**Course overview**

Hi. This is Xuhu WAN,

the instructor of Python and Statistics for Financial Analysis.

How did people trade stocks over the past few decades?

Floor brokers representing their customers to do trading at the Stock Exchange.

The brokers provided the latest concrete and necessary market information

to their customers to make decisions.

More people traded stocks through online brokers nowadays,

and online broker takes the place of human brokers.

We make our own trading decisions,

based on the financial data and market information

available from the Internet, using data science language.

I combine both Python and Statistics

concepts and applying them into analysing financial data,

such as stock data.

After understanding foundational concepts of statistics,

you are going to realise the statistics concepts by means of

Python packages and syntaxes to perform financial data analysis.

At the beginning of the course,

you are going to learn how to import,

pre-process and save financial data and how to manipulate

the existing data by generating new variables using multiple columns

using Python, followed by building our first simple stock trading model.

You are going to view a more advanced model to predict

stock returns using regression models in the later modules.

Finally, you are going to evaluate the performance of the model with

statistics standards and the financial standards

such as Sharpe Ratio and Maximum Drawdown.

We have setup the Jupyter notebook environment on the course platform.

You can practice the financial analysis and

other examples I explained in the lecture videos.

Simply follow the Jupyter notebook files we prepared, anytime and anywhere.

There's no need to install any applications into your computer.

This course is suitable for any learners who are interested in analysing financial data

using Python. You will get the most out of

the course if you have basic knowledge in probability.

Hopefully, at end of this course,

you can write Python codes to build

the statistical models in assisting your financial analysis.

Most of important,

you can ensure the accuracy and performance

of a financial models you built by model evaluation.

Hope to have you all in the course!

# Getting started with Jupyter Notebook

Jupyter Notebook is an application that allows learners to create documents with Python code. The course team has leveraged the "Jupyter Notebook" feature in Coursera, and created Jupyter Notebook documents for every lecture video. Learners are able to practice the Python coding with the explanatory texts given in the document after going through the lecture videos.

You may refer to the basic tutorial in <https://www.datacamp.com/community/tutorials/tutorial-jupyter-notebook> , which explains how to use Jupyter Notebooks.

For learners who wish to install Jupyter Notebook locally in your computer, you may also follow the same tutorial for installation instructions.

# 1.0 Module Introduction

Hello everyone. In this topic,

we will go through the basics of Python,

which are related to statistical analysis of financial data.

Before we start this topic,

we first need to find out how Python is used in financial industry.

Quantitative analysts and engineers of

investment banks use a Python to build all kinds of models,

to predict returns, and evaluate risks.

Engineers use Python to crawl financial news,

to dig out user's opinions and sentiments.

It is said, this new source of data from social media can greatly

help quantitative analysts to improve the performance of the models.

Python is not only used in investment bank,

it has been widely used even in consumer bank.

A lot of data scientists in consumer banks use the Python to view credit risk models.

To lower the risk of lending,

they can view the customer behavior models.

To predict customers behaviors,

they can use the Python to build the recommendation models to make

a more accurate recommendation of new customers among different departments,

which is called "customer migrations".

Why is Python good for financial data analysis?

Two aspects of Python make it easy for beginners, simplicity and readability.

Simplicity means that the grammar of Python is easy to learn.

Readability means that the code of Python is easy to understand.

We will present this quick tutorial of Python in the following steps,

we will first introduce major packages which will be used in this course.

Next, we will show how to read a stock data from data files using DataFrame.

Then, we'll use this data to explore basic methods and attributes of DataFrame.

In practice, it is always necessary to

generate and transform original variables into other forms.

For example, we need get stock return from stock price.

Therefore, in the next step,

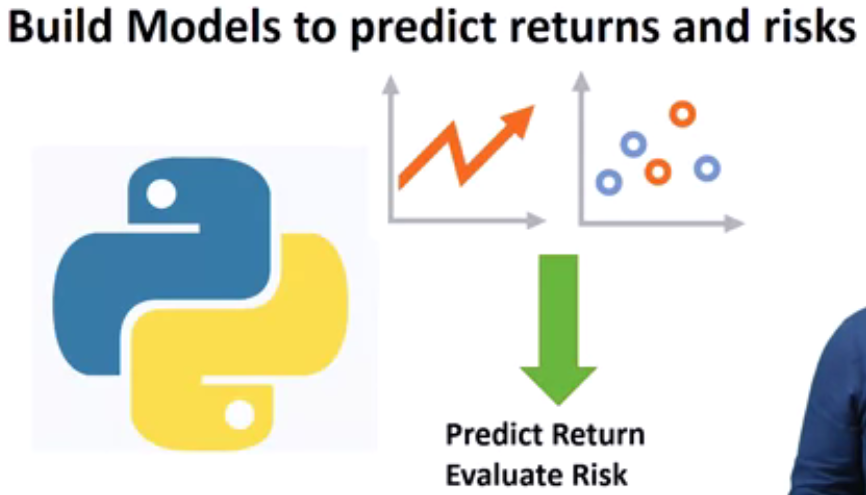
we will present how to generate new variables from our original variables.

