Assignment 2

Write your Name and email below

David Shen ds6870@nyu.edu

Exercise 1

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

The dataset has address

`url='https://github.com/amoreira2/Fin418/blob/main/assets/data/Assignment1.xlsx?raw=true'`

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

This file contains multiple sheets, you should use `read_excel` to get the data that contains the 49 value-weighted industry portfolios.

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

Do the followings:

1. Import this dataframe as `df_ind`
   - Use "sheet_name" to select the desired excel sheet.
   - Use "skip_rows" to skip the initial rows before the data. you want the header, i.e. , the column names to be included! This will be a integer, i.e. just a number like 5
   - Figure out what is the code for missing value and change the option `na_values` appropriately. It will be in string format like that 'number'
   - If you look at the excel file you will see that there are other data sets stacked horizontally. Use the `usecols` option to select the range of columns you want imported
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. Call `df_ind.info()` so you check all the tasks were accomplished.
6. In the next cell, call `df_ind.head()`

```
# 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 = 'Complete the URL to the data file here'
url='https://github.com/amoreira2/Fin418/blob/main/assets/data/Assignment1.xlsx?raw=true'

# Import the data
df_ind = pd.read_excel(
    url,
    sheet_name='49_Industry_Portfolios',
    skiprows=6,
    na_values=['-99.99', '-999'],
    usecols="A:AX"
)


# 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')

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

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

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1069 entries, 1926-07-01 to 2015-07-01
Data columns (total 49 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Agric   1069 non-null   float64
 1   Food    1069 non-null   float64
 2   Soda    625 non-null    float64
 3   Beer    1069 non-null   float64
 4   Smoke   1069 non-null   float64
 5   Toys    1069 non-null   float64
 6   Fun     1069 non-null   float64
 7   Books   1069 non-null   float64
 8   Hshld   1069 non-null   float64
 9   Clths   1069 non-null   float64
 10  Hlth    553 non-null    float64
 11  MedEq   1069 non-null   float64
 12  Drugs   1069 non-null   float64
 13  Chems   1069 non-null   float64
 14  Rubbr   1009 non-null   float64
 15  Txtls   1069 non-null   float64
 16  BldMt   1069 non-null   float64
 17  Cnstr   1069 non-null   float64
 18  Steel   1069 non-null   float64
 19  FabPr   625 non-null    float64
 20  Mach    1069 non-null   float64
 21  ElcEq   1069 non-null   float64
 22  Autos   1069 non-null   float64
 23  Aero    1069 non-null   float64
 24  Ships   1069 non-null   float64
 25  Guns    625 non-null    float64
 26  Gold    625 non-null    float64
 27  Mines   1069 non-null   float64
 28  Coal    1069 non-null   float64
 29  Oil     1069 non-null   float64
 30  Util    1069 non-null   float64
 31  Telcm   1069 non-null   float64
 32  PerSv   1057 non-null   float64
 33  BusSv   1069 non-null   float64
 34  Hardw   1069 non-null   float64
 35  Softw   601 non-null    float64
 36  Chips   1069 non-null   float64
 37  LabEq   1069 non-null   float64
 38  Paper   1024 non-null   float64
 39  Boxes   1069 non-null   float64
 40  Trans   1069 non-null   float64
 41  Whlsl   1069 non-null   float64
 42  Rtail   1069 non-null   float64
 43  Meals   1069 non-null   float64
 44  Banks   1069 non-null   float64
 45  Insur   1069 non-null   float64
 46  RlEst   1069 non-null   float64
 47  Fin     1069 non-null   float64
 48  Other   1069 non-null   float64
dtypes: float64(49)
memory usage: 417.6 KB
```

	Agric	Food	Soda	Beer	Smoke	Toys	Fun	Books	Hshld	Clths	...	Boxes	Trans	Whlsl	Rtail	Meals	Banks	Insur	RlE
date																			
1926-07-01	2.37	0.12	NaN	-5.19	1.29	8.65	2.50	50.21	-0.48	8.08	...	7.70	1.94	-23.79	0.07	1.87	4.61	-0.54	2.
1926-08-01	2.23	2.68	NaN	27.03	6.50	16.81	-0.76	42.98	-3.58	-2.51	...	-2.38	4.88	5.39	-0.75	-0.13	11.83	2.57	5.
1926-09-01	-0.57	1.58	NaN	4.02	1.26	8.33	6.42	-4.91	0.73	-0.51	...	-5.54	0.06	-7.87	0.25	-0.56	-1.75	0.72	-3.
1926-10-01	-0.46	-3.68	NaN	-3.31	1.06	-1.40	-5.09	5.37	-4.68	0.12	...	-5.08	-2.64	-15.38	-2.20	-4.11	-11.82	-4.28	-5.

```
df_ind
```

