-02  1.60210586e-02  1.02377053e-01 -3.38742447e-02
 -4.57424295e-02  1.32767345e-02  3.60325730e-02 -2.34558971e-02
 -1.65651685e-03 -1.04079421e-02 -2.21026587e-03 -6.20989698e-02
 -5.38317530e-02  4.91478654e-02  3.65903605e-02 -3.00751489e-02
 -1.75438674e-02 -6.05481801e-02 -1.01671116e-02  4.00143180e-02
 -3.91618423e-02 -2.02895608e-02 -2.29906485e-02  1.15977276e-01
  7.61439197e-03  1.10446796e-01  3.59706698e-02  1.52313579e-02
  3.31342721e-02  1.89951476e-02  3.30947526e-02  7.49868297e-02
 -1.24064592e-02  3.23295771e-02 -3.03030364e-02  1.13188732e-02
  1.01582303e-02 -1.51743475e-02  1.74258083e-02 -8.83414094e-03
 -9.21596474e-03 -3.88593467e-02  1.34979946e-02  5.46557380e-03
 -5.93564500e-02  1.74692125e-02 -1.64511628e-02  1.06332063e-01
 -3.47964377e-03 -6.02648985e-05 -1.51122528e-02  2.23743915e-02
  7.40851449e-02  8.62878114e-03 -2.87001336e-03 -4.59427644e-03
  5.40510532e-02  1.21340172e-02 -4.84757869e-03 -1.19945162e-02
 -3.47240780e-02  4.25088143e-02  6.50615875e-02  4.50610926e-02
  2.36652784e-03  7.14892769e-02  5.27939190e-02 -1.41481377e-02
  5.02292875e-02  1.41095206e-02 -5.86828306e-02  7.11926070e-02
 -8.54386059e-02  5.47293300e-02  2.14771911e-02 -2.82882713e-02
  1.31766896e-01  3.69236189e-02  2.63898652e-02 -1.14502121e-02
  6.03419715e-03  4.79144906e-02  4.82236974e-02  3.49588397e-02
 -2.41661652e-02 -2.17630416e-01 -8.53488645e-02  7.28614797e-02
  4.04322686e-02  4.18174027e-02 -3.33432595e-02  9.42474355e-03
  3.17612613e-03  4.32560109e-02 -5.41137477e-03 -3.86001045e-02
  3.97293340e-02  2.59644621e-02 -1.88908807e-02  1.46875733e-02
  7.11147916e-02 -2.89441489e-02  2.08059624e-02  5.00899388e-02
  3.51374935e-02 -7.92455412e-03  8.83702592e-02  1.55167164e-02
 -6.54436740e-03 -2.51754522e-02  1.65413246e-02  2.14168161e-02
 -6.88172253e-02  8.53904252e-03  2.42549840e-02 -2.68871387e-02
  9.19891902e-02 -8.88636576e-03 -5.22315808e-03 -9.43141838e-02
 -5.11843071e-02 -6.69821297e-03  5.99342138e-02  2.48277574e-02
  4.15177517e-02  6.72811749e-02  2.22028328e-02  3.19799050e-04
  3.86049746e-02 -4.78926292e-02  4.48593432e-02  1.96487843e-02
 -1.91437375e-02  1.18342370e-02 -4.39037087e-02  1.11587855e-01
 -1.99237505e-02  9.58954208e-03 -2.18318627e-02  2.78927146e-02
  9.63957075e-04 -1.73588321e-02  3.93736841e-02 -2.39975300e-02
  9.10559382e-03  2.10628297e-03  3.02618067e-02  1.01077662e-02
  7.04598945e-03  1.04836274e-02  1.86972978e-02 -2.54168102e-02
  2.27174628e-02  7.55230011e-04 -5.32704594e-03  3.44319569e-02
 -9.98793041e-03  1.93929230e-02 -1.29106260e-02  1.00911733e-02
  3.25007452e-02 -3.00449958e-02 -4.57465114e-02  1.15306439e-02
  1.23971434e-02 -2.67908237e-02  3.14899078e-02  3.75986986e-02
 -2.68775353e-02  2.08338164e-02 -3.95046126e-02  1.22991183e-02
  2.42777109e-02  3.60741498e-02  2.73291934e-02  2.79603588e-02
  3.63117118e-02  2.12785442e-02  3.17760404e-02 -3.20238054e-04
  4.00974724e-02 -4.97933494e-03  4.10490026e-02  1.74438741e-02