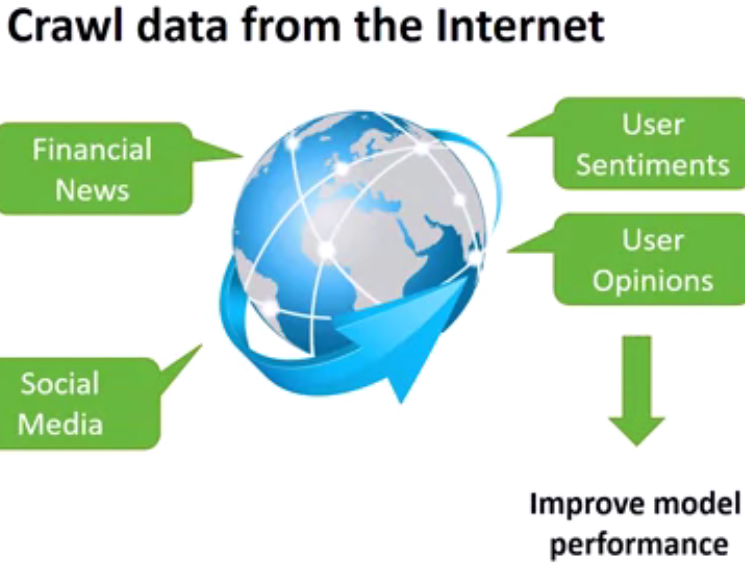
Finally, we will present more advanced workflow using

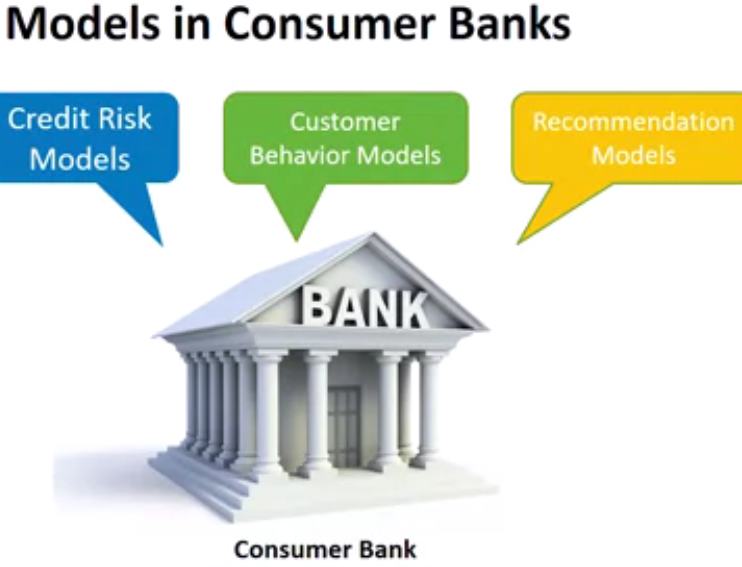
DataFrame to implement a trend following strategy of stock trading.

We expect that, you are able to visualize and play

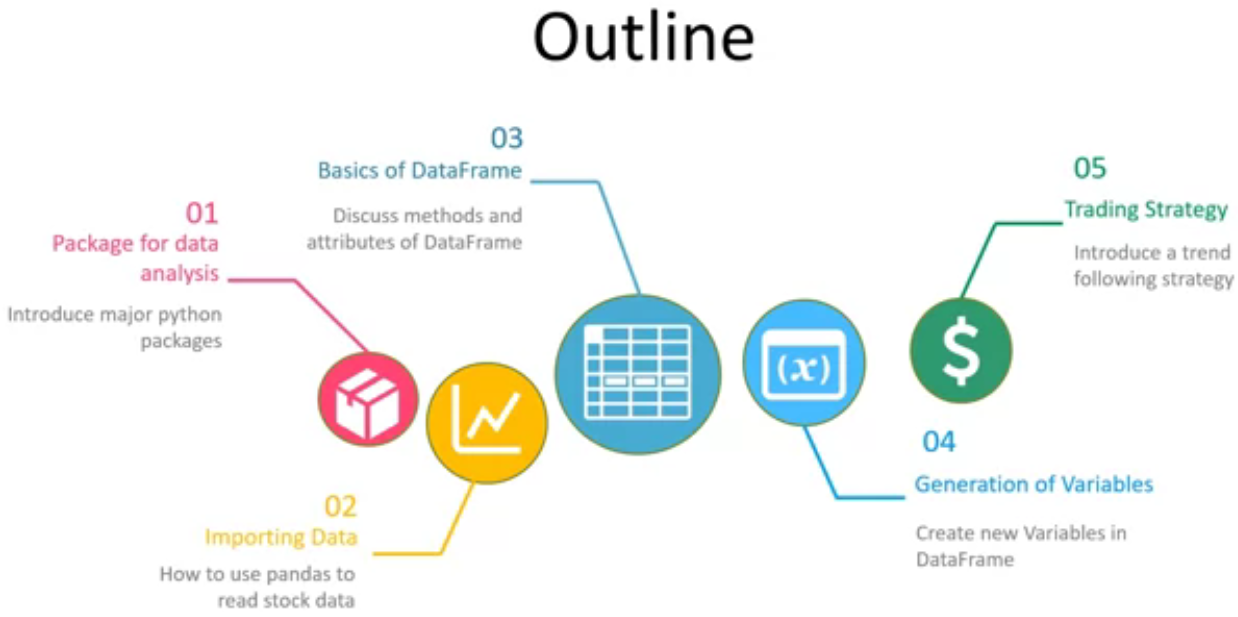
stock data to implement any of your fantastic ideas after the first topic.