	Agric	Food	Soda	Beer	Smoke	Toys	Fun	Books	Hshld	Clths	...	Boxes	Trans	Whlsl	Rtail	Meals	Banks	Insur	RlE
date																			
1926-07-01	2.37	0.12	NaN	-5.19	1.29	8.65	2.50	50.21	-0.48	8.08	...	7.70	1.94	-23.79	0.07	1.87	4.61	-0.54	2.
1926-08-01	2.23	2.68	NaN	27.03	6.50	16.81	-0.76	42.98	-3.58	-2.51	...	-2.38	4.88	5.39	-0.75	-0.13	11.83	2.57	5.
1926-09-01	-0.57	1.58	NaN	4.02	1.26	8.33	6.42	-4.91	0.73	-0.51	...	-5.54	0.06	-7.87	0.25	-0.56	-1.75	0.72	-3.
1926-10-01	-0.46	-3.68	NaN	-3.31	1.06	-1.40	-5.09	5.37	-4.68	0.12	...	-5.08	-2.64	-15.38	-2.20	-4.11	-11.82	-4.28	-5.
1926-11-01	6.75	6.26	NaN	7.29	4.55	0.00	1.82	-6.40	-0.54	1.87	...	3.84	1.60	4.67	6.52	4.33	-2.97	3.58	2.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2015-03-01	-5.28	2.47	-4.64	-2.07	-8.82	-4.11	-2.35	0.48	-1.98	1.23	...	-3.25	-3.62	0.61	0.99	-0.29	-0.83	2.28	3.
2015-04-01	1.07	-0.23	-0.43	-0.52	5.94	9.25	2.62	-4.07	-2.41	-1.53	...	-1.72	-1.14	-1.20	-2.88	0.51	2.13	-1.78	-2.

Exercise 2. Advanced date manipulation

1. convert the date from the start of the month to end of the month.
2. call `df_ind.head()` and verify it works

Hint:

- Read this link: https://stackoverflow.com/questions/37354105/find-the-end-of-the-month-of-a-pandas-dataframe-series. If you google "pandas end of month" that is the first thing that comes out. Read the answer and apply to your problem.
- you aready set date as index, so you cannot do stuff like `df_ind.date` or `df_ind['date']` and have to adjust the code accordingly. Think about how to access the index.

```
# your code below
df_ind.index += MonthEnd(0)
df_ind.head()
```

	Agric	Food	Soda	Beer	Smoke	Toys	Fun	Books	Hshld	Clths	...	Boxes	Trans	Whlsl	Rtail	Meals	Banks	Insur	RlE
date																			
1926-07-31	2.37	0.12	NaN	-5.19	1.29	8.65	2.50	50.21	-0.48	8.08	...	7.70	1.94	-23.79	0.07	1.87	4.61	-0.54	2.
1926-08-31	2.23	2.68	NaN	27.03	6.50	16.81	-0.76	42.98	-3.58	-2.51	...	-2.38	4.88	5.39	-0.75	-0.13	11.83	2.57	5.
1926-09-30	-0.57	1.58	NaN	4.02	1.26	8.33	6.42	-4.91	0.73	-0.51	...	-5.54	0.06	-7.87	0.25	-0.56	-1.75	0.72	-3.
1926-10-31	-0.46	-3.68	NaN	-3.31	1.06	-1.40	-5.09	5.37	-4.68	0.12	...	-5.08	-2.64	-15.38	-2.20	-4.11	-11.82	-4.28	-5.

