Jon M. Huntsman School of Business
Utah State University

HW2- Time series in Python

Let's take a look at time series with numpy and Pandas now. We'll start with a few simple tasks, then we'll move on to more complicated questions.

Important Avoid running the cells directly above the example output shown, otherwise you will end up overwriting it

1. Import Numpy package as np (0.5 point)

```
# CODE HERE
import numpy as np
```

2. Create the following numpy array using two functions: arange() and linspace(). You must generate the same answers. (2 points)

```
# Code here (using arange function)
array1 = np.arange(2,20,4)
array1
```

```
array([ 2,  6, 10, 14, 18])
```

```
# Do NOT write here!
```

```
array([ 2,  6, 10, 14, 18])
```

```
# Code here (using linspace function)
array2 = np.linspace(2,18,5)
array2
```

```
array([ 2.,  6., 10., 14., 18.])
```

```
# Do NOT write here!
```

```
array([ 2.,  6., 10., 14., 18.])
```

3. Create the following matrix of ones and save it as my_matrix. (1 point)

```
# Code here
my_matrix = np.ones((4,4))
my_matrix
```

```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

```
# Do NOT write here
```

```
array([[1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.],
       [1., 1., 1., 1.]])
```

4. Reshape "my_matrix" into a new matrix as shown below. (1 point)

```
# Code here
my_matrix.reshape((2,8))
```

```
array([[1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1.]])
```

```
# Do NOT write here
```

```
array([[1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1.]])
```

5. First, replace the first row of "my_matrix" with zeros. then replace the last column with 2s. Save the new matrix as "my_matrix2". (2 points)

```
# Code here
my_matrix[0] = np.zeros(4)
```

```
my_matrix[:, -1] = 2
my_matrix
```

```
array([[0., 0., 0., 2.],
       [1., 1., 1., 2.],
       [1., 1., 1., 2.],
       [1., 1., 1., 2.]])
```

```
# Do not write here
```

```
array([[0., 0., 0., 2.],
       [1., 1., 1., 2.],
       [1., 1., 1., 2.],
       [1., 1., 1., 2.]])
```

6. Create the following matrix and save it as "A". (2 points)

```
# Code here
A = np.arange(0.01, 1.01, 0.01).reshape((10, 10))
A
```

```
array([[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 ],
       [0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 ],
       [0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 ],
       [0.31, 0.32, 0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 ],
       [0.41, 0.42, 0.43, 0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5 ],
       [0.51, 0.52, 0.53, 0.54, 0.55, 0.56, 0.57, 0.58, 0.59, 0.6 ],
       [0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67, 0.68, 0.69, 0.7 ],
       [0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77, 0.78, 0.79, 0.8 ],
       [0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88, 0.89, 0.9 ],
       [0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1.  ]])
```

```
# Do NOT write here.
A
```

```
array([[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 ],
       [0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 ],
       [0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 ],
       [0.31, 0.32, 0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 ],
       [0.41, 0.42, 0.43, 0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5 ],
       [0.51, 0.52, 0.53, 0.54, 0.55, 0.56, 0.57, 0.58, 0.59, 0.6 ],
       [0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67, 0.68, 0.69, 0.7 ],
       [0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77, 0.78, 0.79, 0.8 ],
       [0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88, 0.89, 0.9 ],
       [0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1.  ]])
```

7. Create a 10 * 10 matrix using random values drawn from a standard normal distribution. Call this matrix "B". Set seed number = 100 (2 points)

```
# Code here
np.random.seed(100)
B = np.random.normal(0,1,100).reshape((10,10))
B
```