# 1.1 Packages for Data Analysis

In this video, we will present packages of Python that will be used for this course.

Pandas is a python package,

providing fast, flexible, and expressive data structures.

It aims to be the fundamental high-level building blocks,

for doing practical real-world data analysis.

For example, DataFrame and the series from Pandas,

are excellent data structures to store table and time series data.

With DataFrame, we can easily pre-process data.

For example, handling missing value, computing pairwise correlation.

We will heavily use DataFrame in this course.

NumPy is a fundamental package for numerical computing of array and matrix.

It is also very convenient tool for generating random numbers,

which could be helpful if we want to shuffle data,

or generate a dataset with normal distribution.

Matplotlib is a plotting package,

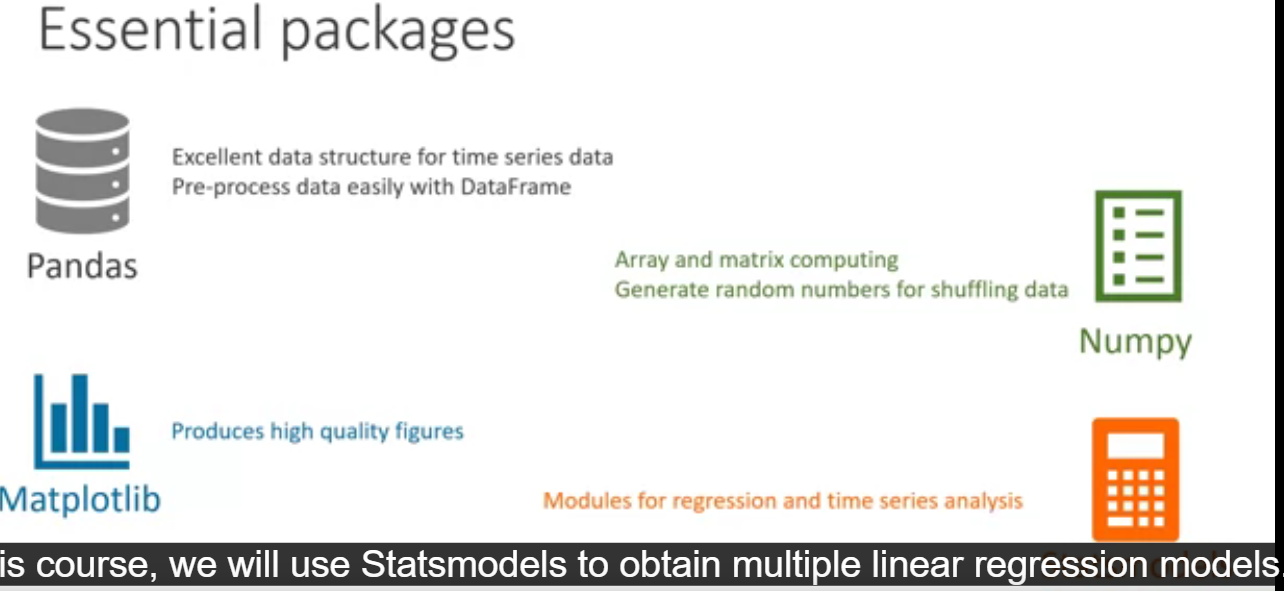
which produces high-quality figures with all kinds of customization.

Statsmodels is a powerful library for statistician.

It contains modules for regression and time series analysis.

In this course, we will use Statsmodels to obtain multiple linear regression models.





Pandas, Numpy, Matplotlib and Statsmodels are the 4 major packages that we are going to use in this course. You may want to get to know more about the software packages in the links below:

pandas: <https://pandas.pydata.org/about.html>

Numpy: <http://www.numpy.org/>

Matplotlib: <https://matplotlib.org/>

Statsmodels: <https://www.statsmodels.org/stable/index.html>

# 1.2 Importing data

The very first step of financial data analysis is to get data.

One of the popular format for storing data is CSV file.

In this video, you will learn how to import files with

CSV format and how to save data into DataFrame structure.

We have to import pandas in the notebook before

we use any functions or data structure from pandas.

To import pandas, you can type,

import pandas in Jupyter Notebook like this.

It is more convenient to use a short name.

Hence, alternatively, you can type import pandas as pd.

Then you just need pd to represent pandas.

For example, if you want to use any modules or functions from pandas,

you can type pd FunctionName or pd ModuleName.

Now, we want to import two data files, facebook.csv and microsoft.csv.

These are historical stock data of Facebook and Microsoft.

We can import these two files with two lines of codes like that.