Exercise 3. Importing risk-free rate

1. In this same file there is another sheet with market returns and the risk-free rate. Import them as `df_rmrf` by following all the steps you did in the above two questions
2. Call `df_rmrf.info()` so you check all the tasks were accomplished.
3. In the next cell, call `df_rmrf.head()`

```
# your code below

# Import the data
df_rmrf = pd.read_excel(
    url,
    sheet_name='Market_proxy',
    skiprows=5,
    na_values=['-99.99', '-999'],
    usecols="A:C"
)

# rename dates
df_rmrf.rename(columns={df_rmrf.columns[0]: 'date'}, inplace=True)

# convert to datetime

df_rmrf['date'] = pd.to_datetime(df_rmrf['date'], format='%Y%m') + MonthEnd(0)

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

df_rmrf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1073 entries, 1926-07-31 to 2015-11-30
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Mkt-RF  1073 non-null   float64
 1   RF      1073 non-null   float64
dtypes: float64(2)
memory usage: 25.1 KB
```

```
df_rmrf.head()
```

	Mkt-RF	RF
date		
1926-07-31	2.96	0.22
1926-08-31	2.64	0.25
1926-09-30	0.36	0.23
1926-10-31	-3.24	0.32
1926-11-30	2.53	0.31

Next steps: Generate code with df_rmrf New interactive sheet

Exercise 4. Constructing excess returns A

1. for the industry `Agric`, construct the excess return by subtracting the risk-free rate RF from it.
2. compute the mean of this excess return.
3. print it along with the mean of the raw returns and the risk free rate to compare

```
# your code below
agric_excess_ret = df_ind['Agric'] - df_rmrf['RF']


print(f"Mean excess return for agric {agric_excess_ret.mean()}")
print(f"Mean of raw returns {df_rmrf['Mkt-RF'].mean()}")
print(f"Mean of risk free rate {df_rmrf['RF'].mean()}")
```

```
Mean excess return for agric 0.681889616463985
Mean of raw returns 0.6500745573159367
Mean of risk free rate 0.2809878844361603
```

```
print(df_ind.index[:5])
print(df_rmrf.index[:5])
print(agric_excess_ret.head())
print(agric_excess_ret.info())
```

```
DatetimeIndex(['1926-07-31', '1926-08-31', '1926-09-30', '1926-10-31',
               '1926-11-30'],
              dtype='datetime64[ns]', name='date', freq=None)
DatetimeIndex(['1926-07-31', '1926-08-31', '1926-09-30', '1926-10-31',
               '1926-11-30'],
              dtype='datetime64[ns]', name='date', freq=None)
date
1926-07-31    2.15
1926-08-31    1.98
```

```
1926-09-30   -0.80
1926-10-31   -0.78
1926-11-30    6.44
dtype: float64
<class 'pandas.core.series.Series'>
DatetimeIndex: 1073 entries, 1926-07-31 to 2015-11-30
Series name: None
Non-Null Count  Dtype
--------------  -----
1069 non-null   float64
dtypes: float64(1)
memory usage: 16.8 KB
None
```

Exercise 5. Constructing excess returns B

1. construct excess returns for all portfolio by subtracting the risk-free rate from all of columns at the same time
2. name the new data frame `df_inde` (for excess returns)

Hint:

- You can do that using the method `.subtract()` with the option axis to tell along which dimension
- Go ahead , google "pandas subtract" to see how this works

```
# your code below

df_inde = df_ind.sub(df_rmrf['RF'], axis=0)

df_inde.head()
```

	Agric	Food	Soda	Beer	Smoke	Toys	Fun	Books	Hshld	Clths	...	Boxes	Trans	Whlsl	Rtail	Meals	Banks	Insur	RlE
date																			
1926-07-31	2.15	-0.10	NaN	-5.41	1.07	8.43	2.28	49.99	-0.70	7.86	...	7.48	1.72	-24.01	-0.15	1.65	4.39	-0.76	2.
1926-08-31	1.98	2.43	NaN	26.78	6.25	16.56	-1.01	42.73	-3.83	-2.76	...	-2.63	4.63	5.14	-1.00	-0.38	11.58	2.32	5.
1926-09-30	-0.80	1.35	NaN	3.79	1.03	8.10	6.19	-5.14	0.50	-0.74	...	-5.77	-0.17	-8.10	0.02	-0.79	-1.98	0.49	-3.
1926-10-31	-0.78	-4.00	NaN	-3.63	0.74	-1.72	-5.41	5.05	-5.00	-0.20	...	-5.40	-2.96	-15.70	-2.52	-4.43	-12.14	-4.60	-6.

Exercise 6. Drop missing observations

You may notice that excess returns of some industries are not available at the beginning of the sample.

If we want all the industries to have same period of data in `df_inde`, we need to drop some observations.

Do the followings:

1. Use method `dropna` to drop rows in `df_inde` if **ANY** industry is missing.
2. After that, `print(df_inde.shape)` to see the changes in the length.

