

COMP7180 Lab 1 Tutorial - Python and Numpy, Matplotlib and Pandas

Installation of Anaconda

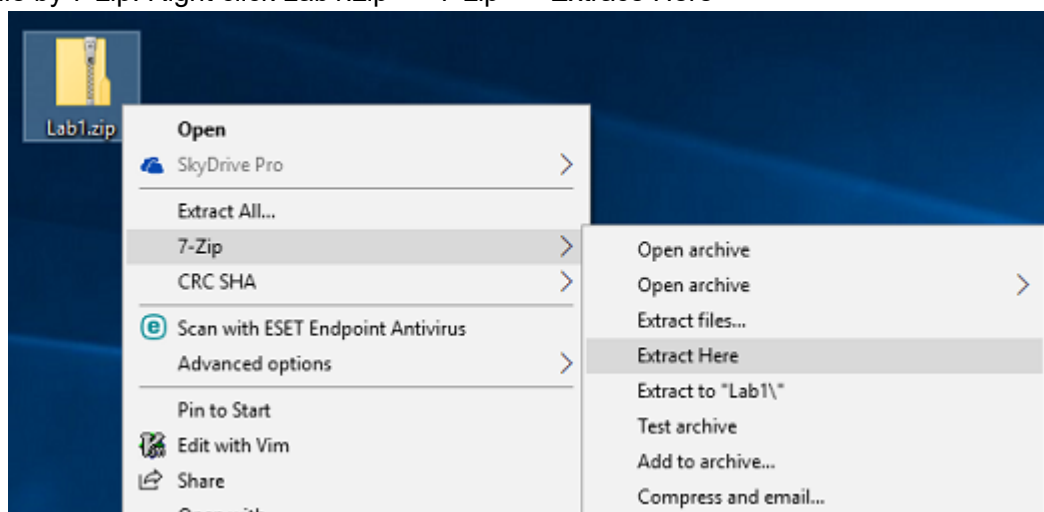
Anaconda is an open source software to manage Python development environments for Windows, Linux, and Mac OS. It provides easy management of a large collection of Python libraries. Anaconda can be obtained from <https://www.anaconda.com/distribution/#download-section> (<https://www.anaconda.com/distribution/#download-section>).

Usage of Jupyter Notebook

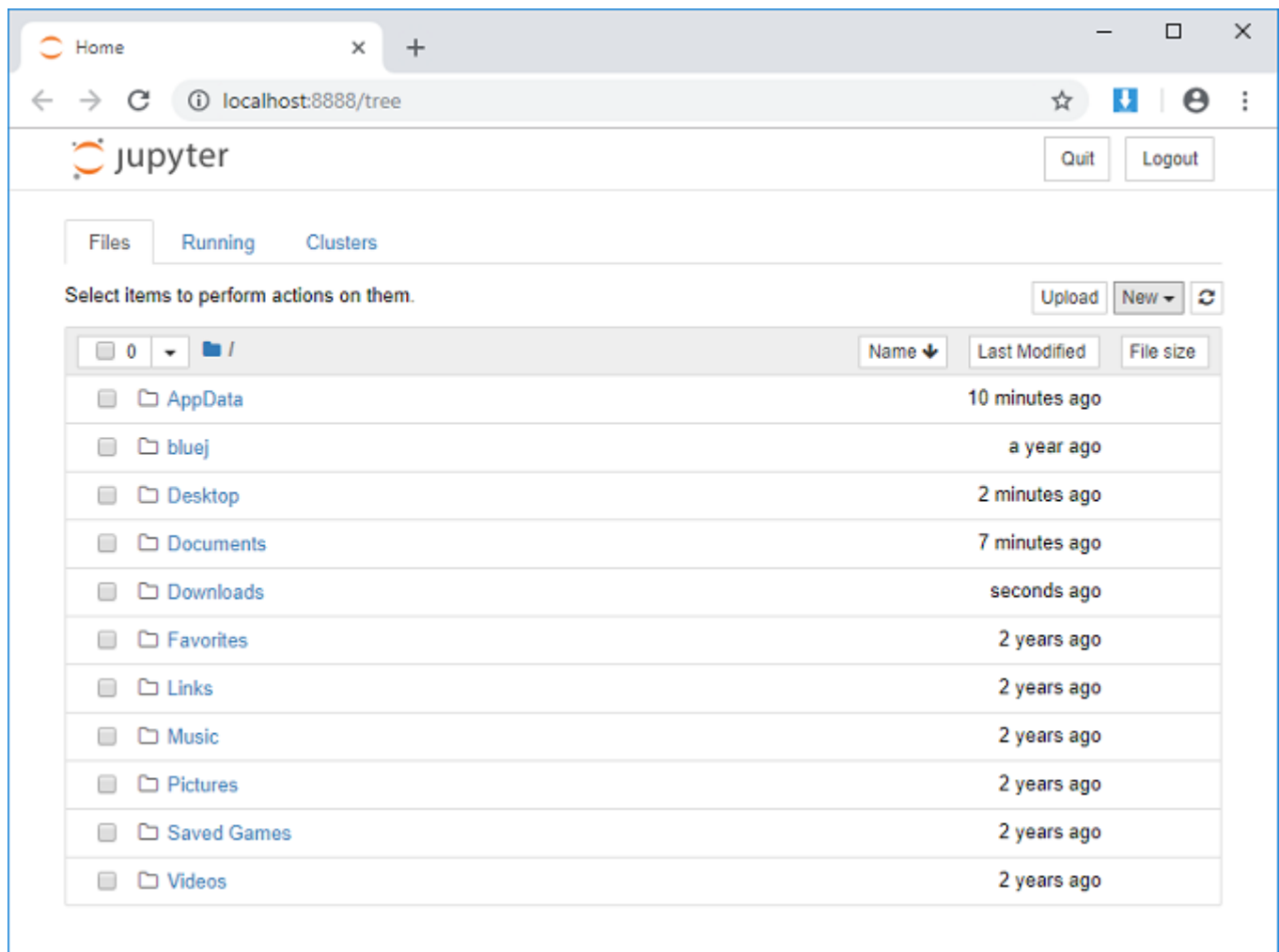
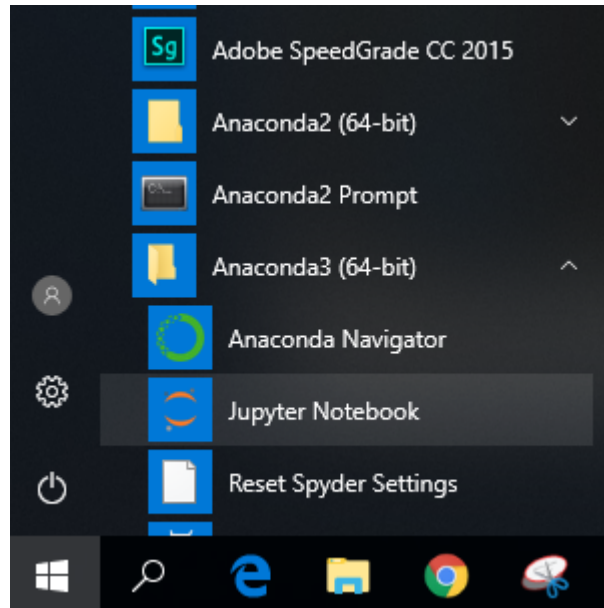
Jupyter Notebook is a web-based powerful tool for interactively developing and presenting data science projects. A notebook integrates codes and its output into a single document that combines visualizations, narrative text etc. Jupyter Notebook is formerly known as iPython notebook.

Steps to open Jupyter Notebook:

1. Download Lab1.zip from Moodle and save it on desktop.
2. Unzip the file by 7-zip. Right click Lab1.zip --> 7-zip --> Extrac Here



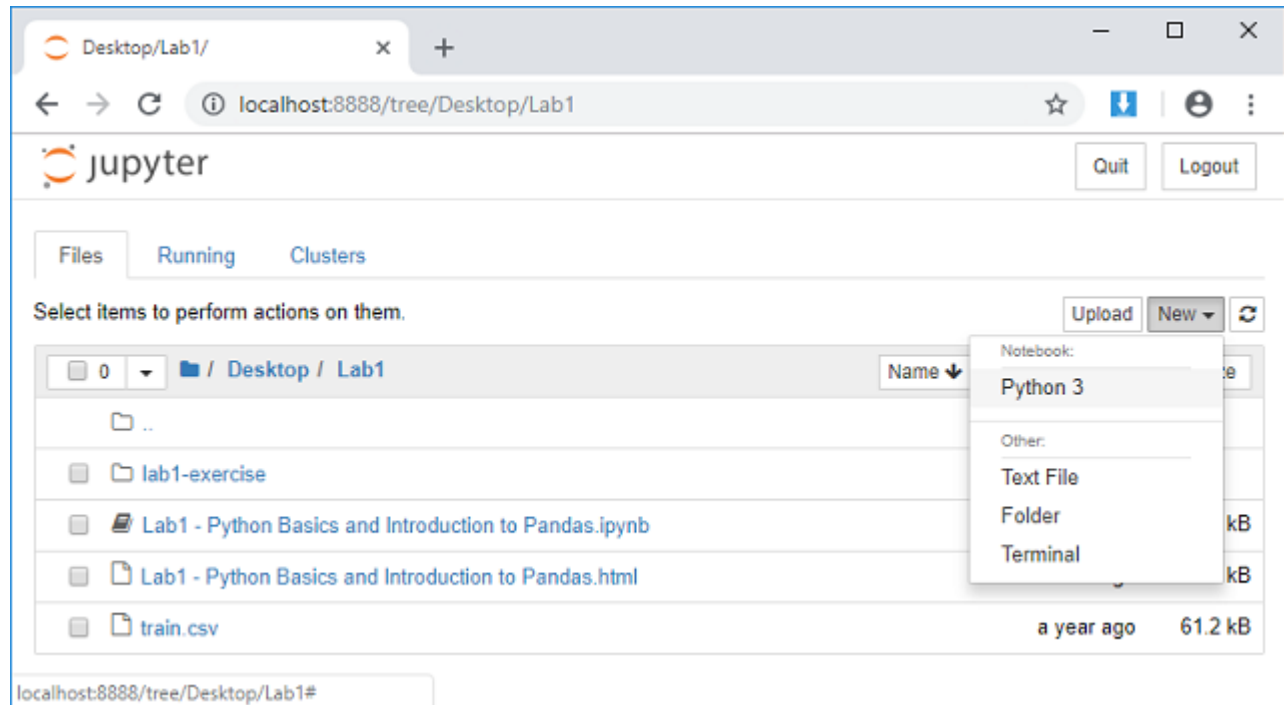
3. Open Jupyter Notebook. Start --> Anaconda3(64-bit) --> Jupyter Notebook