We are using DataFrame to store data,

therefore we need a pd.

The method from\_csv help us to read the CSV files saved in the form of data.

We assign names fb and ms to these two DataFrames.

We can use a method type to check the data structure of fb.

As we can see,

this is a DataFrame from pandas.

By looking at this output,

the historical data of Facebook has been successfully imported as a DataFrame.

In the next video,

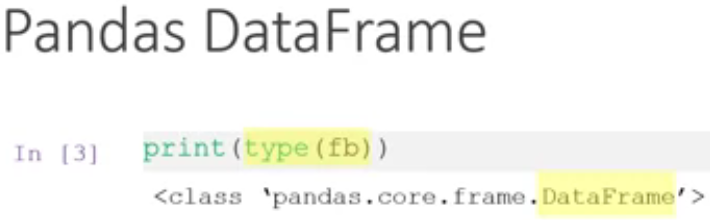
we will explore the details of methods and attributes of DataFrame.











# Importing data from CSV files into Jupyter Notebook

Open Notebook

## **Instructions**

Welcome to the first Jupyter Notebook! In the course, you will practice the python code from the lecture video using the Jupyter Notebook environment embedded on the Coursera platform. You are not required to install any client applications, such as Anaconda in practicing python coding.

A notebook is provided after every lecture video. You are highly recommended to self-learn the materials by following our instructions printed in "Comment", and/or modifying the codes by yourself.

You also need to make use of the Jupyter Notebook in answering some of the quiz weekly questions as well.

Happy learning!

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In this Jupyter Notebook, you will learn how to import stock data, which is usually stored in CSV format, into a new DataFrame for doing data analysis.

## **When using the Jupyter Notebook, be sure you run the codes sequentially from the first block, this is to ensure that all packages and corresponding csv files can be installed/imported properly before practicing other codes**

## **Share My Work**

Generate or update a read-only copy of your lab. [Learn more](https://learner.coursera.help/hc/articles/360044758731-Solving-Common-Issues-with-Coursera-Labs)

# Import data

In this Jupyter Notebook, you will learn how to import data from CSV into Jupyter Notebook

In [2]:



*#import the package "Pandas" into Jupyter Notebook*

**import** pandas **as** pd

In [1]:



*#We import the stock data of Facebook into Jupyter Notebook. The CSV file is located in the folder called "Data" in your Workspace*

*#We then name the DataFrame name as 'fb'*

fb = pd.DataFrame.from\_csv('../data/facebook.csv')

---------------------------------------------------------------------------

NameError Traceback (most recent call last)

<ipython-input-1-0d2f1b815356> in <module>()

**1** #We import the stock data of Facebook into Jupyter Notebook. The CSV file is located in the folder called "Data" in your Workspace

**2** #We then name the DataFrame name as 'fb'

----> 3 fb = pd.DataFrame.from\_csv('../data/facebook.csv')

NameError: name 'pd' is not defined

### Instruction

Now is your turn to import the stock price of Microsoft (microsoft.csv), of which the CSV is located in the same folder, and rename the Dataframe in "ms".

In [3]:



ms = none

---------------------------------------------------------------------------

NameError Traceback (most recent call last)

<ipython-input-3-5c6e42124a11> in <module>()

----> 1 ms = none

NameError: name 'none' is not defined

In [4]:



*# run this cell to ensure Microsoft's stock data is imported*

print(ms.iloc[0, 0])

---------------------------------------------------------------------------

NameError Traceback (most recent call last)

<ipython-input-4-13a65bb40c76> in <module>()

**1** # run this cell to ensure Microsoft's stock data is imported

----> 2 print(ms.iloc[0, 0])

NameError: name 'ms' is not defined

**Expected output:**46.73

# pd.read\_csv or pd.DataFrame.from\_csv

For csv data file loading, pd.DataFram.from\_csv is deprecated since version 0.21.0 and now it stops function.  Use [pandas.read\_csv()](https://pandas.pydata.org/pandas-docs/version/0.23.1/generated/pandas.read_csv.html" \l "pandas.read_csv" \o "pandas.read_csv" \t "_blank) instead. For example

fb = pd.read\_csv('../data/facebook.csv',index\_col=0)

"../data/" is the path to the data. You need to tell which column of the csv file you want to set as index, for example, the first column with "index\_col=0" . Otherwise, it will use 0,1,2,3..as index by default.

It takes time to update video.

# 1.3 Basics of Dataframe

In this video, we will present basic attributes and methods of DataFrame,

which we will use a lot in this course.

Then we will discuss one of the most important skills for beginners,

how to select some portion of data.

To begin with, let's first take a look at what DataFrame looks like.

We can print serveral rows at the top DataFrame,

fb which is historical data for Facebook.

The method we used is head.

It gives us the first five rows of the DataFrame.

As you can see,

Pandas DataFrame is a tabular structure of data.

It has rows and columns.

There is an index,

which gives a base for each row.

There are columns of open,

high, low, adjust close, and volume.

Open and close are the opening price and the closing price.

High and low are the highest in price and the lowest price.

Adjust close is a stock closing price,

on a given day of trading that has been amended to include

any dispution and the corporate actions occurred at anytime before the next day's open.