Hint

- when you call `dropna` function, use `axis` and `how` option to drop missing values if **ANY** industry is missing

```
# your code below
print(df_inde.shape)
df_inde = df_inde.join(df_rmrf, how = "inner")
df_inde = df_inde.dropna(how='any', axis = 0)
df_inde.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 553 entries, 1969-07-31 to 2015-07-31
Data columns (total 51 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Agric   553 non-null    float64
 1   Food    553 non-null    float64
 2   Soda    553 non-null    float64
 3   Beer    553 non-null    float64
 4   Smoke   553 non-null    float64


 6   Fun     553 non-null    float64
 6  Fun     553 non-null   float64
 7  Books   553 non-null   float64
 8  Hshld   553 non-null   float64
 9  Clths   553 non-null   float64
 10 Hlth    553 non-null   float64
 11 MedEq   553 non-null   float64
 12 Drugs   553 non-null   float64
 13 Chems   553 non-null   float64
 14 Rubbr   553 non-null   float64
 15 Txtls   553 non-null   float64
 16 BldMt   553 non-null   float64
 17 Cnstr   553 non-null   float64
 18 Steel   553 non-null   float64
 19 FabPr   553 non-null   float64
 20 Mach    553 non-null   float64
 21 ElcEq   553 non-null   float64
 22 Autos   553 non-null   float64
 23 Aero    553 non-null   float64
 24 Ships   553 non-null   float64
 25 Guns    553 non-null   float64
 26 Gold    553 non-null   float64
 27 Mines   553 non-null   float64
 28 Coal    553 non-null   float64
 29 Oil     553 non-null   float64
 30 Util    553 non-null   float64
 31 Telcm   553 non-null   float64
 32 PerSv   553 non-null   float64
 33 BusSv   553 non-null   float64
 34 Hardw   553 non-null   float64
 35 Softw   553 non-null   float64
 36 Chips   553 non-null   float64
 37 LabEq   553 non-null   float64
 38 Paper   553 non-null   float64
 39 Boxes   553 non-null   float64
 40 Trans   553 non-null   float64
 41 Whlsl   553 non-null   float64
 42 Rtail   553 non-null   float64
 43 Meals   553 non-null   float64
 44 Banks   553 non-null   float64
 45 Insur   553 non-null   float64
 46 RlEst   553 non-null   float64
 47 Fin     553 non-null   float64
 48 Other   553 non-null   float64
 49 Mkt-RF  553 non-null   float64
 50 RF      553 non-null   float64
dtypes: float64(51)
memory usage: 224.7 KB
```

Exercise 7. Moments

We will now estimate the risk-premium in each of these portfolio and the covariance between these portfolios.

Do the followings:

1. using the method `mean` on the excess return data frame to obtain a vector of average excess returns.
2. Using `std` construct an estimator for each asset standard deviation.
3. use `cov` method to estimate the covariance of excess returns.
4. Discuss in each units these variables are

```
# your code below
ERe = df_inde.mean()
std_rets = df_inde.std()
CovRe = df_inde.cov()
```

```
ERe.head()
```

	0
Agric	0.596926
Food	0.737450
Soda	0.747577
Beer	0.745371
Smoke	1.073165

dtype: float64

```
CovRe.head()
```

	Agric	Food	Soda	Beer	Smoke	Toys	Fun	Books	Hshld	Clths	...	Whlsl
Agric	42.283870	13.895412	14.346336	15.357792	13.991825	22.845450	25.603373	20.651770	14.086705	22.358837	...	21.671474
Food	13.895412	20.649781	17.140947	16.638740	16.557008	17.881736	19.822838	16.760152	14.755665	19.149684	...	16.244361
Soda	14.346336	17.140947	44.390220	21.477687	16.788400	22.415114	27.409492	21.478002	20.069215	23.433387	...	20.037168
Beer	15.357792	16.638740	21.477687	28.416039	15.240504	20.739552	22.459002	17.438402	18.277620	19.461693	...	17.317937
Smoke	13.991825	16.557008	16.788400	15.240504	39.239493	16.990493	18.312546	14.179440	14.601341	15.830464	...	16.638705