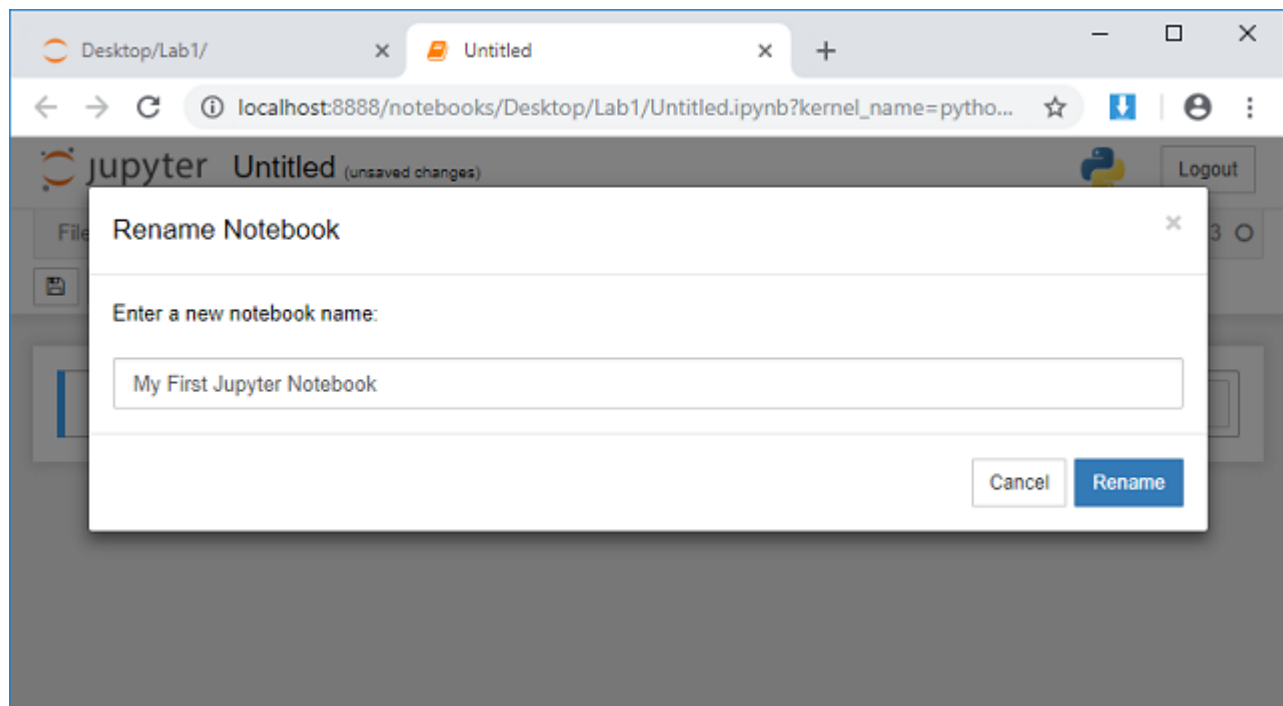
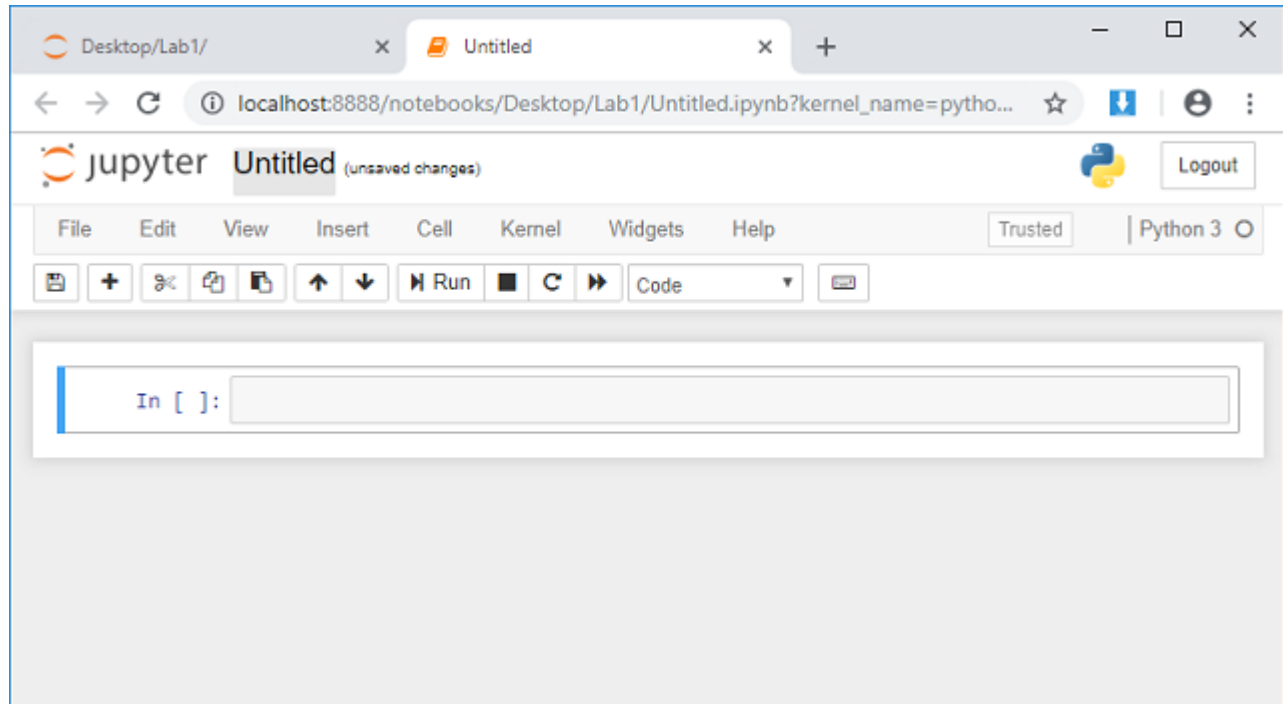
4. Click Desktop and then click inside Lab1.
5. Click on **COMP7180 Lab1 Tutorial - Python Basics; Pandas; Matplotlib.ipynb** to open the notebook

Steps to create a Jupyter Notebook:

1. Create a Jupyter Notebook. Click the 'New' button at the top right and then select 'Python 3'



2. Click on the 'Untitled' and rename the notebook to 'My First Jupyter Notebook'



Editing the Jupyter Notebook:

Modal user interface

Jupyter Notebook has a modal user interface. This means that the keyboard does different things depending on which mode the Notebook is in. There are two modes: command mode and edit mode .

1. Command mode

Command mode is indicated by a grey cell border with a blue left margin:

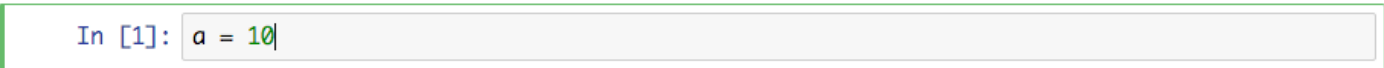
A screenshot of a Jupyter Notebook cell in command mode. The cell has a grey border and a blue left margin. The text inside is "In [1]: a = 10".

```
In [1]: a = 10
```

When you are in command mode, you are able to edit the notebook as a whole, but not type into individual cells. Most importantly, in command mode, the keyboard is mapped to a set of shortcuts that let you perform notebook and cell actions efficiently. For example, if you are in command mode and you press `c` , you will copy the current cell.

2. Edit Mode

Edit mode is indicated by a green cell border and a prompt showing in the editor area:

A screenshot of a Jupyter Notebook cell in edit mode. The cell has a green border and a green left margin. The text inside is "In [1]: a = 10" with a cursor at the end of the line.

```
In [1]: a = 10|
```

When a cell is in edit mode, you can type into the cell, like a normal text editor. Enter edit mode by pressing `Enter` or using the mouse to click on a cell's editor area.

Cell Types

The notebook consists of a sequence of cells. A cell is a multiline text input field, and its contents can be executed by using `Shift-Enter`, or by clicking either the 'Run' button in the toolbar, or 'Run Cell' in the 'Cell' menu bar.

Code cells

A code cell allows you to edit and write new code.

Markdown cells

Markdown cell displays text which can be formatted using markdown language.

After cell creation, a cell is a code cell by default.

Steps to edit Jupyter Notebook:

1. Type in `x = 10` in the code cell. Press `Enter` to move to the second line. Then type `x` in the second line of the same cell.

```
In [ ]: x = 10  
x |
```

2. Press `Shift-Enter` to run the cell.

```
In [1]: x = 10  
x
```

```
Out[1]: 10
```

```
In [ ]:
```

3. Press `Esc` to exit the Edit Mode. Press `m` to change the current cell to a markdown cell.

```
In [1]: x = 10  
x
```

```
Out[1]: 10
```

4. Press `Enter` to enter the Edit Mode. Type in "This is a markdown cell". Then press `Shift-Enter` to run the cell.

```
In [1]: x = 10  
x
```

```
Out[1]: 10
```

This is a markdown cell

```
In [ ]:
```

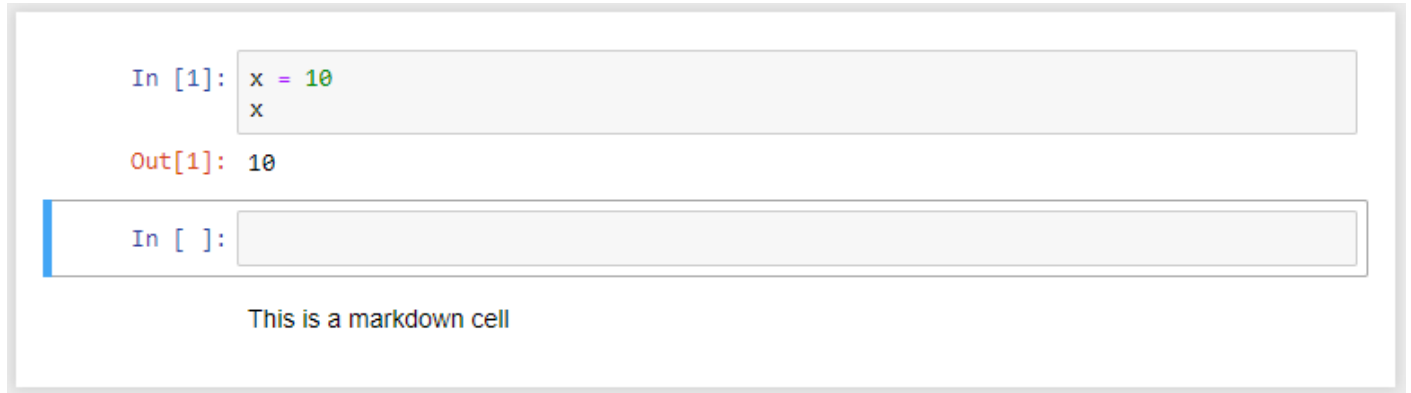
5. Press `Esc` to exit the Edit Mode. Press `dd` to delete the current cell.

```
In [1]: x = 10  
x
```

```
Out[1]: 10
```