Every data type in Python had its own methods and attributes.

As the example of attributes,

we can check index like this.

They also can get the first index and the last index,

so that we can extract information

about the starting date and the final date of this data.

We can also can get columns by typing fb columns.

Another important attribute of a DataFrame is the size.

The size of DataFrame can be described by the number of rows and columns.

We use shape to get a size.

In this course, number rows stands for the number of observations,

and the column number represents how many variables in the data.

Play video starting at :2:28 and follow transcript2:28

As methods, head and tail are often used to check whether

data is correct or to check contents of index and columns.

With tail, you can get the last five rows.

In fact, you can specify how many rows

you want by putting number of rows inside the parentheses.

There is another method called describe,

which can give you some summary statistics for each column.

For example, in this output which is also the format of DataFrame,

the first column lists number mean standard deviation etc., for open price.

Play video starting at :3:18 and follow transcript3:18

Next, we will discuss selection of data from DataFrame.

Sometimes you only need a subset data.

For example, select only close price of 2015. How to do that.

There are two ways to slice a DataFrame,

selection by label and selection by position.

For example, if you want the close price on the first day of 2015,

you can use a method loc along with labels of index and the column.

The first entry is the index label.

The second entry of label is a column name.

Play video starting at :4:5 and follow transcript4:05

Alternatively, you also can select by position using iloc.

The first entry stands for row number.

The second is a column number.

The position starts with zero.

Play video starting at :4:23 and follow transcript4:23

You can select multiple rows.

For example, we can get close price for the whole year of 2015.

This selects close price from 2015,

January 1 to 2015, December 31.

We also can select multiple columns.

The colon sign in the second entry of iloc means all columns.

Play video starting at :4:54 and follow transcript4:54

DataFrame has built in method of plot,

which means you can plot data directly without importing matplotlib.

There is a method called plot in pandas DataFrame.

You can easily visualize the close price with this method.

You even can slice close price into different years,

and visualize them one by one.

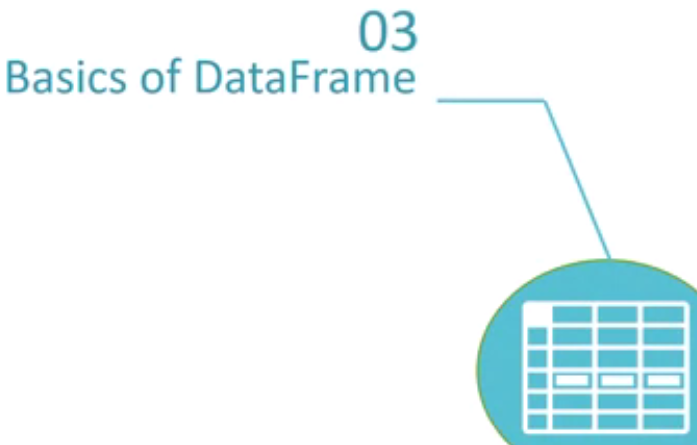
Play video starting at :5:24 and follow transcript5:24

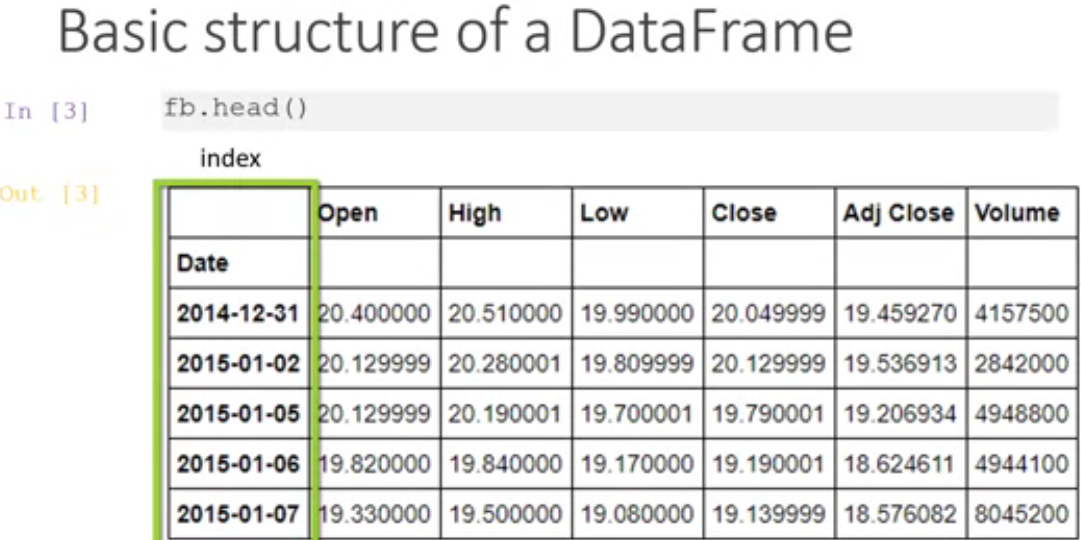
In this video, you have learned attributes methods of DataFrame.