```
array([[-1.74976547,  0.3426804 ,  1.1530358 , -0.25243604,  0.98132079,
         0.51421884,  0.22117967, -1.07004333, -0.18949583,  0.25500144],
       [-0.45802699,  0.43516349, -0.58359505,  0.81684707,  0.67272081,
        -0.10441114, -0.53128038,  1.02973269, -0.43813562, -1.11831825],
       [ 1.61898166,  1.54160517, -0.25187914, -0.84243574,  0.18451869,
         0.9370822 ,  0.73100034,  1.36155613, -0.32623806,  0.05567601],
       [ 0.22239961, -1.443217  , -0.75635231,  0.81645401,  0.75044476,
        -0.45594693,  1.18962227, -1.69061683, -1.35639905, -1.23243451],
       [-0.54443916, -0.66817174,  0.00731456, -0.61293874,  1.29974807,
        -1.73309562, -0.9833101 ,  0.35750775, -1.6135785 ,  1.47071387],
       [-1.1880176 , -0.54974619, -0.94004616, -0.82793236,  0.10886347,
         0.50780959, -0.86222735,  1.24946974, -0.07961125, -0.88973148],
       [-0.88179839,  0.01863895,  0.23784462,  0.01354855, -1.6355294 ,
        -1.04420988,  0.61303888,  0.73620521,  1.02692144, -1.43219061],
       [-1.8411883 ,  0.36609323, -0.33177714, -0.68921798,  2.03460756,
        -0.55071441,  0.75045333, -1.30699234,  0.58057334, -1.10452309],
       [ 0.69012147,  0.68689007, -1.56668753,  0.90497412,  0.7788224 ,
         0.42823287,  0.10887199,  0.02828363, -0.57882582, -1.1994512 ],
       [-1.70595201,  0.36916396,  1.87657343, -0.37690335,  1.83193608,
         0.00301743, -0.07602347,  0.00395759, -0.18501411, -2.48715154]])
```

```
# Do NOT write here

B
```

```
array([[-1.74976547,  0.3426804 ,  1.1530358 , -0.25243604,  0.98132079,
         0.51421884,  0.22117967, -1.07004333, -0.18949583,  0.25500144],
       [-0.45802699,  0.43516349, -0.58359505,  0.81684707,  0.67272081,
        -0.10441114, -0.53128038,  1.02973269, -0.43813562, -1.11831825],
       [ 1.61898166,  1.54160517, -0.25187914, -0.84243574,  0.18451869,
         0.9370822 ,  0.73100034,  1.36155613, -0.32623806,  0.05567601],
       [ 0.22239961, -1.443217  , -0.75635231,  0.81645401,  0.75044476,
        -0.45594693,  1.18962227, -1.69061683, -1.35639905, -1.23243451],
       [-0.54443916, -0.66817174,  0.00731456, -0.61293874,  1.29974807,
        -1.73309562, -0.9833101 ,  0.35750775, -1.6135785 ,  1.47071387],
       [-1.1880176 , -0.54974619, -0.94004616, -0.82793236,  0.10886347,
         0.50780959, -0.86222735,  1.24946974, -0.07961125, -0.88973148],
       [-0.88179839,  0.01863895,  0.23784462,  0.01354855, -1.6355294 ,
        -1.04420988,  0.61303888,  0.73620521,  1.02692144, -1.43219061],
       [-1.8411883 ,  0.36609323, -0.33177714, -0.68921798,  2.03460756,
        -0.55071441,  0.75045333, -1.30699234,  0.58057334, -1.10452309],
       [ 0.69012147,  0.68689007, -1.56668753,  0.90497412,  0.7788224 ,
         0.42823287,  0.10887199,  0.02828363, -0.57882582, -1.1994512 ],
       [-1.70595201,  0.36916396,  1.87657343, -0.37690335,  1.83193608,
         0.00301743, -0.07602347,  0.00395759, -0.18501411, -2.48715154]])
```

8. Multiply matrix A and B. (use the Matrix multiplication, NOT the element-wised multiplication) (1 point)

```
# Code here
np.dot(A,B)
```

```
array([[-0.38520099,  0.06358488, -0.05722663, -0.06955691,  0.43190936,
        -0.12156679,  0.06534707,  0.02587607, -0.11242174, -0.5925726 ],
       [-0.96896951,  0.17349491, -0.17278352, -0.17456096,  1.13265468,
        -0.2713685 ,  0.18147958,  0.09578209, -0.42840209, -1.36081353],
       [-1.55273803,  0.28340494, -0.28834041, -0.279565  ,  1.8334    ,
        -0.4211702 ,  0.2976121 ,  0.16568812, -0.74438244, -2.12905447],
       [-2.13650655,  0.39331498, -0.40389731, -0.38456905,  2.53414533,
        -0.57097191,  0.41374462,  0.23559414, -1.06036278, -2.8972954 ],
       [-2.72027506,  0.50322501, -0.5194542 , -0.48957309,  3.23489065,
        -0.72077361,  0.52987714,  0.30550017, -1.37634313, -3.66553634],
       [-3.30404358,  0.61313505, -0.63501109, -0.59457714,  3.93563597,
        -0.87057532,  0.64600966,  0.37540619, -1.69232348, -4.43377727],
       [-3.8878121 ,  0.72304508, -0.75056798, -0.69958118,  4.6363813 ,
        -1.02037702,  0.76214218,  0.44531222, -2.00830382, -5.20201821],
       [-4.47158062,  0.83295511, -0.86612487, -0.80458523,  5.33712662,
        -1.17017873,  0.8782747 ,  0.51521824, -2.32428417, -5.97025915],
       [-5.05534913,  0.94286515, -0.98168176, -0.90958927,  6.03787194,
        -1.31998043,  0.99440722,  0.58512427, -2.64026452, -6.73850008],
       [-5.63911765,  1.05277518, -1.09723865, -1.01459332,  6.73861727,
        -1.46978214,  1.11053974,  0.65503029, -2.95624487, -7.50674102]])
```