This is a markdown cell

6. Press **a** to create a cell above the current cell.



The image shows a Jupyter Notebook interface. The top cell is a code cell with the following content:

```
In [1]: x = 10
x
```

Below the code cell is the output:

```
Out[1]: 10
```

Below the output is a new, empty code cell with the prompt:

```
In [ ]:
```

Below the empty code cell is a markdown cell with the text:

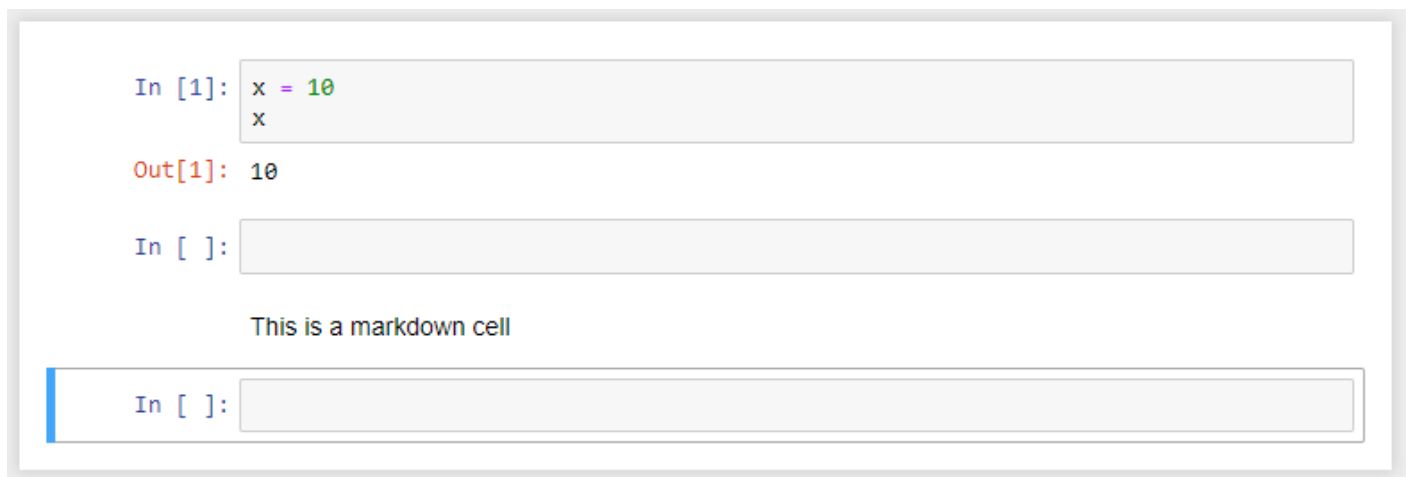
This is a markdown cell

7. Press '**↓**' to move to the markdown cell at the bottom. Or mouse click on the markdown cell to select the cell.



The image shows the same Jupyter Notebook interface as before, but the blue selection bar on the left is now on the markdown cell, indicating it is the active cell.

8. Press **b** to create a cell below the current cell.



The image shows the Jupyter Notebook interface after pressing 'b'. A new code cell has been added below the markdown cell.

The top cell is a code cell with the following content:

```
In [1]: x = 10
x
```

Below the code cell is the output:

```
Out[1]: 10
```

Below the output is an empty code cell with the prompt:

```
In [ ]:
```

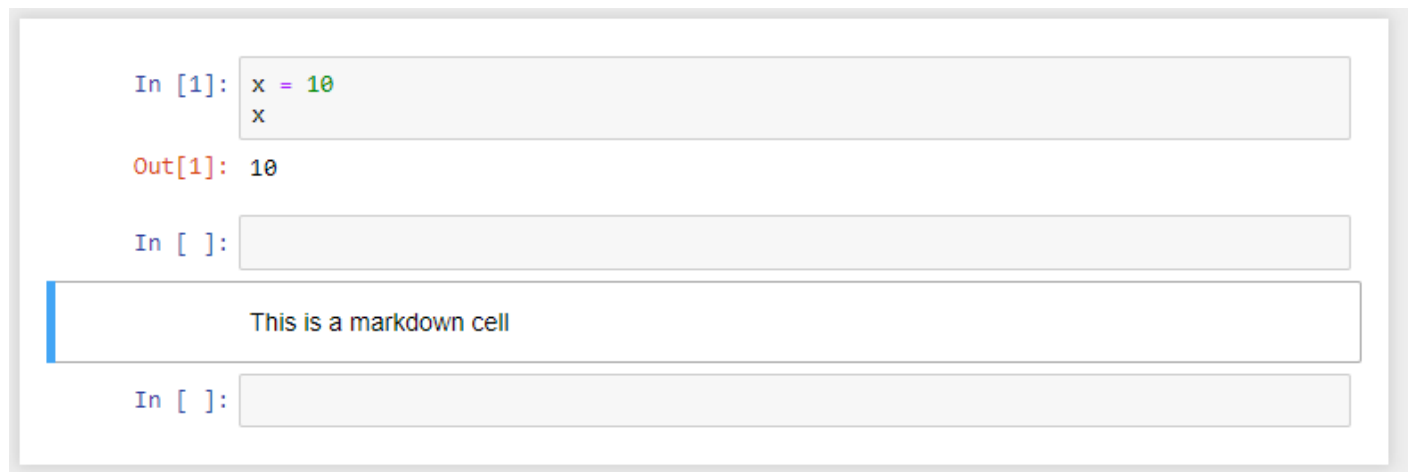
Below the empty code cell is a markdown cell with the text:

This is a markdown cell

Below the markdown cell is a new, empty code cell with the prompt:

```
In [ ]:
```


9. Press '↑' to move back to the markdown cell. Or mouse click on the markdown cell to select the cell.



The image shows a Jupyter Notebook interface with a light gray background. It contains four cells. The first cell is a code cell with the prompt 'In [1]:' in blue, followed by the code 'x = 10' and 'x' on separate lines. The second cell is an output cell with the prompt 'Out[1]:' in red, followed by the value '10'. The third cell is a markdown cell with the prompt 'In []:' in blue, followed by the text 'This is a markdown cell'. The fourth cell is a code cell with the prompt 'In []:' in blue, followed by an empty input field. A blue vertical bar is on the left side of the notebook, and the third cell is currently selected.

```
In [1]: x = 10
x
```

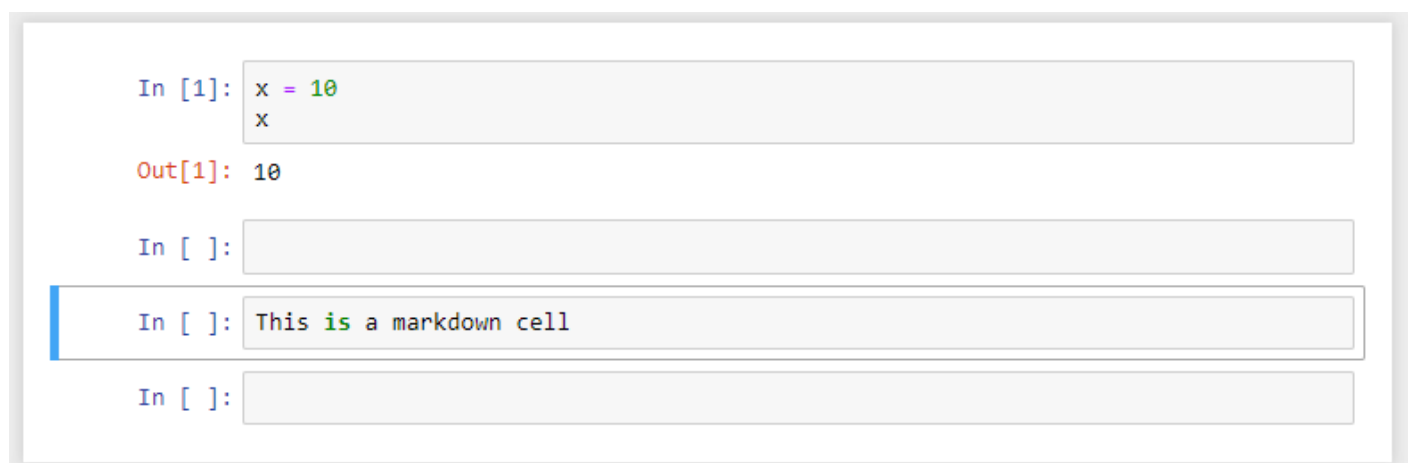
Out[1]: 10

In []:

This is a markdown cell

In []:

10. Press 'y' to change the cell into a code cell.



The image shows the same Jupyter Notebook interface as before, but the third cell has been converted to a code cell. The first cell is a code cell with 'In [1]:', 'x = 10', and 'x'. The second cell is an output cell with 'Out[1]:' and '10'. The third cell is now a code cell with the prompt 'In []:' in blue, followed by the text 'This is a markdown cell'. The fourth cell is a code cell with the prompt 'In []:' in blue, followed by an empty input field. The third cell is currently selected, indicated by the blue vertical bar on the left.

```
In [1]: x = 10
x
```

Out[1]: 10

In []:

```
In [ ]: This is a markdown cell
```

In []:

11. Press `Enter` to enter the edit mode. Delete the original content and type in `y = 20` in the first line and `y` in the second line. Then press `Shift-Enter`.

```
In [1]: x = 10  
x
```

```
Out[1]: 10
```