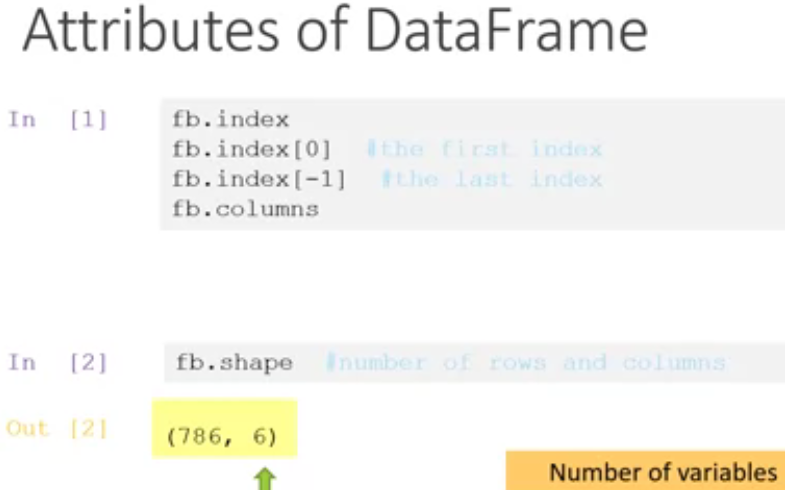
Please, get familiar with this part,

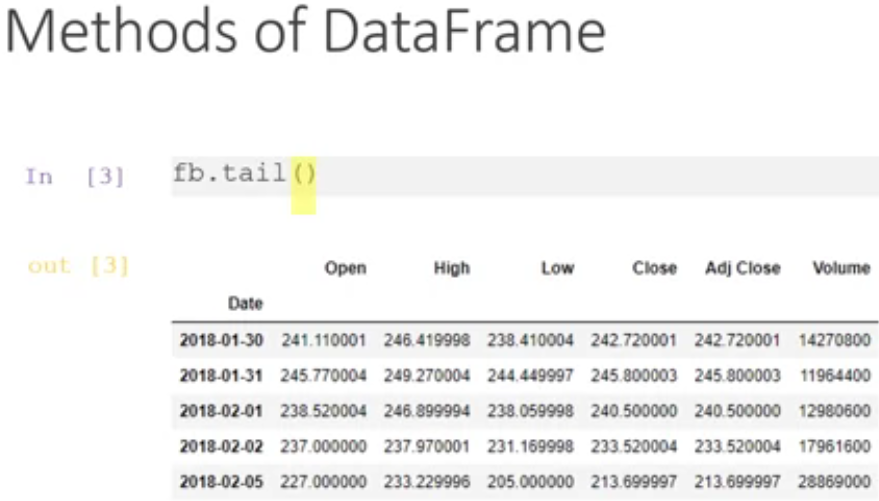
and do some practice in Jupyter Notebook.

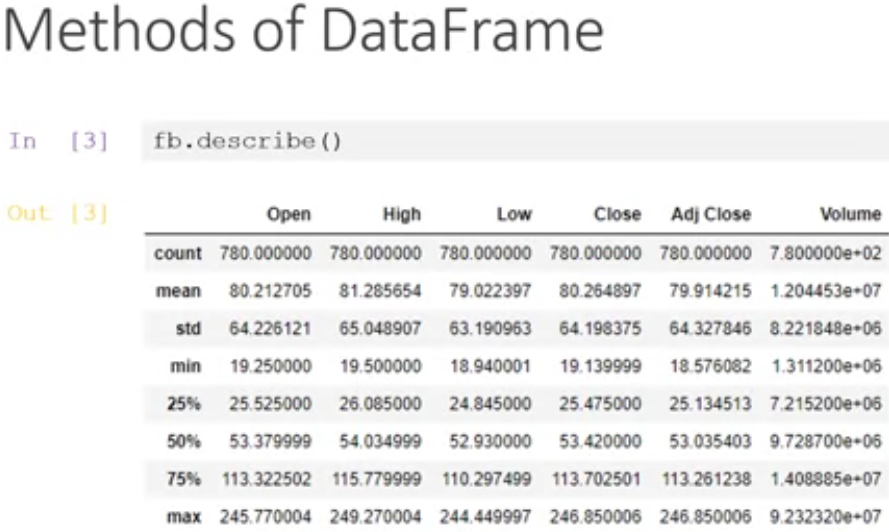
We will use these skills a lot in our formal analysis of financial data.