```
# Do NOT write here
```

```
array([[-0.38520099,  0.06358488, -0.05722663, -0.06955691,  0.43190936,
        -0.12156679,  0.06534707,  0.02587607, -0.11242174, -0.5925726 ],
       [-0.96896951,  0.17349491, -0.17278352, -0.17456096,  1.13265468,
        -0.2713685 ,  0.18147958,  0.09578209, -0.42840209, -1.36081353],
       [-1.55273803,  0.28340494, -0.28834041, -0.279565  ,  1.8334    ,
        -0.4211702 ,  0.2976121 ,  0.16568812, -0.74438244, -2.12905447],
       [-2.13650655,  0.39331498, -0.40389731, -0.38456905,  2.53414533,
        -0.57097191,  0.41374462,  0.23559414, -1.06036278, -2.8972954 ],
       [-2.72027506,  0.50322501, -0.5194542 , -0.48957309,  3.23489065,
        -0.72077361,  0.52987714,  0.30550017, -1.37634313, -3.66553634],
       [-3.30404358,  0.61313505, -0.63501109, -0.59457714,  3.93563597,
        -0.87057532,  0.64600966,  0.37540619, -1.69232348, -4.43377727],
       [-3.8878121 ,  0.72304508, -0.75056798, -0.69958118,  4.6363813 ,
        -1.02037702,  0.76214218,  0.44531222, -2.00830382, -5.20201821],
       [-4.47158062,  0.83295511, -0.86612487, -0.80458523,  5.33712662,
        -1.17017873,  0.8782747 ,  0.51521824, -2.32428417, -5.97025915],
       [-5.05534913,  0.94286515, -0.98168176, -0.90958927,  6.03787194,
        -1.31998043,  0.99440722,  0.58512427, -2.64026452, -6.73850008],
       [-5.63911765,  1.05277518, -1.09723865, -1.01459332,  6.73861727,
        -1.46978214,  1.11053974,  0.65503029, -2.95624487, -7.50674102]])
```

9. Import Pandas package as pd (0.5 point)

```
# Code here
import pandas as pd
```

10. Create a data frame with three columns A, B, C. Name your data frame as df. Again use the seed number = 100

- Column A: 20 random numbers from the uniform distribution (2 point)
- Column B: the first 20 Capitals of english letters (3 points)
- Column C: range of numbers between 20 and 39 (1 point)

```
# Code here
np.random.seed(100)
df = pd.DataFrame({"A": np.random.uniform(0,1,20), "B": np.array([chr(i) for i in range(ord(
df
```

	A	B	C
0	0.543405	A	20
1	0.278369	B	21
2	0.424518	C	22
3	0.844776	D	23
4	0.004719	E	24
5	0.121569	F	25
6	0.670749	G	26
7	0.825853	H	27
8	0.136707	I	28
9	0.575093	J	29
10	0.891322	K	30
11	0.209202	L	31
12	0.185328	M	32
13	0.108377	N	33
14	0.219697	O	34
15	0.978624	P	35
16	0.811683	Q	36
17	0.171941	R	37
18	0.816225	S	38
19	0.274074	T	39

```
# Do Not write here

df
```

	A	B	C
0	0.543405	A	20
1	0.278369	B	21
2	0.424518	C	22
3	0.844776	D	23
4	0.004719	E	24
5	0.121569	F	25
6	0.670749	G	26
7	0.825853	H	27
8	0.136707	I	28
9	0.575093	J	29
10	0.891322	K	30
11	0.209202	L	31
12	0.185328	M	32
13	0.108377	N	33
14	0.219697	O	34
15	0.978624	P	35
16	0.811683	Q	36
17	0.171941	R	37
18	0.816225	S	38
19	0.274074	T	39

11. Create a new column in df and name is as "default". If A>0.5 then default =1 otherwise 0. (2 points)