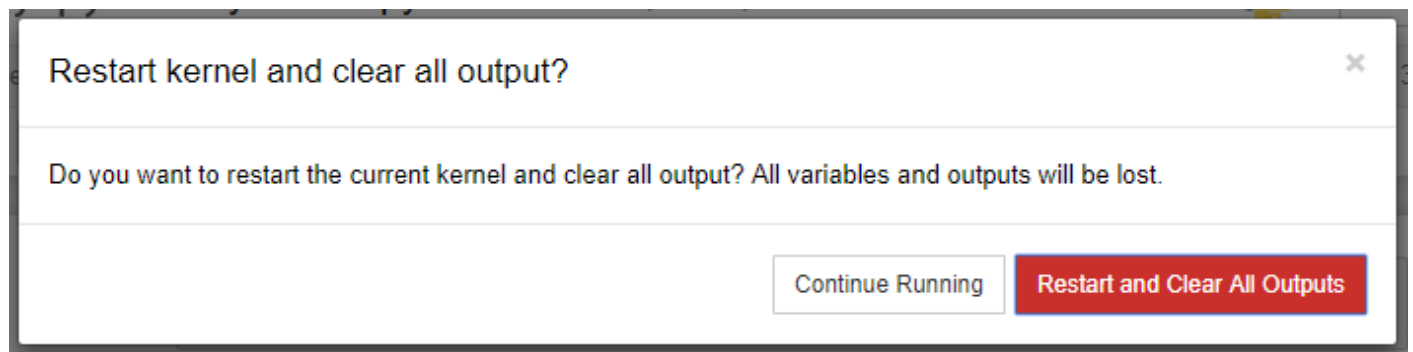
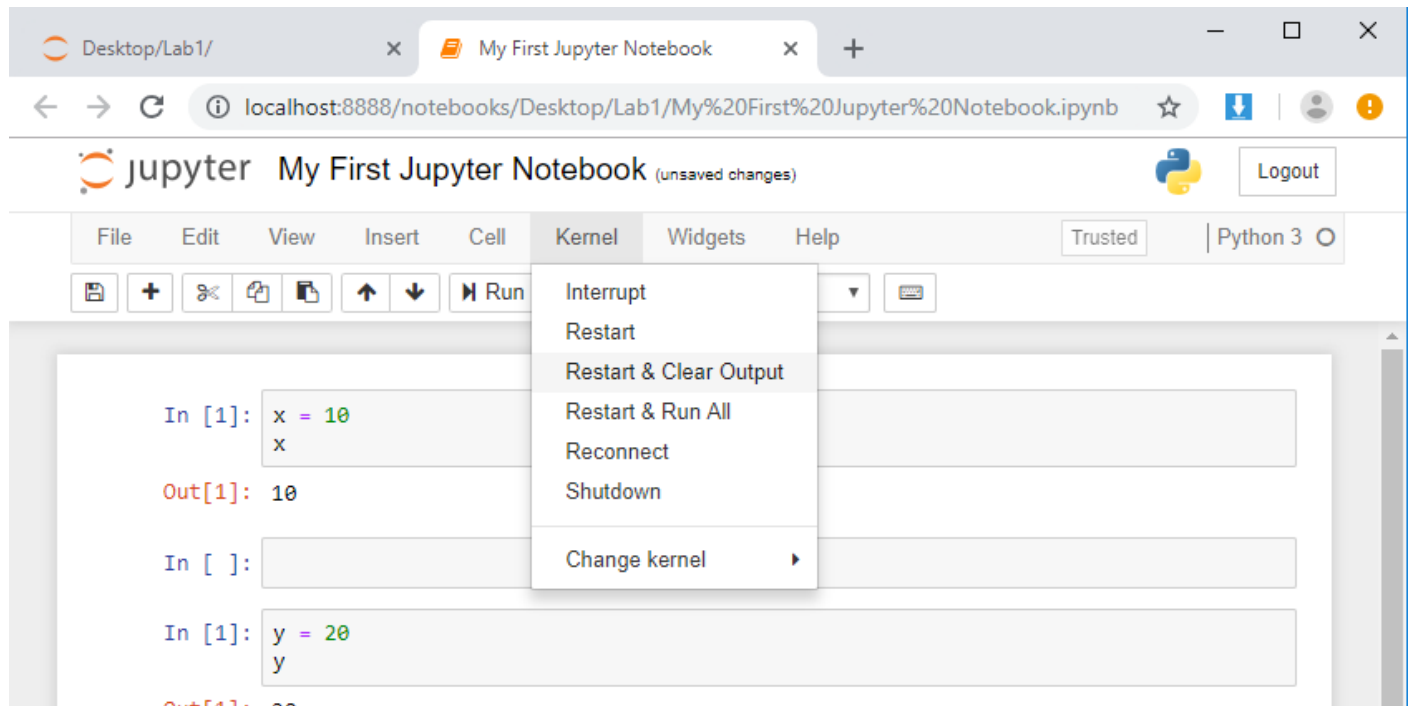
```
In [ ]:
```

```
In [1]: y = 20  
y
```

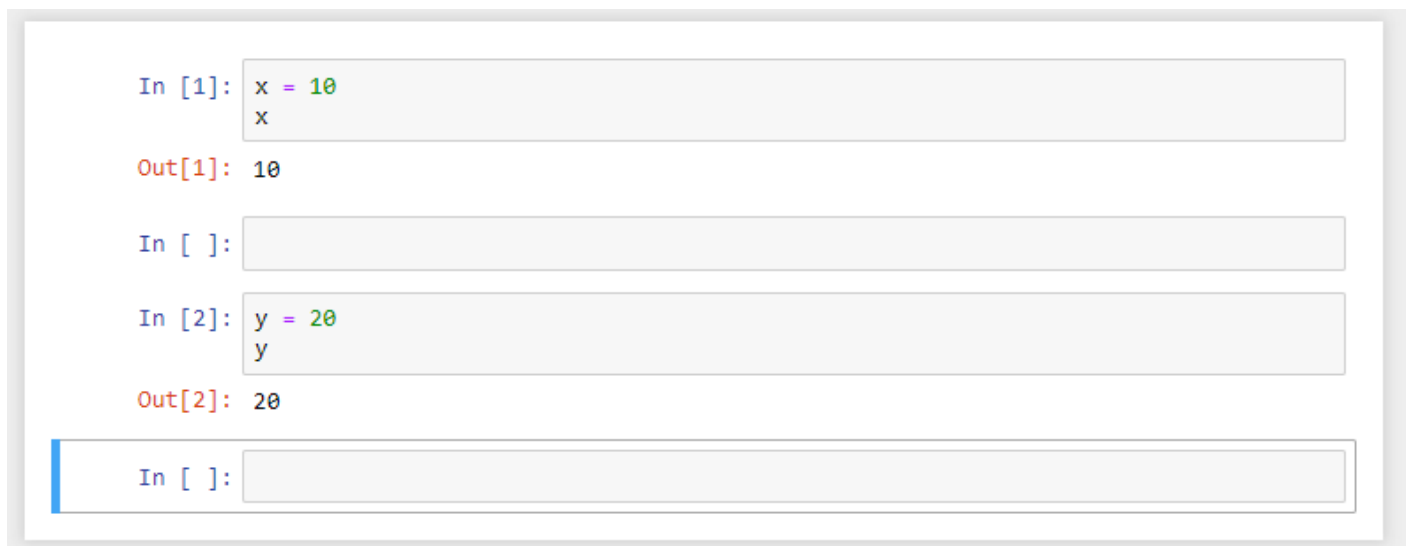
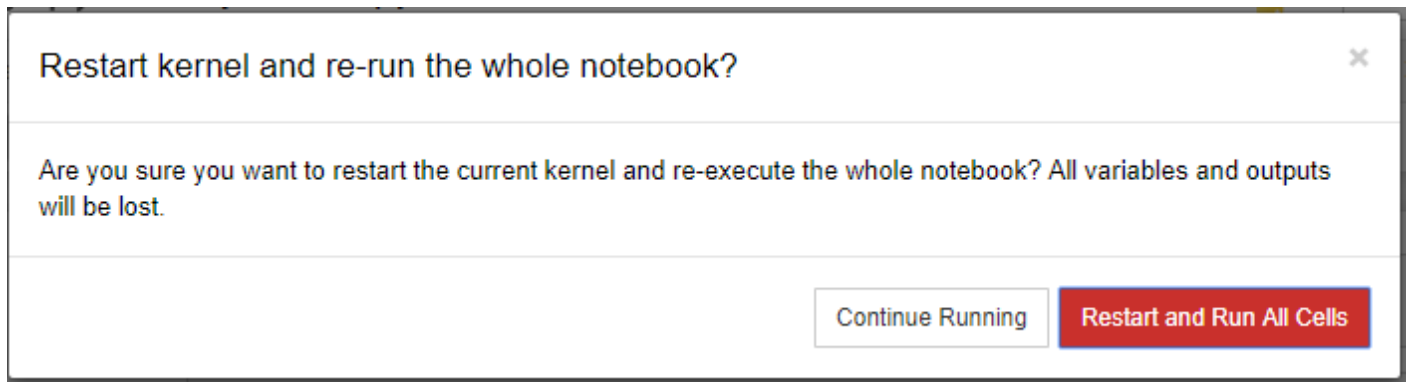
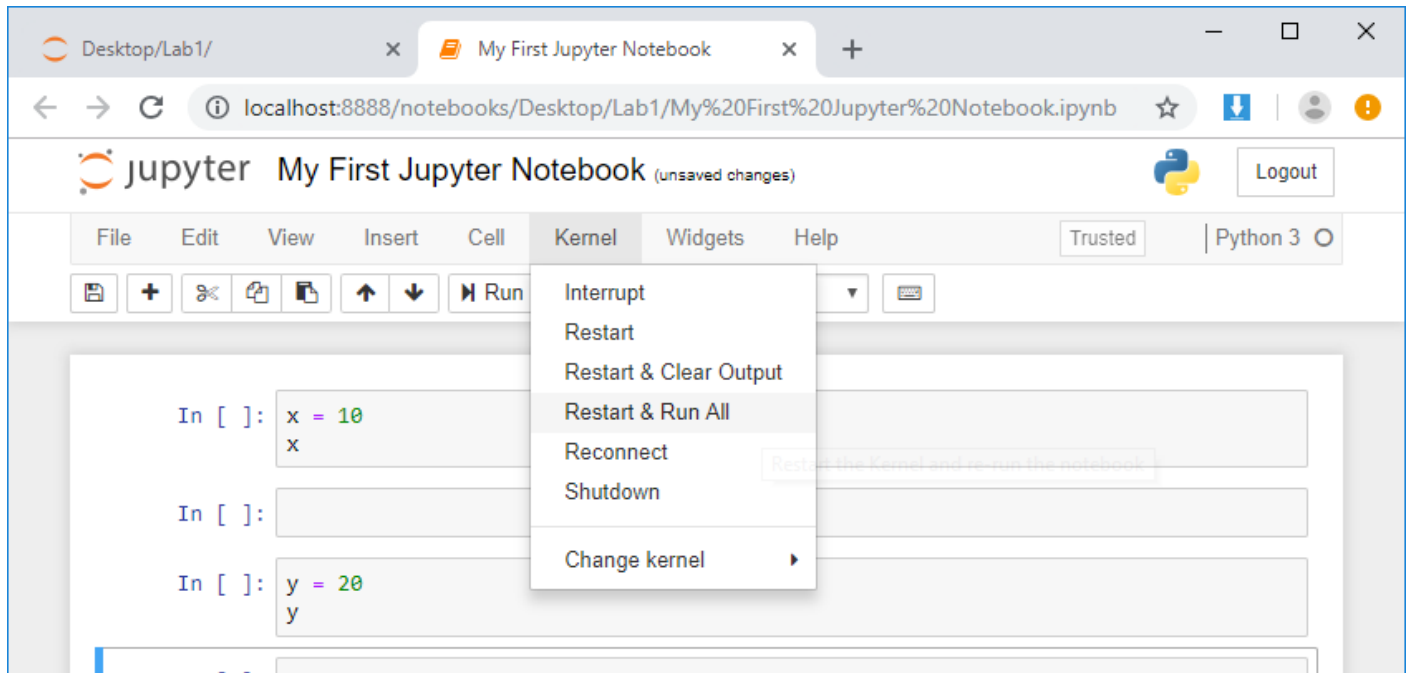
```
Out[1]: 20
```

```
In [ ]:
```

12. Click the **Kernel** menu and select **Restart & Clear Output**. Click **Restart** and **Clear All Outputs** in the pop-up to clear all the output.