  3.26173871e-02  6.93370155e-03  7.91656708e-03  1.34314222e-02
  2.28534057e-02  2.25671032e-03 -4.57480172e-02  1.88139351e-02
  5.42033276e-02  2.61010032e-02  7.33761517e-02 -2.15054154e-02
  6.13170392e-02  5.92759001e-03 -4.26140104e-02  5.84055794e-02
  5.85768842e-02  4.34526156e-02  7.81458095e-02 -5.74656318e-02
  5.31535899e-02 -3.44776972e-02  4.45868549e-02  1.57315979e-02
  1.01501768e-02  7.04491930e-02  4.99457142e-02  9.07646925e-03
 -1.88262235e-02  3.94382411e-02 -1.16153270e-02 -1.45796719e-01
  6.23955156e-02  8.02942284e-02  5.91259963e-02  5.63752866e-02
  4.10094415e-02 -3.22825627e-02  3.87942154e-02  3.79440279e-02
 -2.49704809e-02  5.44383310e-02 -3.20460922e-02 -6.25409204e-03
 -2.85517923e-02  6.25395416e-02  1.90618586e-02  5.78438949e-02
 -5.09035487e-02 -2.01080833e-02  9.67198283e-02 -3.07957556e-02
 -2.19150505e-02  2.39335496e-02 -1.63412762e-02  6.06991052e-02
 -5.34829747e-02 -4.94948587e-03 -8.00686100e-02  4.05344553e-03
  3.46365771e-02 -9.22907358e-02 -6.42047105e-02  7.68143618e-02
  5.09018712e-03 -2.50353891e-02 -1.07401501e-02 -6.41084066e-02
 -8.17234060e-02  1.80991114e-02  7.51759043e-02  7.57383419e-03
 -1.55738326e-02 -2.07662113e-02  3.67388936e-02 -6.14176262e-02
 -9.08148108e-03 -7.24553544e-02 -7.90042750e-02  4.88143308e-03
 -1.10024318e-01  8.64488010e-02  5.70696209e-02 -6.03325727e-02
 -2.74146926e-02 -1.70036085e-02  8.35756857e-03  8.10441069e-02
  5.08987097e-02  1.13222145e-02  1.62237024e-02  1.78732034e-02
 -1.19443645e-02  5.49614864e-02  7.12850385e-03  5.07655410e-02
  1.72763870e-02  1.22089733e-02 -1.63589192e-02 -1.67907519e-02
  1.20834502e-02  1.79889987e-02 -3.42905194e-02  2.28734984e-03
  9.36387569e-03  1.40142436e-02  3.85949362e-02  3.24582131e-02
 -2.52904671e-02  1.89033461e-02 -1.91176557e-02 -2.01085819e-02
  2.99520463e-02 -1.42714226e-04  3.59682872e-02 -1.12221049e-02
  6.94902469e-03 -1.77407801e-02  3.51860959e-02 -9.52348992e-04
  2.54667713e-02  4.53157631e-04  1.10958105e-02  1.21556527e-02
 -3.09169161e-02  8.65962278e-05  5.08587880e-03  2.12741930e-02
  2.45662985e-02  3.15080030e-02  1.64666567e-02  1.26157829e-02
  1.40590427e-02 -2.18461760e-02  9.97998290e-03  4.32906906e-02
  3.25492287e-02 -1.78163222e-02 -3.19818783e-02  1.28635715e-02
  3.57940083e-02  1.48223383e-02 -4.40434172e-02 -8.62850903e-03
 -6.11634319e-02 -3.47611928e-02 -5.95962361e-03  4.75466979e-02
  1.06741817e-02 -8.59623849e-02 -9.85937109e-03  1.21904645e-02
 -9.07914338e-02 -1.69424524e-01 -7.48490426e-02  7.82157690e-03
 -8.56573429e-02 -1.09931198e-01  8.54044616e-02  9.39250799e-02
  5.30814463e-02  1.95826530e-04  7.41417270e-02  3.35601892e-02
  3.57233458e-02 -1.97620009e-02  5.73640620e-02  1.77705712e-02
 -3.69742462e-02  2.85136889e-02  5.87964260e-02  1.47592299e-02
 -8.19758419e-02 -5.38824420e-02  6.87778499e-02 -4.74491840e-02
  8.75511029e-02  3.68559944e-02 -2.29024980e-03  6.53000405e-02