```
# Code here
df["default"] = np.where(df["A"] > 0.5, 1, 0)
df
```

	A	B	C	default
0	0.543405	A	20	1
1	0.278369	B	21	0
2	0.424518	C	22	0
3	0.844776	D	23	1
4	0.004719	E	24	0
5	0.121569	F	25	0
6	0.670749	G	26	1
7	0.825853	H	27	1
8	0.136707	I	28	0
9	0.575093	J	29	1
10	0.891322	K	30	1
11	0.209202	L	31	0
12	0.185328	M	32	0
13	0.108377	N	33	0
14	0.219697	O	34	0
15	0.978624	P	35	1
16	0.811683	Q	36	1
17	0.171941	R	37	0
18	0.816225	S	38	1
19	0.274074	T	39	0

```
# Do Not write here
df
```

	A	B	C	default
0	0.543405	A	20	1
1	0.278369	B	21	0
2	0.424518	C	22	0
3	0.844776	D	23	1
4	0.004719	E	24	0
5	0.121569	F	25	0
6	0.670749	G	26	1
7	0.825853	H	27	1
8	0.136707	I	28	0
9	0.575093	J	29	1
10	0.891322	K	30	1
11	0.209202	L	31	0
12	0.185328	M	32	0
13	0.108377	N	33	0
14	0.219697	O	34	0
15	0.978624	P	35	1
16	0.811683	Q	36	1
17	0.171941	R	37	0
18	0.816225	S	38	1
19	0.274074	T	39	0

12. Install the yfinance package on your local computer (or Google Colab) (0.5 point)

```
# Code here
!pip install yfinance
```

```
Collecting yfinance
  Downloading yfinance-0.2.43-py2.py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: pandas>=1.3.0 in c:\users\aggie\appdata\local\programs\py
Requirement already satisfied: numpy>=1.16.5 in c:\users\aggie\appdata\local\programs\py
Collecting requests>=2.31 (from yfinance)
  Downloading requests-2.32.3-py3-none-any.whl.metadata (4.6 kB)
Collecting multitasking>=0.0.7 (from yfinance)
  Downloading multitasking-0.0.11-py3-none-any.whl.metadata (5.5 kB)
Collecting lxml>=4.9.1 (from yfinance)
```