13. Click the **Kernel** menu and select **Restart & Run All**. Click **Restart** and **Run All Cells** in the pop-up to run all the cells in the notebook.



Keyboard Navigation

The most important keyboard shortcuts are `Enter` , which enters edit mode, and `Esc` , which enters command mode.

Find more information on keyboard shortcuts at <https://www.cheatography.com/weidadeyue/cheat-sheets/jupyter-notebook/> (<https://www.cheatography.com/weidadeyue/cheat-sheets/jupyter-notebook/>).

Python Basics

Basic Data Types

Use `type()` function to return the object's type

Numbers: Integers and floats

```
In [143]: type(1)
```

```
Out[143]: int
```

```
In [144]: type(-12345)
```

```
Out[144]: int
```

```
In [3]: type(1.0)
```

```
Out[3]: float
```

```
In [146]: type(12e3)
```

```
Out[146]: float
```

```
In [147]: type(25E6)
```

```
Out[147]: float
```

```
In [148]: # create a variable to store value  
x=5  
type(x)
```

```
Out[148]: int
```

```
In [149]: # print x  
x
```

Out[149]: 5

```
In [150]: # Addition  
x + 1
```

Out[150]: 6

```
In [151]: # Subtraction  
x - 2
```

Out[151]: 3

```
In [152]: # Multiplicatin  
x * 3
```

Out[152]: 15

```
In [153]: # Divison  
x / 2
```

Out[153]: 2.5

```
In [154]: # Floor Division  
x // 2
```

Out[154]: 2

```
In [155]: # Modulus  
x % 2
```

Out[155]: 1

```
In [156]: # Exponentiation  
x ** 2    # x to the power 2
```

Out[156]: 25

Booleans

```
In [157]: type(True)
```

Out[157]: bool

```
In [158]: type(False)
```

Out[158]: bool

```
In [159]: x = 10  
         y = 12
```

```
In [160]: # Greater than  
         x > y
```

Out[160]: False

```
In [161]: # Less than  
         x < y
```

Out[161]: True

```
In [162]: # Equal to  
         x == y
```

Out[162]: False

```
In [163]: # Not equal to  
         x != y
```

Out[163]: True

```
In [164]: # Greater than or equal to  
         x >= y
```

Out[164]: False

```
In [165]: # Less than or equal to  
         x <= y
```

Out[165]: True

```
In [166]: t = True  
         f = False  
         type(t)
```

Out[166]: bool

```
In [167]: # print t  
         t
```

Out[167]: True

```
In [168]: # Logical AND  
         t and f
```

Out[168]: False

```
In [169]: # Logical OR  
         t or f
```

Out[169]: True

```
In [170]: # Logical NOT  
not t
```

```
Out[170]: False
```

Strings

```
In [171]: type('This is a string')
```

```
Out[171]: str
```

```
In [172]: h = 'hello'    # String literals can use single quotes  
w = "world"             # or double quotes; it does not matter.
```

```
In [173]: # print  
h
```

```
Out[173]: 'hello'
```

```
In [174]: # String concatenation  
hw = h + ' ' + w  
hw
```

```
Out[174]: 'hello world'
```

```
In [175]: # sprintf stype string formatting  
hw100 = hw12 = '%s %s %d' % (h, w, 100)  
hw100
```

```
Out[175]: 'hello world 100'
```

Data Structure

Common data structures in Python:

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.
- **Tuple** is a collection which is ordered and **unchangeable**. Allows duplicate members.

See <https://docs.python.org/3.5/tutorial/datastructures.html#tuples-and-sequences>
(<https://docs.python.org/3.5/tutorial/datastructures.html#tuples-and-sequences>) for more information.

List

Lists are written with square brackets. It is resizeable and can contain elements of different types.

```
In [176]: # Create a list  
x = [1, 2, 3]
```



```
In [177]: # Print x
x
```

```
Out[177]: [1, 2, 3]
```

```
In [178]: # Access list item by index number
x[2]
```

```
Out[178]: 3
```

```
In [179]: # Update a value in list
x[2] = 'foo' # Note: Lists can contain elements of different types
x
```

```
Out[179]: [1, 2, 'foo']
```

```
In [180]: # Add a new element to the end of the list
x.append('apple')
x
```

```
Out[180]: [1, 2, 'foo', 'apple']
```

```
In [181]: # Remove the last element of the list
x.pop()
x
```

```
Out[181]: [1, 2, 'foo']
```

```
In [182]: # Length of a list
len(x)
```

```
Out[182]: 3
```

```
In [183]: # List concatenation
newList = [1, 2] + [3, 4]
newList
```

```
Out[183]: [1, 2, 3, 4]
```

```
In [184]: # Repeat lists
newList = [1, 2] * 3
newList
```

```
Out[184]: [1, 2, 1, 2, 1, 2]
```

```
In [185]: # Use the in keyword to determine if a specified item is present in a list
2 in [1, 2, 3]
```

```
Out[185]: True
```

```
In [186]: # Loop over the elements of a list
animals = ['cat', 'dog', 'monkey']
for animal in animals:
    print(animal)
```

```
cat
dog
monkey
```

Slicing

In addition to accessing list elements one at a time, Python provides concise syntax to access sublists; this is known as slicing.

When constructing a slice, as in `[m:n]`, the first index number `m` is where the slice starts (inclusive), and the second index number `n` is where the slice end (exclusive).

```
Python indexes and slices for a six-element list.
Indexes enumerate the elements, slices enumerate the spaces between the elements.
```

Index from rear:	-6	-5	-4	-3	-2	-1	<code>a=[0,1,2,3,4,5]</code>	<code>a[1:]==[1,2,3,4,5]</code>	
Index from front:	0	1	2	3	4	5	<code>len(a)==6</code>	<code>a[:5]==[0,1,2,3,4]</code>	
	+---+---+---+---+---+---+						<code>a[0]==0</code>	<code>a[:2]==[0,1,2,3]</code>	
	a b c d e f	<code>a[5]==5</code>	<code>a[1:2]==[1]</code>						
	+---+---+---+---+---+---+						<code>a[-1]==5</code>	<code>a[1:-1]==[1,2,3,4]</code>	
Slice from front:	:	1	2	3	4	5	:	<code>a[-2]==4</code>	
Slice from rear:	:	-5	-4	-3	-2	-1	:		

```

b=a[:]
b==[0,1,2,3,4,5] (shallow copy of a)
```

image source: <https://wiki.python.org/moin/MovingToPythonFromOtherLanguages>

```
In [187]: x = ['a', 'b', 'c', 'd', 'e', 'f']
x
```

```
Out[187]: ['a', 'b', 'c', 'd', 'e', 'f']
```

```
In [188]: #Get a slice from index 2 to 4 (exclusive)
x[2:4]
```

```
Out[188]: ['c', 'd']
```

```
In [189]: # Get a slice from index 2 to the end
x[2:]
```

```
Out[189]: ['c', 'd', 'e', 'f']
```

```
In [190]: # Get a slice from the start to index 2 (exclusive)  
x[:2]
```

```
Out[190]: ['a', 'b']
```

```
In [4]: # Get a slice of the whole list
```

```
In [5]: # Get a slice from index 0 to 5(exclusive) with the negative indexing
```

Dictionary

A dictionary stores (key, value) pairs.

```
d = {  
    <key>: <value>,  
    <key>: <value>,  
    .  
    .  
    .  
    <key>: <value>  
}
```

```
In [7]: # Create a dictionary  
d = {'cat': 'cute',  
     'dog': 'furry'}
```

```
In [8]: # Print d  
d
```

```
Out[8]: {'cat': 'cute', 'dog': 'furry'}
```

```
In [9]: # Get an entry from a dictionary  
d['cat']
```

```
Out[9]: 'cute'
```

```
In [10]: # Check if a dictionary has a given key  
'cat' in d
```

```
Out[10]: True
```

```
In [11]: # Set an entry in a dictionary  
d['fish'] = 'wet'  
d
```

```
Out[11]: {'cat': 'cute', 'dog': 'furry', 'fish': 'wet'}
```

```
In [12]: # Remove an element from a dictionary
del d['fish']
d
```

```
Out[12]: {'cat': 'cute', 'dog': 'furry'}
```

```
In [13]: # Length of a dictionary. i.e number of key-value pairs
len(d)
```

```
Out[13]: 2
```

```
In [16]: for key in d:
          print(key)

        for key, value in d.items():
          print(key, value)
```

```
cat
dog
cat cute
dog furry
```

The values in dictionaries are not limited to single strings or words. Values can be any Python object such as numbers, lists, tuples, or even other dictionaries.

```
In [200]: # Create a dictionary with lists as values
d2 = {
    "name": ["Ada", "Bob", "Cindy"],
    "age": [18, 24, 15],
    "gender": ['F', 'M', 'F']
}
print(d2)
```

```
{'name': ['Ada', 'Bob', 'Cindy'], 'age': [18, 24, 15], 'gender': ['F', 'M', 'F']}
```

```
In [201]: # Use in to determine if a specified key is present in a dictionary use the in keyword:
"name" in d2
```

```
Out[201]: True
```

```
In [202]: # Iterate over the keys in a dictionary
d = {'person': 2, 'cat': 4, 'spider': 8}
for key in d:
    print('A %s has %d legs' % (key, d[key]))
```

```
A person has 2 legs
A cat has 4 legs
A spider has 8 legs
```