  2.26455740e-02  3.19565641e-02 -1.04731320e-03  2.84953804e-02
 -1.35009528e-02 -1.82574613e-02 -2.14744258e-02 -5.67911075e-02
 -7.17619883e-02  1.07723039e-01 -5.05871519e-03  8.53276395e-03
  4.35830622e-02  4.05894641e-02  3.13323145e-02 -7.49745271e-03
 -6.26507256e-02  3.95549821e-02  1.25975741e-02  1.97633697e-02
  2.42361537e-02 -1.97894099e-02  2.84671702e-03  7.06823046e-03
  5.04280965e-02  1.10606492e-02  3.59877235e-02  1.80857679e-02
  2.07628117e-02 -1.49993016e-02  4.94620798e-02 -3.12980190e-02
  2.97495232e-02  4.45957526e-02  2.80494716e-02  2.35627916e-02
 -3.55829057e-02  4.31170300e-02  6.93216561e-03  6.20078897e-03
  2.10302800e-02  1.90583317e-02 -1.50798306e-02  3.76552955e-02
 -1.55138372e-02  2.32014608e-02  2.45335888e-02 -4.18858788e-03
 -3.10408058e-02  5.48925110e-02 -1.73961069e-02  8.52081973e-03
  1.04913824e-02 -2.10116724e-02  1.97420297e-02 -6.25808182e-02
 -2.64428316e-02  8.29831178e-02  5.04869261e-04 -1.75301852e-02
 -5.07353220e-02 -4.12836043e-03  6.59911146e-02  2.69939848e-03
  1.53246024e-02  9.10921121e-04  3.56098011e-02 -1.21924314e-03
 -1.23445084e-03 -1.94256793e-02  3.41745222e-02  1.82007622e-02
  1.78843582e-02  3.71981603e-02 -3.89197188e-04  9.09120855e-03
  1.15762514e-02  4.81377509e-03  1.93488261e-02  5.46432811e-04
  1.93029785e-02  2.21881353e-02  3.72004301e-03  3.43425574e-02
  5.61787044e-02 -3.89473721e-02 -2.68844986e-02  2.71877513e-03
  2.16083420e-02  4.84243602e-03  3.60215562e-02  3.02632115e-02
  4.29428710e-03 -6.94033560e-02  1.78593568e-02 -9.17768946e-02
  7.86844017e-02  2.97288891e-02  1.79242562e-02  3.93134984e-02
 -6.57777312e-02  6.89301639e-02  1.31281521e-02 -1.80916273e-02
  1.71811778e-02  2.04317707e-02  3.40470374e-02  2.85898190e-02] [ 0.00671062 -0.00362888 -0.0182689
3  0.00495973 -0.02405441 -0.01548622
 -0.01027287 -0.00389231 -0.00488492 -0.01734241 -0.007992    0.0070488
  0.01499947 -0.02200277 -0.01242498  0.01190126 -0.03225806  0.01909778
 -0.02282847  0.00662538 -0.02078281  0.01839405  0.02570224 -0.01693086
 -0.0196354  -0.02178496 -0.08225517  0.03131113 -0.01783741 -0.08732612
 -0.080863    0.00967222  0.14416058  0.00199298  0.01711167 -0.05320814
 -0.01694149 -0.06051883 -0.02506851  0.00183717  0.0535952  -0.05521809
 -0.04601936  0.0347323  -0.07039558  0.05667081 -0.03227335 -0.06375826
 -0.01728571 -0.04211173  0.04396309  0.107142   -0.06451464 -0.03191104
  0.06273434 -0.006003    0.01056788 -0.01394422 -0.01767758  0.05347048
  0.04734098  0.07455732  0.0793575  -0.03575671  0.035416   -0.05513018
  0.05621806 -0.01653161  0.03444032 -0.05548946  0.00041897 -0.06124161
 -0.0080429   0.04594595 -0.02368579  0.00308707 -0.01934916  0.01793722
 -0.03964758 -0.02889982 -0.02172737 -0.05891024 -0.02000946  0.08638827
 -0.0009654   0.03232111  0.06962542 -0.00655239 -0.01011363  0.02976382
 -0.05737774  0.01830665  0.02382096  0.09174785 -0.00281528 -0.02137032
  0.00617981 -0.00777985  0.03962028  0.01111551  0.06792238  0.00955882