5 rows × 51 columns

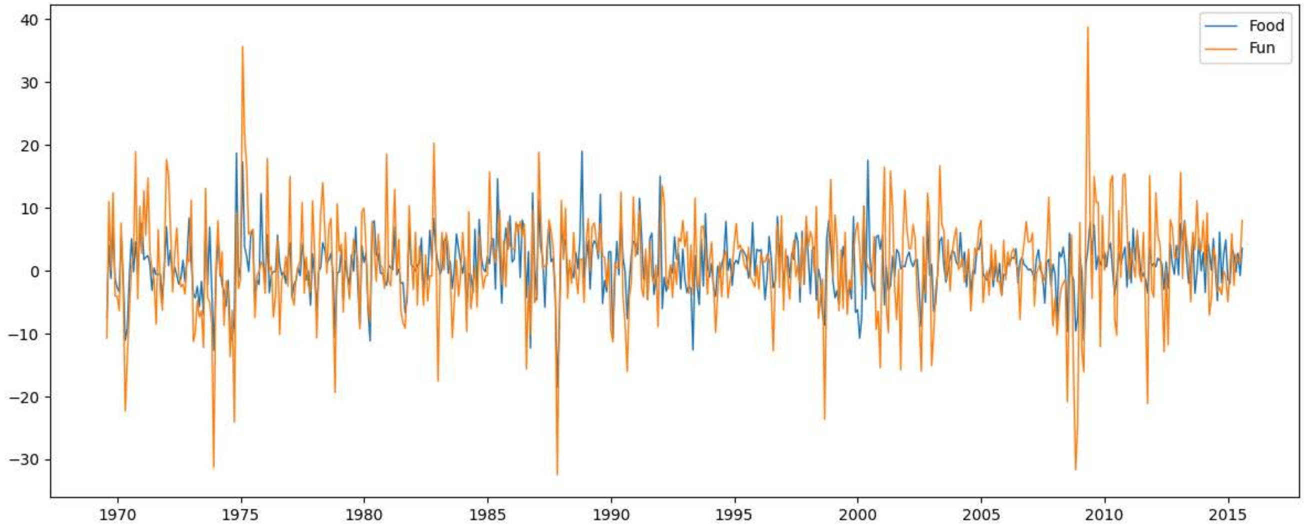
Exercise 8. Plotting

Choose a couple of industry portfolios to plot their time-series.

Use the behavior of these two assets to discuss an important economic event in US history

as you discuss make sure to use the magnitudes in your discussion to show that you do understand what this data means

```
# your code below


fig, ax = plt.subplots(figsize=(12,5))

ax.plot(df_inde.index, df_inde["Food"], label = "Food", linewidth = 1)
ax.plot(df_inde.index, df_inde["Fun"], label = "Fun", linewidth = 1)


ax.legend()
plt.tight_layout()
```



In periods of uncertainty like 2008, the returns seem to go down a lot. Food dropped to around -10pts while fun dropped to about -30pts. Food dropped less because people still need to eat, but they don't want to spend as much on fun.

Exercise 9. Cumulative returns

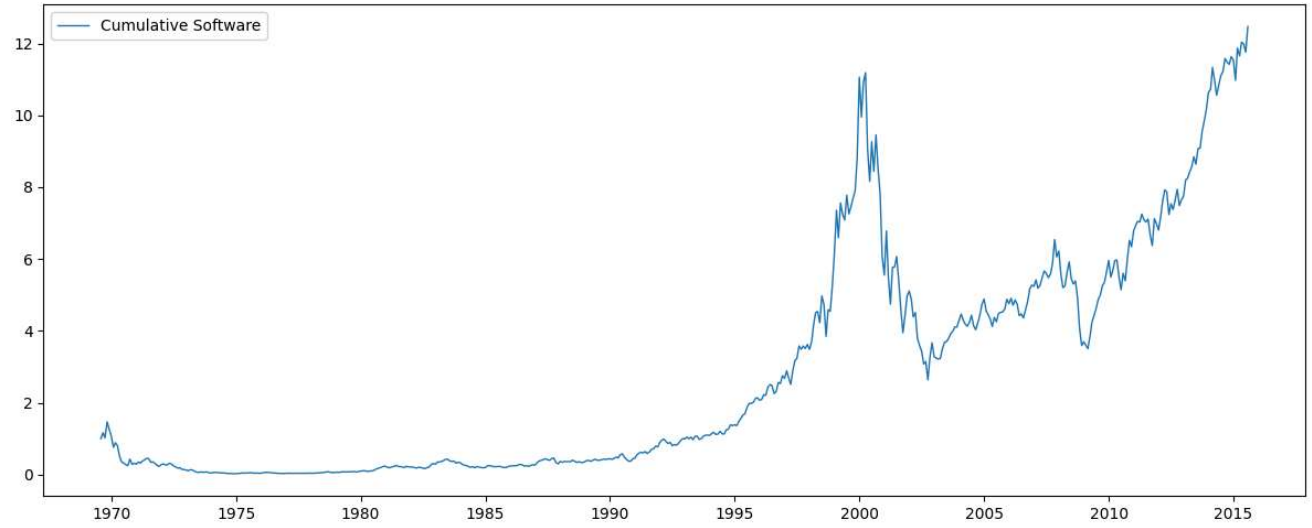
Choose two industry portfolios to plot the cumulative returns over time.