Tuples

A tuple is an (immutable) ordered list of values. A tuple is in many ways similar to a list; one of the most important differences is that tuples can be used as keys in dictionaries and as elements of sets, while lists cannot.

```
In [203]: # Create a tuple  
x = (1, 'a', 2, 'b')  
x
```

```
Out[203]: (1, 'a', 2, 'b')
```

```
In [204]: # Access tuple item by index number  
x[1]
```

```
Out[204]: 'a'
```

```
In [205]: # Length of a tuple  
len(x)
```

```
Out[205]: 4
```

```
In [206]: # Tuple concatenation  
newTuple = (1, 2, 3) + (4, 5, 6)  
newTuple
```

```
Out[206]: (1, 2, 3, 4, 5, 6)
```

```
In [207]: # Repeat tuple  
t = (1, 2, 3) * 3  
t
```

```
Out[207]: (1, 2, 3, 1, 2, 3, 1, 2, 3)
```

```
In [208]: # Use the in keyword to determine if a specified item is present in a tuple  
3 in (1, 2, 3)
```

```
Out[208]: True
```

```
In [209]: # Loop over the elements of a tuple  
fruits = ("apple", "banana", "cherry")  
for fruit in fruits:  
    print(fruit)
```

```
apple  
banana  
cherry
```

```
In [210]: # Create a dictionary with tuple keys
points = {(1, 2): 'A',
          (3, 4): 'B',
          (5, 6): 'C'}

points[(3,4)]
```

Out[210]: 'B'

Control Flow

If statements

```
In [211]: # if
x = 0

if x == 0:
    print(x, "is zero")
```

0 is zero

```
In [212]: # if-else
x = 5

if x == 0:
    print(x, "is zero")
else:
    print(x, "is not zero")
```

5 is not zero

```
In [213]: # if-elseif
x = 15

if x == 0:
    print(x, "is zero")
elif x > 0:
    print(x, "is positive")
elif x < 0:
    print(x, "is negative")
else:
    print(x, "is unlike anything I've ever seen...")
```

15 is positive

while loop

```
In [214]: # while
i = 0
while i < 5:
    print(i)
    i = i + 1
```

```
0
1
2
3
4
```

for loop

```
In [215]: # for
for item in range(5, 10):
    print(item)
```

```
5
6
7
8
9
```

Functions

Python functions are defined using the `def` keyword. See

<https://docs.python.org/3.5/tutorial/controlflow.html#defining-functions>

(<https://docs.python.org/3.5/tutorial/controlflow.html#defining-functions>) for more information.

```
In [216]: # A function to determine the sign of a number
def sign(x):
    if x > 0:
        return 'positive'
    elif x < 0:
        return 'negative'
    else:
        return 'zero'

#call the function
sign(5)
```

```
Out[216]: 'positive'
```

```
In [217]: # A function can return more than one value
def squareCube(x):
    return x**2 , x**3    # return the square and the cube of x

#call the function
sq, cub = squareCube(5)
```

```
In [218]: # Print sq
sq
```

Out[218]: 25

```
In [219]: # Print cub
cub
```

Out[219]: 125

Introduction to Pandas

Pandas is an open source library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

Titanic Dataset

- Provides information on the fate of passengers on the Titanic, summarized according to economic status (class), sex, age and survival.
- Below are the features provided in the dataset.
 - Passenger Id: and id given to each traveler on the boat
 - Survived: the passenger survived(1) or dead(0)
 - Pclass: the passenger class. It has three possible values: 1,2,3 (first, second and third class)
 - The Name of the passenger
 - Sex: gender of the passenger
 - Age: age of the passenger
 - SibSp: number of siblings and spouses traveling with the passenger
 - Parch: number of parents and children traveling with the passenger
 - The ticket number
 - The ticket Fare
 - The cabin number
 - The embarkation. This describe three possible areas of the Titanic from which the people embark. Three possible values S,C,Q
- Learn more about the titanic dataset: [Kaggle.com \(https://www.kaggle.com/c/titanic/data\)](https://www.kaggle.com/c/titanic/data)
- We will use the training dataset, train.csv, to demonstrate how to use pandas

Import the pandas library

```
In [17]: import pandas as pd
```

Load Data into Pandas DataFrame

Pandas DataFrames are two-dimensional labeled data structures with columns of potentially different types. In plain terms, think of a DataFrame as a table of data, i.e. a single set of formatted two-dimensional data, with the following characteristics:

- There can be multiple rows and columns in the data
- Each row represents a sample of data
- Each column contains a different variable that describes the samples
- The data in every column is usually the same type of data – e.g. numbers, strings, dates

```
In [19]: # Load CSV files into Python to create Pandas Dataframes using the read_csv function  
# syntax: pd.read_csv('file path',  
#                header = 'Row number to use as the column names' or None,  
#                sep = 'delimiter')  
  
df = pd.read_csv('train.csv', header=0, sep=',')
```

Exploring the data

```
In [20]: # check the size of the dataframe  
df.shape
```

```
Out[20]: (891, 12)
```

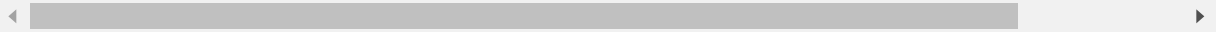
There are 891 rows and 12 columns in the dataframe

```
In [21]: # chek if the data Loaded properly

# show the first few rows, 5 rows by defalut
df.head()
```

Out[21]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	



```
In [22]: # show the first 2 rows
df.head(2)
```

Out[22]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85



In [23]: *# show the last few rows, 5 by default*
`df.tail()`

Out[23]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	Na
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B4
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	Na
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C14
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	Na



In [24]: *# show the last 3 rows*
`df.tail(3)`

Out[24]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	Na
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C14
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	Na



```
In [25]: # set the index of the dataframe to PassengerId
df = df.set_index('PassengerId')
df.head()
```

Out[25]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin
PassengerId										
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	female	38.0	1	0	PC 17599	71.2833	C85
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C125
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN

```
In [26]: # display a summary of the dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 11 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Survived    891 non-null    int64
1   Pclass      891 non-null    int64
2   Name        891 non-null    object
3   Sex         891 non-null    object
4   Age         714 non-null    float64
5   SibSp       891 non-null    int64
6   Parch       891 non-null    int64
7   Ticket      891 non-null    object
8   Fare        891 non-null    float64
9   Cabin       204 non-null    object
10  Embarked    889 non-null    object
dtypes: float64(2), int64(4), object(5)
memory usage: 83.5+ KB
```

As we can see in the output, the summary includes list of all columns with their data types and the number of non-null values in each column. we also have the value of rangeindex provided for the index axis.

```
In [27]: # get basic statistics on numerical columns in the dataframe
df.describe()
```

Out[27]:

	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```
In [28]: # check if any missing value in the dataframe
df.isnull().values.any()
```

Out[28]: True

```
In [29]: # check if any missing value in each column
df.isnull().any()
```

```
Out[29]: Survived    False
Pclass      False
Name        False
Sex         False
Age         True
SibSp       False
Parch       False
Ticket      False
Fare        False
Cabin       True
Embarked    True
dtype: bool
```

```
In [30]: # count the missing value in each column
df.isnull().sum()
```

```
Out[30]: Survived    0
Pclass      0
Name        0
Sex         0
Age         177
SibSp       0
Parch       0
Ticket      0
Fare        0
Cabin       687
Embarked    2
dtype: int64
```

```
In [31]: # count the missing value in the dataframe (method 1)
df.isnull().sum().sum()
```

Out[31]: 866

```
In [32]: # count the missing value in the dataframe (method 2)
df.isnull().values.sum()
```

Out[32]: 866

```
In [38]: # get all rows with any mssing value in columns (if any)
df[df.isnull().any(axis=1)]
```

Out[38]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cat
PassengerId										
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	N.
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	N.
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	N.
6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	N.
8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	N.
...
885	0	3	Sutehall, Mr. Henry Jr	male	25.0	0	0	SOTON/OQ 392076	7.0500	N.
886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.1250	N.
887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	N.
889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	N.
891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	N.

708 rows × 11 columns



Deleteing rows and columns

```
In [235]: # get all the column names
df.columns.values
```

```
Out[235]: array(['Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch',
                'Ticket', 'Fare', 'Cabin', 'Embarked'], dtype=object)
```

```
In [236]: # delete the Carbin column
df = df.drop('Cabin', axis = 1)    #axis = 1 for columns
df.head()
```

```
Out[236]:
```

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
PassengerId										
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	female	38.0	1	0	PC 17599	71.2833	
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

```
In [237]: # delete the row with index 1
df = df.drop(1, axis = 0) #axis = 0 for rows
df.head()
```

Out[237]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
PassengerId										
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	female	38.0	1	0	PC 17599	71.2833	
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	
6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	


```
In [238]: # delete the rows where at least one element is missing
df = df.dropna()
df.head()
```

Out[238]:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...)	female	38.0	1	0	PC 17599	71.2833	
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	
7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	

```
In [239]: # check if any missing values in the dataframe
df.isnull().values.any()
```

Out[239]: False

```
In [240]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 711 entries, 2 to 891
Data columns (total 10 columns):
Survived    711 non-null int64
Pclass      711 non-null int64
Name        711 non-null object
Sex         711 non-null object
Age         711 non-null float64
SibSp       711 non-null int64
Parch       711 non-null int64
Ticket      711 non-null object
Fare        711 non-null float64
Embarked    711 non-null object
dtypes: float64(2), int64(4), object(4)
memory usage: 61.1+ KB
```

Selecting and Manipulating Data

Selecting columns

There are three main methods of selecting columns:

- using a dot notation, e.g. `df.column_name`
- using square braces and the name of the column as a string, e.g. `df['column_name']`
- using number indexing and the `iloc` selector, e.g. `df.iloc[:, column_number]`

```
In [241]: # get all the column names
df.columns.values
```

```
Out[241]: array(['Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp', 'Parch',
                'Ticket', 'Fare', 'Embarked'], dtype=object)
```

```
In [242]: # get the Name column (method 1)
df.Name.head()
```

```
Out[242]: PassengerId
2    Cumings, Mrs. John Bradley (Florence Briggs Th...
3                                Heikkinen, Miss. Laina
4    Futrelle, Mrs. Jacques Heath (Lily May Peel)
5                                Allen, Mr. William Henry
7                                McCarthy, Mr. Timothy J
Name: Name, dtype: object
```

```
In [243]: # get the Name column (method 2)
df['Name'].head()
```

```
Out[243]: PassengerId
2    Cumings, Mrs. John Bradley (Florence Briggs Th...
3                                Heikkinen, Miss. Laina
4    Futrelle, Mrs. Jacques Heath (Lily May Peel)
5                                Allen, Mr. William Henry
7                                McCarthy, Mr. Timothy J
Name: Name, dtype: object
```