```
  Downloading lxml-5.3.0-cp312-cp312-win_amd64.whl.metadata (3.9 kB)
Requirement already satisfied: platformdirs>=2.0.0 in c:\users\aggie\appdata\roaming\pyt
Requirement already satisfied: pytz>=2022.5 in c:\users\aggie\appdata\local\programs\pyt
Collecting frozendict>=2.3.4 (from yfinance)
  Downloading frozendict-2.4.4-py312-none-any.whl.metadata (23 kB)
Collecting peewee>=3.16.2 (from yfinance)
  Downloading peewee-3.17.6.tar.gz (3.0 MB)
     ---------------------------------------- 0.0/3.0 MB ? eta -:--:--
     ---------------------------------------- 3.0/3.0 MB 34.5 MB/s eta 0:00:00
  Installing build dependencies: started
  Installing build dependencies: finished with status 'done'
  Getting requirements to build wheel: started
  Getting requirements to build wheel: finished with status 'done'
  Preparing metadata (pyproject.toml): started
  Preparing metadata (pyproject.toml): finished with status 'done'
Collecting beautifulsoup4>=4.11.1 (from yfinance)
  Downloading beautifulsoup4-4.12.3-py3-none-any.whl.metadata (3.8 kB)
Collecting html5lib>=1.1 (from yfinance)
  Downloading html5lib-1.1-py2.py3-none-any.whl.metadata (16 kB)
Collecting soupsieve>1.2 (from beautifulsoup4>=4.11.1->yfinance)
  Downloading soupsieve-2.6-py3-none-any.whl.metadata (4.6 kB)
Requirement already satisfied: six>=1.9 in c:\users\aggie\appdata\roaming\python\python3
Collecting webencodings (from html5lib>=1.1->yfinance)
  Downloading webencodings-0.5.1-py2.py3-none-any.whl.metadata (2.1 kB)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\aggie\appdata\roaming\
Requirement already satisfied: tzdata>=2022.7 in c:\users\aggie\appdata\local\programs\p
Collecting charset-normalizer<4,>=2 (from requests>=2.31->yfinance)
  Downloading charset_normalizer-3.3.2-cp312-cp312-win_amd64.whl.metadata (34 kB)
Collecting idna<4,>=2.5 (from requests>=2.31->yfinance)
  Downloading idna-3.10-py3-none-any.whl.metadata (10 kB)
Collecting urllib3<3,>=1.21.1 (from requests>=2.31->yfinance)
  Downloading urllib3-2.2.3-py3-none-any.whl.metadata (6.5 kB)
Collecting certifi>=2017.4.17 (from requests>=2.31->yfinance)
  Downloading certifi-2024.8.30-py3-none-any.whl.metadata (2.2 kB)
Downloading yfinance-0.2.43-py2.py3-none-any.whl (84 kB)
Downloading beautifulsoup4-4.12.3-py3-none-any.whl (147 kB)
Downloading frozendict-2.4.4-py312-none-any.whl (16 kB)
Downloading html5lib-1.1-py2.py3-none-any.whl (112 kB)
Downloading lxml-5.3.0-cp312-cp312-win_amd64.whl (3.8 MB)
   ---------------------------------------- 0.0/3.8 MB ? eta -:--:--
   ---------------------------------------- 3.8/3.8 MB 37.8 MB/s eta 0:00:00
Downloading multitasking-0.0.11-py3-none-any.whl (8.5 kB)
Downloading requests-2.32.3-py3-none-any.whl (64 kB)
Downloading certifi-2024.8.30-py3-none-any.whl (167 kB)
Downloading charset_normalizer-3.3.2-cp312-cp312-win_amd64.whl (100 kB)
Downloading idna-3.10-py3-none-any.whl (70 kB)
Downloading soupsieve-2.6-py3-none-any.whl (36 kB)
Downloading urllib3-2.2.3-py3-none-any.whl (126 kB)
Downloading webencodings-0.5.1-py2.py3-none-any.whl (11 kB)
```

13. Import yfinance as yf (0.5 point)

```
# Code here
import yfinance as yf
```

14. Make a data frame "df" with the following 2 columns (3 point):

- Column "GOOG": Google adjusted close price since 1/1/2021 to 1/1/2023
- Column "MSFT": Microsoft adjusted close price since 1/1/2021 to 1/1/2023

```
# Code here
df = pd.DataFrame({"GOOG": yf.download("GOOG", start="2021-01-01", end="2023-01-01")['Adj Cl
df.tail()
```