  0.11070648  0.06754098 -0.01812088 -0.08789444  0.08984911 -0.01919497
  0.02374182 -0.03039848  0.02036147 -0.00221679  0.01492013 -0.00562928
  0.01446693 -0.0303847  -0.0604418  -0.01123268  0.00895119 -0.0371892
 -0.02445131 -0.04976334  0.06574924 -0.00789096  0.02169197  0.06263326
 -0.00166553 -0.04703189 -0.02169998 -0.00858676  0.02598338 -0.02462299
 -0.01334238  0.03947368  0.02461322 -0.02848264 -0.00212052  0.02973453
 -0.03540683  0.01140413  0.00740014  0.04022327  0.0514464   0.02718208
 -0.03331308 -0.00998339  0.03773585 -0.00093994 -0.01192444 -0.03969465
 -0.02083386 -0.00675448 -0.03468101  0.06128802  0.00132758  0.00961273
 -0.07189705  0.02334516  0.01624667  0.03333333  0.01974984  0.00742363
 -0.03043846  0.00793126  0.01311475 -0.02394822 -0.00795756  0.01136364
  0.02941229  0.02760525 -0.00468355 -0.00408085  0.0497982  -0.03452649
 -0.01274826  0.00188875 -0.00691506  0.00284849  0.01988889  0.03868514
 -0.00298451  0.01614353 -0.03529412 -0.0030439   0.03271421  0.01776729
  0.02968177 -0.01243286  0.02002399 -0.00616663  0.02878458  0.01316875
  0.03060499  0.00393482 -0.01517512 -0.02551151 -0.00845594  0.02255666
 -0.05433435 -0.04778157 -0.00149104 -0.01107506 -0.01904909  0.03206899
 -0.01733369 -0.04073905 -0.01331321  0.0044983   0.01023345  0.02501246
  0.0219196   0.01117924  0.00628335  0.07840789  0.00081961 -0.02723622
  0.02432357  0.01021599  0.02788859  0.01862128  0.01383597 -0.00773009
 -0.04048802 -0.03245882 -0.02376075 -0.01575258  0.02007337  0.01027729
 -0.02993098  0.02357159  0.02587666  0.02771619 -0.03883495  0.00645567
 -0.01728906 -0.02439268  0.02035005  0.00085723  0.01765831  0.02797078
 -0.01414486  0.02788281  0.01208485  0.01141401  0.01994223  0.00128428
 -0.01875437  0.01178107 -0.00336232  0.01219836  0.01411191 -0.00151379
  0.02711117  0.04710786 -0.03910113  0.01911837 -0.01875972 -0.00662122
 -0.05183021 -0.0010328   0.00311828 -0.03505885 -0.00484952 -0.00459617
 -0.01033043 -0.00850525  0.00276488 -0.1797872  -0.03164461  0.03337969
  0.01749664  0.03504762 -0.01373042 -0.00065306  0.01329793  0.01311318
  0.01641928 -0.02485238  0.01974637  0.02062361  0.04101122 -0.01087683
  0.00773365 -0.01947365 -0.02737906 -0.00679989  0.00560073  0.02849186
 -0.00089612  0.02259664  0.04539239 -0.06089639 -0.038129    0.03214981
 -0.01482428 -0.03498605  0.04517048 -0.01460735  0.01729454  0.03217471
  0.04970353  0.01596865 -0.02619317 -0.02039078  0.01907599  0.01815607
  0.02117586 -0.01555038  0.01136142  0.03727057 -0.01003542 -0.09500914
 -0.01622173  0.07493525 -0.0348557  -0.00722105  0.00494529  0.02983152
  0.01124859  0.00973526 -0.05535433 -0.02507743  0.0221244   0.01463319
  0.0279677   0.0005655   0.01485615 -0.02789834  0.02329747  0.02026997
 -0.02285475 -0.00362784  0.02767912  0.03073776 -0.00659595 -0.02258023
  0.02527638 -0.0349947  -0.01730989  0.00027733  0.02292241 -0.01557158