```
In [244]: # get the Name column (method 3)
df.iloc[:, 2].head() #selecting all rows in column 2, 2 is the column number
of the Name column
```

```
Out[244]: PassengerId
2    Cumings, Mrs. John Bradley (Florence Briggs Th...
3                                Heikkinen, Miss. Laina
4    Futrelle, Mrs. Jacques Heath (Lily May Peel)
5                                Allen, Mr. William Henry
7                                McCarthy, Mr. Timothy J
Name: Name, dtype: object
```

```
In [245]: # get the Name column (method 3)
df.iloc[:, [2]].head() #return a dataframe if the input column number is a list
```

Out[245]:

	Name
PassengerId	
2	Cumings, Mrs. John Bradley (Florence Briggs Th...
3	Heikkinen, Miss. Laina
4	Futrelle, Mrs. Jacques Heath (Lily May Peel)
5	Allen, Mr. William Henry
7	McCarthy, Mr. Timothy J

```
In [246]: # get multiple columns
# using square-brace selection with a list of column names
df[['Sex', 'Age', 'Ticket']].head()
```

Out[246]:

	Sex	Age	Ticket
PassengerId			
2	female	38.0	PC 17599
3	female	26.0	STON/O2. 3101282
4	female	35.0	113803
5	male	35.0	373450
7	male	54.0	17463

```
In [247]: # get multiple columns
# using numeric indexing with the iloc selector and a list of column numbers
df.iloc[:, [3, 4, 7]].head()
```

Out[247]:

	Sex	Age	Ticket
PassengerId			
2	female	38.0	PC 17599
3	female	26.0	STON/O2. 3101282
4	female	35.0	113803
5	male	35.0	373450
7	male	54.0	17463

```
In [248]: # get multiple columns
# using numeric indexing with the iloc selector and a list of column numbers
df.iloc[:, 4:8].head() # equivalent to df.iloc[:, [4,5,6,7,]].head()
```

Out[248]:

	Age	SibSp	Parch	Ticket
PassengerId				
2	38.0	1	0	PC 17599
3	26.0	0	0	STON/O2. 3101282
4	35.0	1	0	113803
5	35.0	0	0	373450
7	54.0	0	0	17463

Selecting rows

There are three main methods of selecting rows:

- using the iloc selector, e.g. `df.iloc[row_number, :]`
- using the loc selector, e.g. `df.loc[row number, :]`, but this is only applicably if "index" has been set on the dataframe
- using boolean/conditional lookup statements, e.g. `df[df['column_name'] == 'value']`

```
In [249]: # get row 5 using iloc
df.iloc[[5], :]
```

Out[249]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
PassengerId										
8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.075	S

```
In [250]: # get row 5 to 9 using iloc
df.iloc[5:10, :]
```

Out[250]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
PassengerId										
8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	
9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	
11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	
12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	

```
In [251]: # get row with index 10
df.loc[[10], :]
```

Out[251]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
PassengerId										
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	C

```
In [252]: # get row with index 10 to 19 using loc
df.loc[10:20, :]
```

Out[252]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Emb
PassengerId										
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	
11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	
12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	
13	0	3	Saunderscock, Mr. William Henry	male	20.0	0	0	A/5. 2151	8.0500	
14	0	3	Andersson, Mr. Anders Johan	male	39.0	1	5	347082	31.2750	
15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14.0	0	0	350406	7.8542	
16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55.0	0	0	248706	16.0000	
17	0	3	Rice, Master. Eugene	male	2.0	4	1	382652	29.1250	
19	0	3	Vander Planke, Mrs. Julius (Emelia Maria Vande...	female	31.0	1	0	345763	18.0000	

```
In [253]: # get all rows of female  
df[df['Sex'] == 'female']
```

Out[253]:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	
9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	
10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	
11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	
12	1	1	Bonnell, Miss. Elizabeth	female	58.0	0	0	113783	26.5500	
15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14.0	0	0	350406	7.8542	
16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55.0	0	0	248706	16.0000	
19	0	3	Vander Planke, Mrs. Julius (Emelia Maria Vande...	female	31.0	1	0	345763	18.0000	
23	1	3	McGowan, Miss. Anna "Annie"	female	15.0	0	0	330923	8.0292	
25	0	3	Palsson, Miss. Torborg Danira	female	8.0	3	1	349909	21.0750	

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
26	1	3	Asplund, Mrs. Carl Oscar (Selma Augusta Emilia...	female	38.0	1	5	347077	31.3875	
39	0	3	Vander Planke, Miss. Augusta Maria	female	18.0	2	0	345764	18.0000	
40	1	3	Nicola-Yarred, Miss. Jamila	female	14.0	1	0	2651	11.2417	
41	0	3	Ahlin, Mrs. Johan (Johanna Persdotter Larsson)	female	40.0	1	0	7546	9.4750	
42	0	2	Turpin, Mrs. William John Robert (Dorothy Ann ...	female	27.0	1	0	11668	21.0000	
44	1	2	Laroche, Miss. Simonne Marie Anne Andree	female	3.0	1	2	SC/Paris 2123	41.5792	
45	1	3	Devaney, Miss. Margaret Delia	female	19.0	0	0	330958	7.8792	
50	0	3	Arnold-Franchi, Mrs. Josef (Josefine Franchi)	female	18.0	1	0	349237	17.8000	
53	1	1	Harper, Mrs. Henry Sleeper (Myna Haxtun)	female	49.0	1	0	PC 17572	76.7292	
54	1	2	Faunthorpe, Mrs. Lizzie (Elizabeth Anne Wilkin...	female	29.0	1	0	2926	26.0000	
57	1	2	Rugg, Miss. Emily	female	21.0	0	0	C.A. 31026	10.5000	
59	1	2	West, Miss. Constance Mirium	female	5.0	1	2	C.A. 34651	27.7500	

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
67	1	2	Nye, Mrs. (Elizabeth Ramell)	female	29.0	0	0	C.A. 29395	10.5000	
69	1	3	Andersson, Miss. Erna Alexandra	female	17.0	4	2	3101281	7.9250	
72	0	3	Goodwin, Miss. Lillian Amy	female	16.0	5	2	CA 2144	46.9000	
80	1	3	Dowdell, Miss. Elizabeth	female	30.0	0	0	364516	12.4750	
85	1	2	Ilett, Miss. Bertha	female	17.0	0	0	SO/C 14885	10.5000	
86	1	3	Backstrom, Mrs. Karl Alfred (Maria Mathilda Gu...)	female	33.0	3	0	3101278	15.8500	
...
797	1	1	Leader, Dr. Alice (Farnham)	female	49.0	0	0	17465	25.9292	
798	1	3	Osman, Mrs. Mara	female	31.0	0	0	349244	8.6833	
800	0	3	Van Impe, Mrs. Jean Baptiste (Rosalie Paula Go...)	female	30.0	1	1	345773	24.1500	
802	1	2	Collyer, Mrs. Harvey (Charlotte Annie Tate)	female	31.0	1	1	C.A. 31921	26.2500	
808	0	3	Pettersson, Miss. Ellen Natalia	female	18.0	0	0	347087	7.7750	
810	1	1	Chambers, Mrs. Norman Campbell (Bertha Griggs)	female	33.0	1	0	113806	53.1000	
814	0	3	Andersson, Miss. Ebba Iris Alfrida	female	6.0	4	2	347082	31.2750	
817	0	3	Heininen, Miss. Wendla Maria	female	23.0	0	0	STON/O2. 3101290	7.9250	

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
821	1	1	Hays, Mrs. Charles Melville (Clara Jennings Gr...	female	52.0	1	1	12749	93.5000	
824	1	3	Moor, Mrs. (Beila)	female	27.0	0	1	392096	12.4750	
831	1	3	Yasbeck, Mrs. Antoni (Selini Alexander)	female	15.0	1	0	2659	14.4542	
836	1	1	Compton, Miss. Sara Rebecca	female	39.0	1	1	PC 17756	83.1583	
843	1	1	Serepeca, Miss. Augusta	female	30.0	0	0	113798	31.0000	
853	0	3	Boulos, Miss. Nourelain	female	9.0	1	1	2678	15.2458	
854	1	1	Lines, Miss. Mary Conover	female	16.0	0	1	PC 17592	39.4000	
855	0	2	Carter, Mrs. Ernest Courtenay (Lilian Hughes)	female	44.0	1	0	244252	26.0000	
856	1	3	Aks, Mrs. Sam (Leah Rosen)	female	18.0	0	1	392091	9.3500	
857	1	1	Wick, Mrs. George Dennick (Mary Hitchcock)	female	45.0	1	1	36928	164.8667	
859	1	3	Baclini, Mrs. Solomon (Latifa Qurban)	female	24.0	0	3	2666	19.2583	
863	1	1	Swift, Mrs. Frederick Joel (Margaret Welles Ba...	female	48.0	0	0	17466	25.9292	
866	1	2	Bystrom, Mrs. (Karolina)	female	42.0	0	0	236852	13.0000	
867	1	2	Duran y More, Miss. Asuncion	female	27.0	1	0	SC/PARIS 2149	13.8583	

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	En
872	1	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female	47.0	1	1	11751	52.5542	
875	1	2	Abelson, Mrs. Samuel (Hannah Wizosky)	female	28.0	1	0	P/PP 3381	24.0000	
876	1	3	Najib, Miss. Adele Kiamie "Jane"	female	15.0	0	0	2667	7.2250	
880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	female	56.0	0	1	11767	83.1583	
881	1	2	Shelley, Mrs. William (Imanita Parrish Hall)	female	25.0	0	1	230433	26.0000	
883	0	3	Dahlberg, Miss. Gerda Ulrika	female	22.0	0	0	7552	10.5167	
886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.1250	
888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	

259 rows × 10 columns