You can plot for the whole period or just a subperiod.

You should explain what the numbers mean in terms of how much money people would have if they had invested in these assets

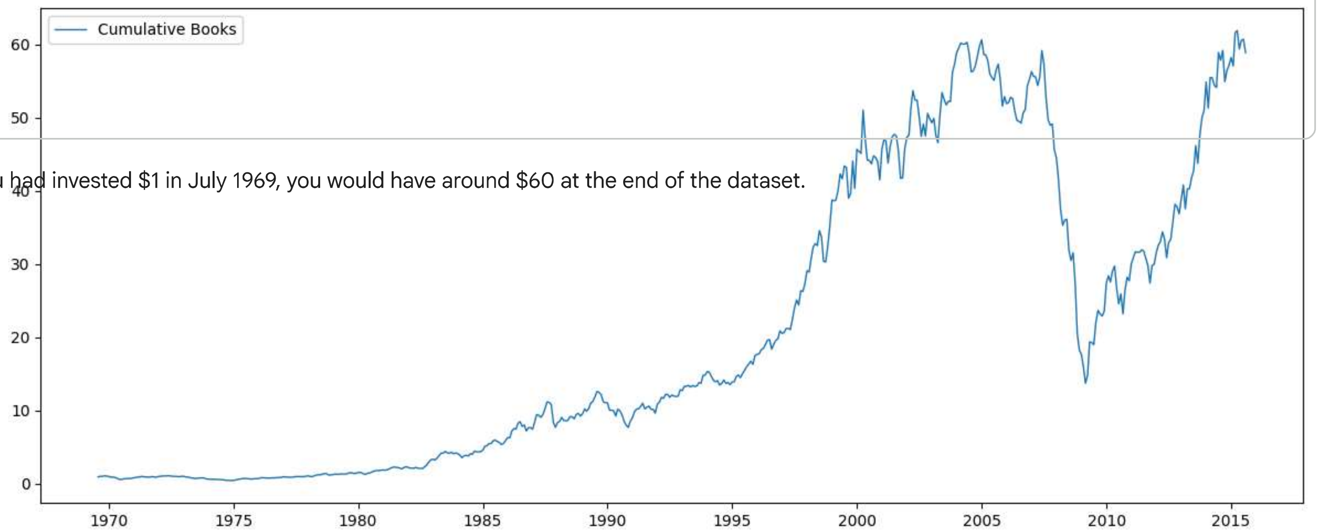
```
fig, ax = plt.subplots(figsize=(12,5))

cum_softw = (1 + df_inde['Softw']/100 + df_inde['RF']/100).cumprod()

cum_softw = cum_softw.fillna(method='ffill').fillna(1)

cum_softw = cum_softw / cum_softw.iloc[0]

ax.plot(cum_softw.index, cum_softw, label = "Cumulative Software", linewidth = 1)


ax.legend()
plt.tight_layout()
```

```
/tmp/ipython-input-1371095993.py:5: FutureWarning: Series.fillna with 'method' is deprecated and will raise in a future version.
  cum_softw = cum_softw.fillna(method='ffill').fillna(1)
```



If you invested $1 in July 1969, you would have more than $12 at the date at the end of the dataset.

```
fig, ax = plt.subplots(figsize=(12,5))

cum_books = (1 + df_inde['Books']/100 + df_inde['RF']/100).cumprod()


ax.plot(df_inde.index, cum_books, label = "Cumulative Books", linewidth = 1)


ax.legend()
plt.tight_layout()
```



If you had invested $1 in July 1969, you would have around $60 at the end of the dataset.