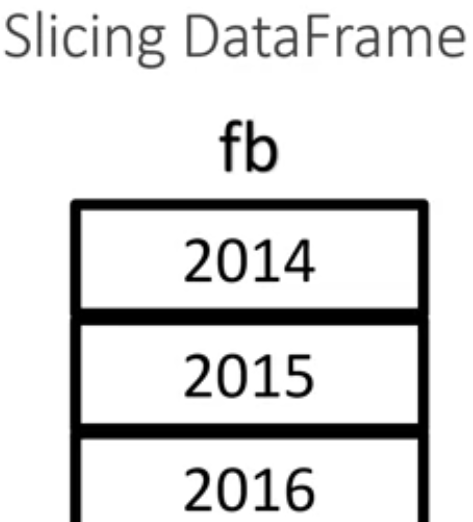




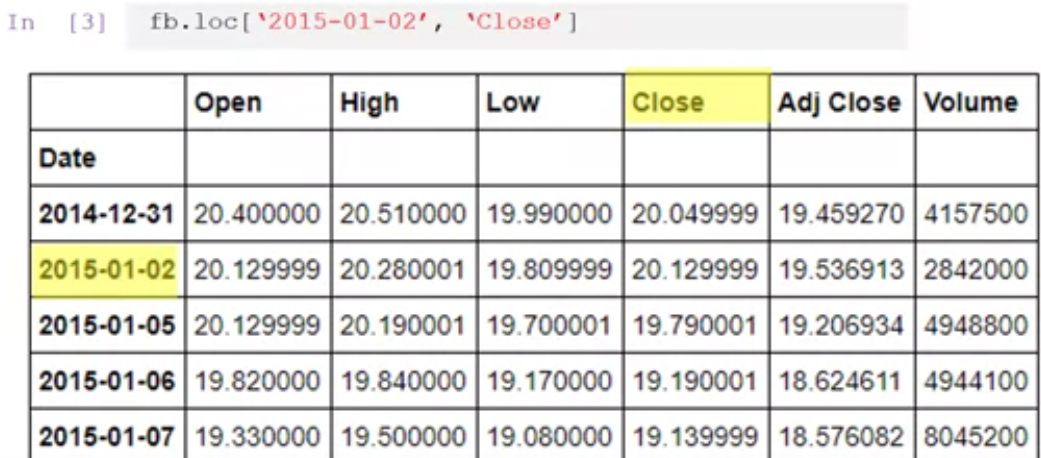


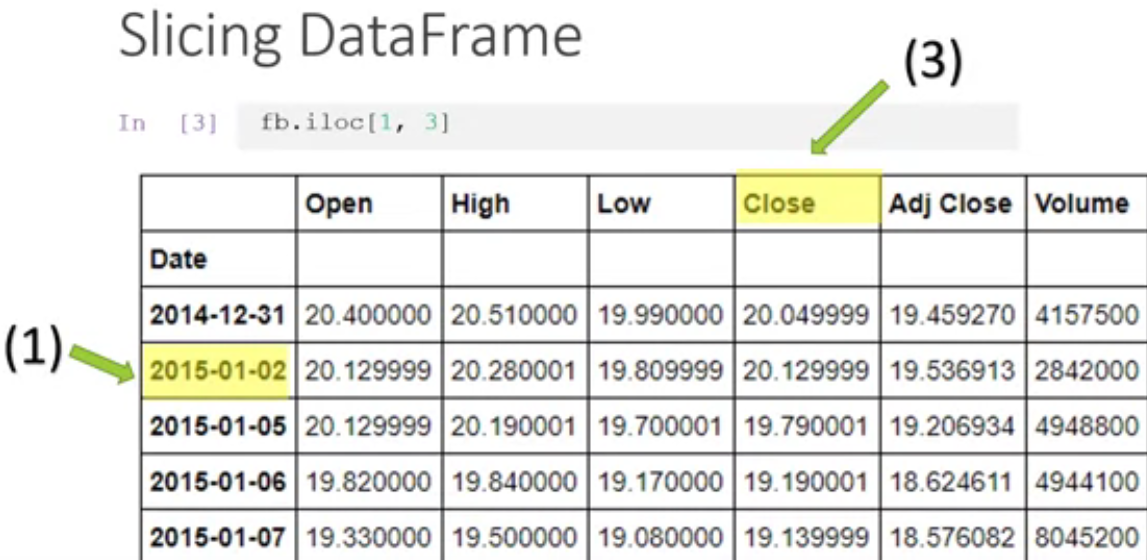


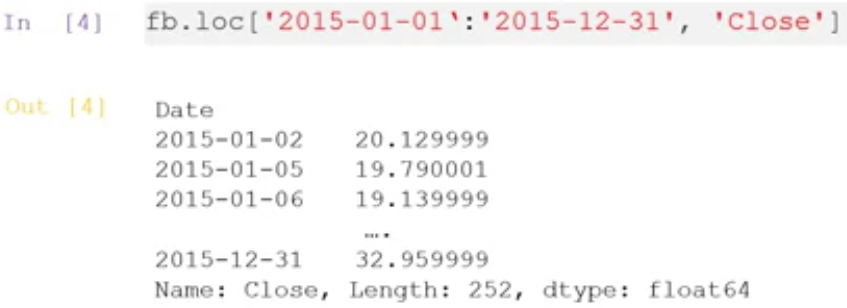


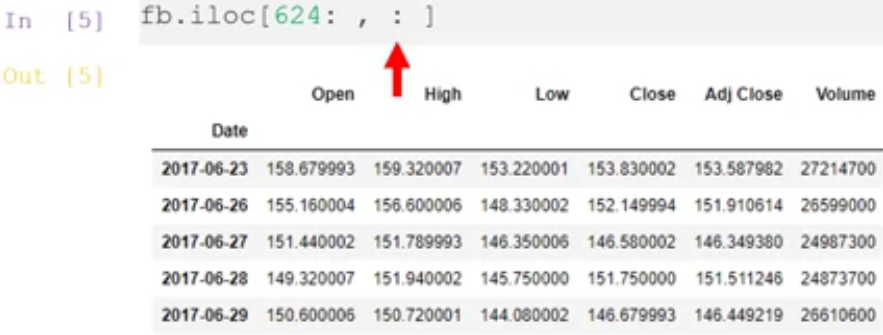














# Basics of DataFrame

Open Notebook

## **Instructions**

In this Jupyter Notebook, you will practice the following basics of DataFrame:

1. Import stock data (in csv format) into a new DataFrame

2. Display the size of a DataFrame using ".shape"

3. Display the summary statistics of a DataFrame using ".describe()"

4. Slice row(s) of data of a DataFrame using "Selection by label" - loc and "Selection of position - iloc"

5. Plot the data of a DataFrame

# DataFrame

In [6]:



*#import the packages "Pandas" and "MatPlotLib" into Jupyter Notebook*

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**%**matplotlib inline

In [7]:



*#import Facebook's stock data*

fb = pd.DataFrame.from\_csv('../data/facebook.csv')

In [8]:



print(fb.head())

Open High Low Close Adj Close Volume

Date

2014-12-31 20.400000 20.510000 19.990000 20.049999 19.459270 4157500

2015-01-02 20.129999 20.280001 19.809999 20.129999 19.536913 2842000

2015-01-05 20.129999 20.190001 19.700001 19.790001 19.206934 4948800

2015-01-06 19.820000 19.840000 19.170000 19.190001 18.624611 4944100

2015-01-07 19.330000 19.500000 19.080000 19.139999 18.576082 8045200

In [9]:



*#It is your turn to import Microsoft's stock data - "microsoft.csv", which is located in the same folder of facebook.csv*

*#Replace "None" with your code*

ms = **None**

In [10]:



*# print head of ms, 1 line*

​

**Expected Output:**

DateOpenHighLowCloseAdj CloseVolume2014-12-3146.73000047.43999946.45000146.45000142.848763215525002015-01-0246.66000047.41999846.54000146.75999843.134731279139002015-01-0546.36999946.73000046.25000046.33000242.738068396739002015-01-0646.38000146.75000045.54000145.65000242.110783364479002015-01-0745.98000046.45999945.49000246.23000042.64581729114100

## Show the size of a DataFrame using "Shape"

In [11]:



print(fb.shape)

(780, 6)

In [12]:



*# print the shape of ms, 1 line*

​

## Show summary statistics of a DataFrame

In [13]:



*# print summary statistics of Facebook*

print(fb.describe())

Open High Low Close Adj Close \

count 780.000000 780.000000 780.000000 780.000000 780.000000

mean 80.212705 81.285654 79.022397 80.264897 79.914215

std 64.226121 65.048907 63.190963 64.198375 64.327846

min 19.250000 19.500000 18.940001 19.139999 18.576082

25% 25.525000 26.085000 24.845000 25.475000 25.134513

50% 53.379999 54.034999 52.930000 53.420000 53.035403

75% 113.322502 115.779999 110.297499 113.702501 113.261238

max 245.770004 249.270004 244.449997 246.850006 246.850006

Volume

count 7.800000e+02

mean 1.204453e+07

std 8.221848e+06

min 1.311200e+06

25% 7.215200e+06

50% 9.728700e+06

75% 1.408885e+07

max 9.232320e+07

In [14]:



*# print summary statistics of Microsoft*

​

## Locate a particular row of data using "Selection by label"

In [15]:



*# select all the price information of Facebook in 2016.*

fb\_2015 = fb.loc['2015-01-01':'2015-12-31']

In [16]:



*# print the price of Facebook on '2015-03-16'*

print(fb\_2015.loc['2015-03-16'])

Open 2.288000e+01

High 2.311000e+01

Low 2.273000e+01

Close 2.297000e+01

Adj Close 2.237908e+01

Volume 5.923900e+06

Name: 2015-03-16 00:00:00, dtype: float64

In [17]:



*# select all the price information of Microsoft in 2016.*

​

In [18]:



*# print the price of Microsoft on '2016-03-16'*

​

**Expected Output:**

Open5.345000e+01High5.460000e+01Low5.340000e+01Close5.435000e+01Adj Close5.187095e+01Volume3.169170e+07

## Locate a particular row of data using "Selection by position"

In [19]:



*# print the opening price of the first row*

print(fb.iloc[0, 0])

20.4

In [20]:



*# print the opening price of the last row*

​

**Expected Output:**90.559998

## Plot the stock data using plot() method

In [21]:



plt.figure(figsize=(10, 8))

fb['Close'].plot()

plt.show()

In [22]:



plt.figure(figsize=(10, 8))

*# plot only the Close price of 2016 of Microsoft, 1 line*

​

Out[22]:

<matplotlib.figure.Figure at 0x7fada3adfc88>

<matplotlib.figure.Figure at 0x7fada3adfc88>

**Expected Ouput:**