```
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
                 GOOG        MSFT
      Date
2022-12-23  89.589981  235.345566
2022-12-27  87.714592  233.600662
2022-12-28  86.248192  231.205109
2022-12-29  88.732086  237.593216
2022-12-30  88.512634  236.420135
```

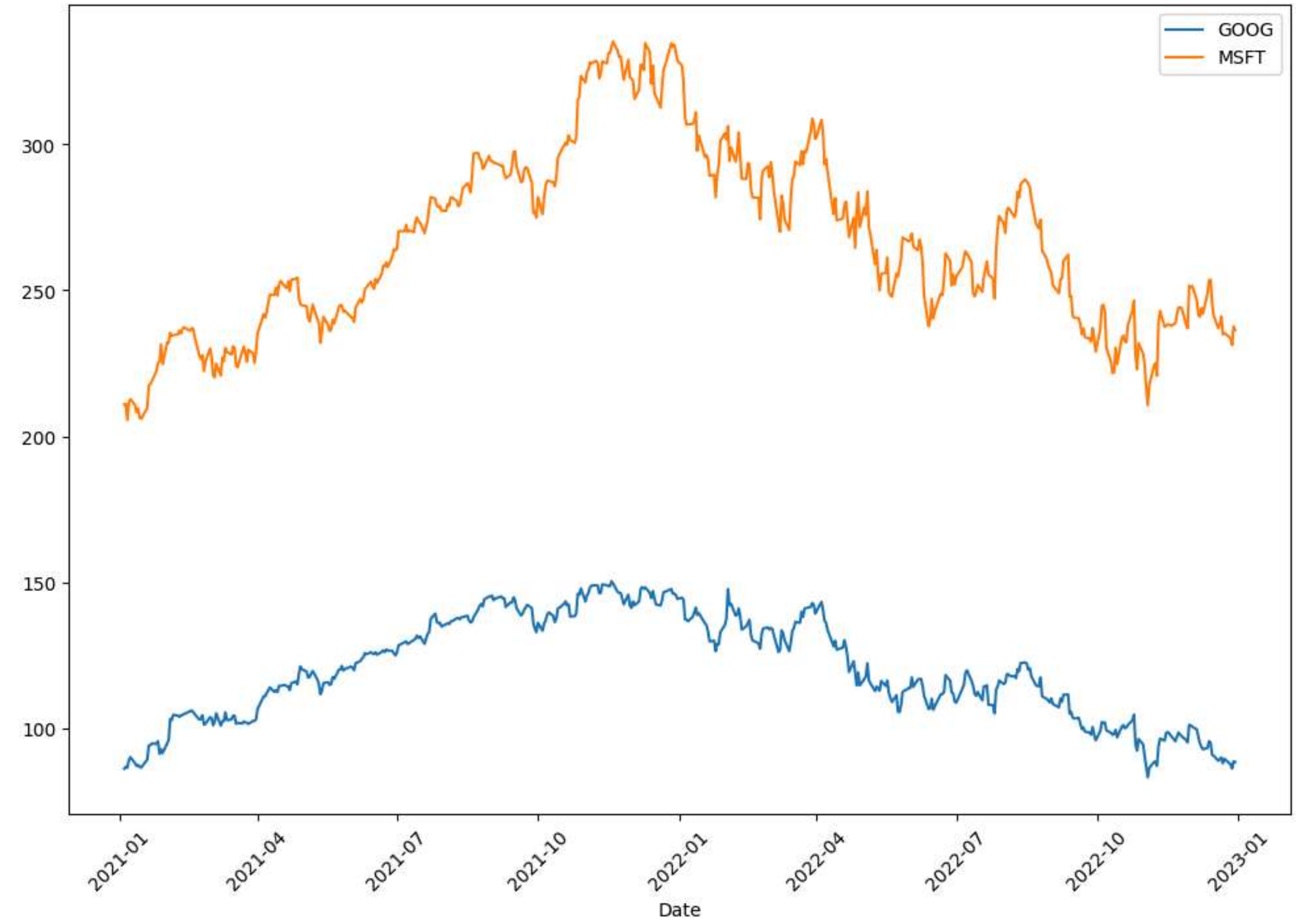
```
df.tail()
```