 -0.00028421 -0.03136938 -0.02579083  0.0005837   0.00235225  0.01642714
  0.026265    0.01265242  0.00555403  0.01298208 -0.02535224 -0.01483185
  0.02498592 -0.0130128   0.00336794 -0.02881336 -0.00432258  0.01880016
  0.0122113   0.00280954  0.02489281  0.03083843 -0.00105904  0.0159025
 -0.01017691 -0.00777146 -0.00425023 -0.0258766   0.01232326  0.00338223
  0.01900759  0.00079333 -0.03939152  0.15054308  0.03695788 -0.0494815
 -0.0313065   0.04684915 -0.06449627 -0.06216462  0.0320515  -0.00541805
  0.02046196  0.02213583 -0.02318511  0.0151278  -0.05369949  0.02660873
 -0.00581856 -0.0133014   0.02426529  0.03421952  0.04250403 -0.00292969
  0.04358472  0.00140779 -0.02061856 -0.04425799 -0.02678387 -0.01286008
  0.0039087  -0.00025995  0.0184324   0.02752994 -0.02604851  0.05145186
  0.06516512  0.05731178 -0.02258584 -0.0118838   0.01982218  0.00873553
 -0.02684563 -0.01757543  0.02966522  0.07037605 -0.01643722  0.00459578
  0.00561414 -0.00971844 -0.0070996   0.00757098 -0.02671675 -0.01972978
  0.01050098  0.0036801   0.02933563 -0.05909472 -0.0325167  -0.02739374
 -0.01325443  0.02086792  0.0056391  -0.02967252 -0.0089096   0.04883421
 -0.00926569 -0.00771606  0.02309142  0.00506679 -0.00022878  0.01168883
  0.00543725  0.0067598   0.01029543  0.00708906  0.0241971   0.03672715
  0.0586282   0.03835616 -0.04937806 -0.02676416 -0.02974128  0.05878646
 -0.0188383   0.02061439 -0.01306931  0.00060225 -0.01022691  0.04072156
  0.01129063  0.01251172 -0.01330798  0.00828547  0.06210554 -0.00053947
 -0.01800179 -0.01411578 -0.02975084 -0.08317363  0.00710702 -0.00083022
 -0.00664728  0.0154747   0.00164745  0.01007434 -0.00305345  0.00632944
  0.00750627 -0.01610954  0.00040933  0.00327332 -0.00978793 -0.03418451
 -0.01513825  0.01666992 -0.02448859  0.00545697  0.00564481 -0.0174867
  0.00637187 -0.02205205 -0.0109403   0.01512451  0.04425135 -0.00553663
 -0.01456103  0.0306392  -0.00421674  0.04742747  0.00262753  0.01733871
  0.05212081 -0.01940131 -0.01267768 -0.0200386 ]
```

```python
In [295]: grader.check("q2d")
```

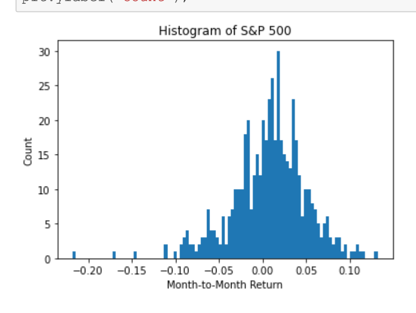
Out[295]: All tests passed!

2e) Plot the histogram of S&P 500 month-to-month returns. The y-axis is count, and the x-axis is month-to-month return. The number of the bars should be 100. Try to reproduce the plot below.

Specs of plot:

1. Color in default
2. plot title is 'Histogram of S&P 500'
3. x-label is 'Month-to-Month Return'
4. y-label is 'Count'
5. number of the bars should be 100.

```python
In [299]: plt.hist(x3, bins = 100)
plt.title("Histogram of S&P 500")
plt.xlabel("Month-to-Month Return")
plt.ylabel("Count");
```




```python
In [300]: grader.check("q2e")
```

Out[300]: All tests passed!

Submit

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output. **Please save before submitting!**

```python
In [301]: # Save your notebook first, then run this cell to create a pdf for your reference.
```

```python
In [ ]:
```