```
[*********************100%***********************]  2 of 2 completed
                 GOOG        MSFT
      Date
2022-12-23  89.809998  238.729996
2022-12-27  87.930000  236.960007
2022-12-28  86.459999  234.529999
2022-12-29  88.949997  241.009995
2022-12-30  88.730003  239.820007
```

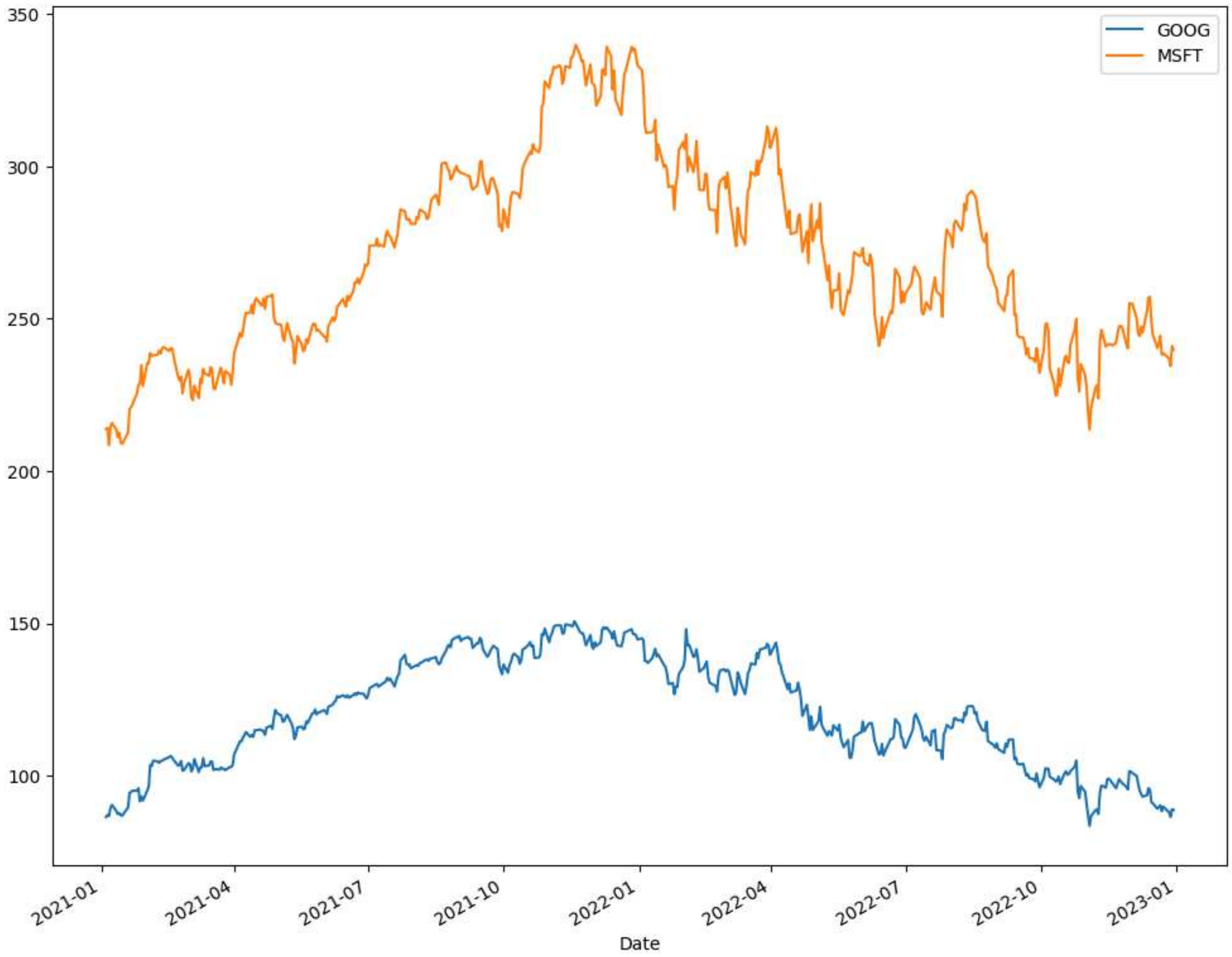
15. Replicate the following chart! (you need to import matplotlib.pyplot). (1 points)

```
# Code here
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(12, 8))
plt.plot(df.index, df["GOOG"], label="GOOG")
plt.plot(df.index, df["MSFT"], label="MSFT")
plt.xlabel("Date")
plt.xticks(rotation=45)
plt.legend()
plt.show()
```



Start coding or generate with AI.

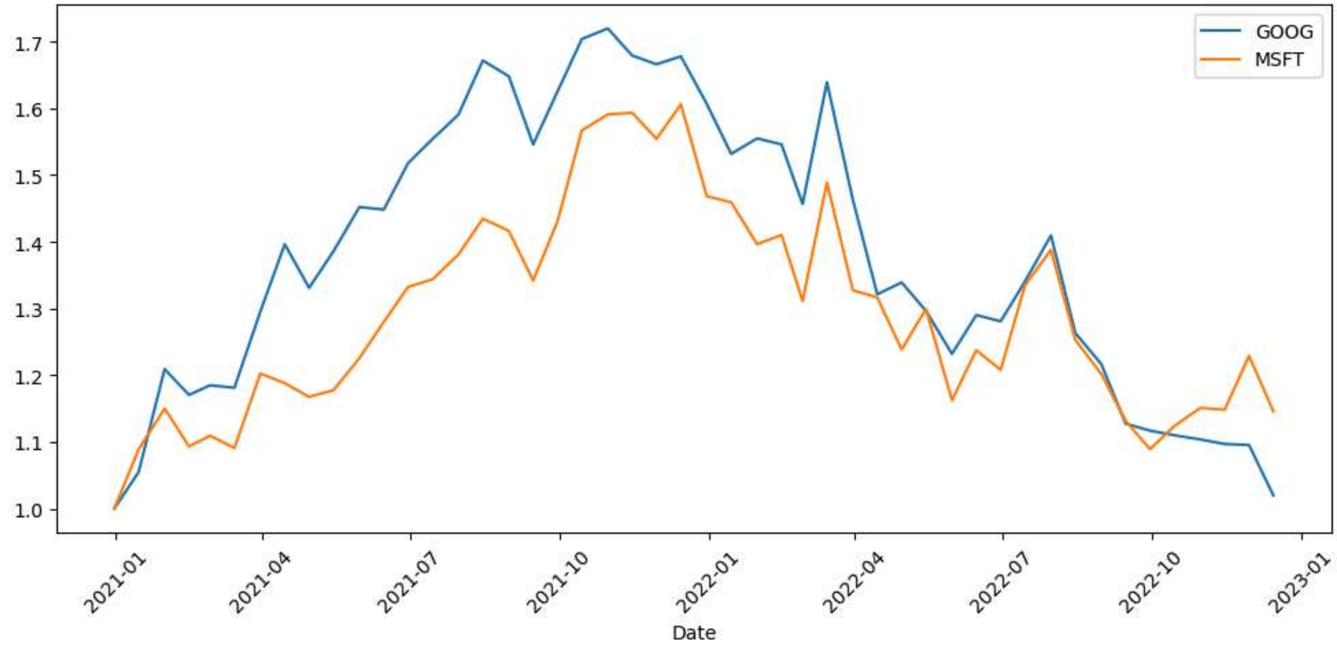


16. Replicate the following normalized chart! (2 points)

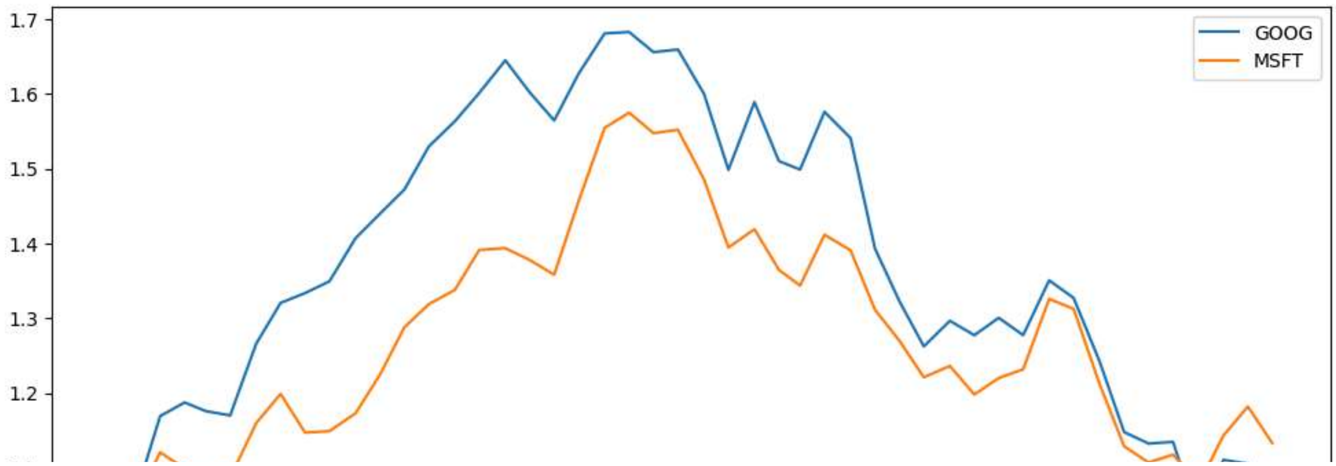
- Data is resamples by "Semi-month end frequency". Find the correct rule here: https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#offset-aliases

```
# Code here
resample = df.resample("SME").last()
normalized = resample/resample.min()
```

```
plt.figure(figsize=(12, 5))
plt.plot(normalized.index, normalized["GOOG"], label="GOOG")
plt.plot(normalized.index, normalized["MSFT"], label="MSFT")
plt.xlabel("Date")
plt.xticks(rotation=45)
plt.legend()
plt.show()
```



Start coding or generate with AI.



17. Along with the Microsoft closing prices, plot the 10-day and 60-day simple rolling moving averages. (3 points)

```
# Code here
plt.figure(figsize=(15, 5))
plt.plot(df.index, df["MSFT"], label="MSFT")
plt.plot(df.index, df["MSFT"].rolling(10).mean(), label="MSFT_MA10")
plt.plot(df.index, df["MSFT"].rolling(60).mean(), label="MSFT_MA60")
plt.xlabel("Date")
plt.xticks(rotation=45)
plt.legend()
plt.show()
```