```
In [254]: # get all rows with age greater than 60  
df[df['Age'] > 60]
```

Out[254]:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Em
34	0	2	Wheadon, Mr. Edward H	male	66.0	0	0	C.A. 24579	10.5000	
55	0	1	Ostby, Mr. Engelhart Cornelius	male	65.0	0	1	113509	61.9792	
97	0	1	Goldschmidt, Mr. George B	male	71.0	0	0	PC 17754	34.6542	
117	0	3	Connors, Mr. Patrick	male	70.5	0	0	370369	7.7500	
171	0	1	Van der hoef, Mr. Wyckoff	male	61.0	0	0	111240	33.5000	
253	0	1	Stead, Mr. William Thomas	male	62.0	0	0	113514	26.5500	
276	1	1	Andrews, Miss. Kornelia Theodosia	female	63.0	1	0	13502	77.9583	
281	0	3	Duane, Mr. Frank	male	65.0	0	0	336439	7.7500	
327	0	3	Nysveen, Mr. Johan Hansen	male	61.0	0	0	345364	6.2375	
439	0	1	Fortune, Mr. Mark	male	64.0	1	4	19950	263.0000	
457	0	1	Millet, Mr. Francis Davis	male	65.0	0	0	13509	26.5500	
484	1	3	Turkula, Mrs. (Hedwig)	female	63.0	0	0	4134	9.5875	
494	0	1	Artagaveytia, Mr. Ramon	male	71.0	0	0	PC 17609	49.5042	
546	0	1	Nicholson, Mr. Arthur Ernest	male	64.0	0	0	693	26.0000	
556	0	1	Wright, Mr. George	male	62.0	0	0	113807	26.5500	
571	1	2	Harris, Mr. George	male	62.0	0	0	S.W./PP 752	10.5000	
626	0	1	Sutton, Mr. Frederick	male	61.0	0	0	36963	32.3208	
631	1	1	Barkworth, Mr. Algernon Henry Wilson	male	80.0	0	0	27042	30.0000	

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
PassengerId										
673	0	2	Mitchell, Mr. Henry Michael	male	70.0	0	0	C.A. 24580	10.5000	
746	0	1	Crosby, Capt. Edward Gifford	male	70.0	1	1	WE/P 5735	71.0000	
852	0	3	Svensson, Mr. Johan	male	74.0	0	0	347060	7.7750	

In [255]: *# get all rows of female with age greater than 60*
`df[(df['Sex']=='female') & (df['Age'] > 60)]`

Out[255]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
PassengerId										
276	1	1	Andrews, Miss. Kornelia Theodosia	female	63.0	1	0	13502	77.9583	
484	1	3	Turkula, Mrs. (Hedwig)	female	63.0	0	0	4134	9.5875	

In [256]: `# get all rows with age less than 1 and age greater than 70`
`df[(df['Age'] < 1) | (df['Age'] > 70)]`

Out[256]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Err
PassengerId										
79	1	2	Caldwell, Master. Alden Gates	male	0.83	0	2	248738	29.0000	
97	0	1	Goldschmidt, Mr. George B	male	71.00	0	0	PC 17754	34.6542	
117	0	3	Connors, Mr. Patrick	male	70.50	0	0	370369	7.7500	
306	1	1	Allison, Master. Hudson Trevor	male	0.92	1	2	113781	151.5500	
470	1	3	Bacini, Miss. Helene Barbara	female	0.75	2	1	2666	19.2583	
494	0	1	Artagaveytia, Mr. Ramon	male	71.00	0	0	PC 17609	49.5042	
631	1	1	Barkworth, Mr. Algernon Henry Wilson	male	80.00	0	0	27042	30.0000	
645	1	3	Bacini, Miss. Eugenie	female	0.75	2	1	2666	19.2583	
756	1	2	Hamalainen, Master. Viljo	male	0.67	1	1	250649	14.5000	
804	1	3	Thomas, Master. Assad Alexander	male	0.42	0	1	2625	8.5167	
832	1	2	Richards, Master. George Sibley	male	0.83	1	1	29106	18.7500	
852	0	3	Svensson, Mr. Johan	male	74.00	0	0	347060	7.7750	



In [257]: `# get all rows with missing values (if any)`
`df[df.isnull().any(axis=1)]`

Out[257]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
PassengerId										

Summarising Groups in the DataFrame

```
In [258]: # Number of non-null unique Pclass values
df['Pclass'].nunique()
```

Out[258]: 3

```
In [259]: # What are the unique values for Pclass
df['Pclass'].unique()
```

Out[259]: array([1, 3, 2], dtype=int64)

```
In [260]: # How many samples are there for each Pclass?

# Method 1
# get the Pclass column and count each value in the column
df['Pclass'].value_counts()
```

Out[260]: 3 354
1 184
2 173
Name: Pclass, dtype: int64

```
In [261]: # How many samples are there for each Pclass?

# Method 2
# get the Pclass column and another column for counting (Name in this case)
# group the data by Pclass
# count the PassengerId in each group of Pclass
df[['Pclass', 'Name']].groupby('Pclass').count()
```

Out[261]:

	Name
Pclass	
1	184
2	173
3	354

```
In [262]: # How many male and female passengers?

# get the Sex column and count each value in the column
df['Sex'].value_counts()
```

Out[262]: male 452
female 259
Name: Sex, dtype: int64

```
In [263]: # How many male and female passengers in each Pclass?

# 1. group the data by Pclass
# 2. count each value in the Sex column (in each Pclass group)
df.groupby('Pclass')['Sex'].value_counts()
```

```
Out[263]: Pclass  Sex
1         male    101
          female    83
2         male     99
          female    74
3         male   252
          female   102
Name: Sex, dtype: int64
```

```
In [264]: # the average age in each Pclass

# Method 1
# 1. group the data by Pclass
# 2. find the average value of Age column (in each Pclass group)
df.groupby('Pclass')['Age'].mean()
```

```
Out[264]: Pclass
1    38.105543
2    29.877630
3    25.149492
Name: Age, dtype: float64
```

```
In [265]: # the average age in each Pclass

# Method 2
# 1. get the Pclass and Age columns
# 2. group the data by Pclass
# 2. find the average value of Age column (in each Pclass group)
df[['Pclass', 'Age']].groupby('Pclass').mean()
```

```
Out[265]:
```

	Age
Pclass	
1	38.105543
2	29.877630
3	25.149492

```
In [266]: # the average age of each gender group in each Pclass
df[['Pclass', 'Age', 'Sex']].groupby(['Pclass', 'Sex']).mean()
```

Out[266]:

Age		
Pclass	Sex	
1	female	34.240964
	male	41.281386
2	female	28.722973
	male	30.740707
3	female	21.750000
	male	26.525476

Here is a quick reference summary table of some common functions.

Function	Description
count	Number of non-NA observations
sum	Sum of values
mean	Mean of values
min	Minimum
max	Maximum
mode	Mode
abs	Absolute Value
var	Unbiased variance
std	Bessel-corrected sample standard deviation
quantile	Sample quantile (value at %)
median	Arithmetic median of values

image source: https://pandas.pydata.org/pandas-docs/stable/getting_started/basics.html

Visualizing the data

Matplotlib Magic Function

Run the following magic function to allow displaying the plot within the cell

```
In [267]: %matplotlib inline
```

1. The Pandas Plot Function

Pandas has a built in `.plot()` function as part of the `DataFrame` class. It has several key parameters:

- `kind`: 'line', 'bar', 'pie', 'scatter', etc.
- `title`: The string title of the plot
- `legend`: a boolean value to display or hide the legend
- `figsize`: a tuple (width, height) in inches

See <https://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.plot.html> (<https://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.plot.html>) for more information.

The major use cases for `.plot()` is when you have a meaningful index, which usually happens in two situations:

- You have just done a `.value_counts()` or a `.groupby()`
- You have used `.set_index`, probably with dates

These are fairly straightforward to use. Let us do some examples using `.plot()`.

Bar charts

Plot the no. of passengers in each gender group

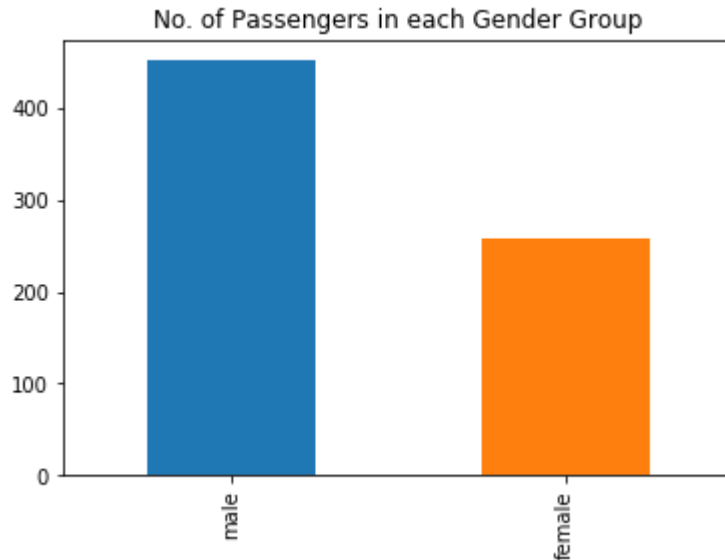
```
In [268]: df['Sex'].value_counts()
```

```
Out[268]: male      452  
female    259  
Name: Sex, dtype: int64
```

```
In [269]: # Plot the no. of passengers in each gender group

df['Sex'].value_counts().plot(kind='bar',
                               title='No. of Passengers in each Gender Group',
                               figsize=(6, 4))
```

Out[269]: <matplotlib.axes._subplots.AxesSubplot at 0x2427f3fa668>



Plot the no. of passengers in each pclass

```
In [270]: df[['Pclass', 'Name']].groupby('Pclass').count()
```

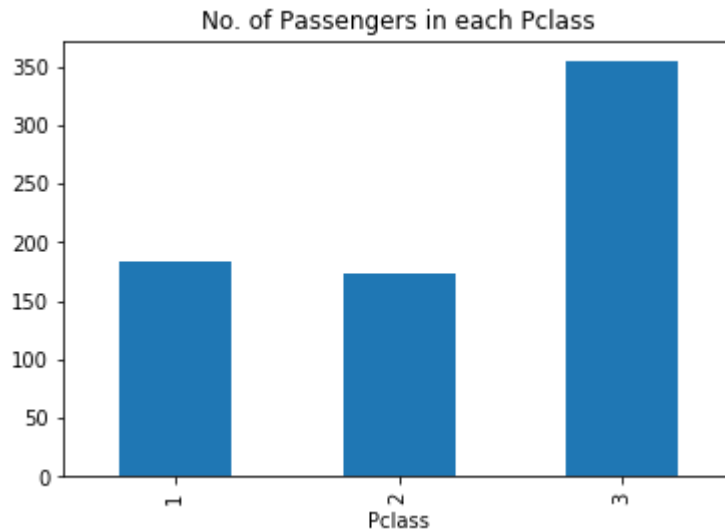
Out[270]:

Name	
Pclass	
1	184
2	173
3	354

```
In [271]: # plot the no. of passengers in each pclass

df[['Pclass', 'Name']].groupby('Pclass').count().plot(kind='bar',
                                                    title='No. of Pas
sengers in each Pclass',
                                                    legend=False,
                                                    figsize=(6, 4))
```

Out[271]: <matplotlib.axes._subplots.AxesSubplot at 0x2427f3c6a90>



Plot the mean age of each pclass

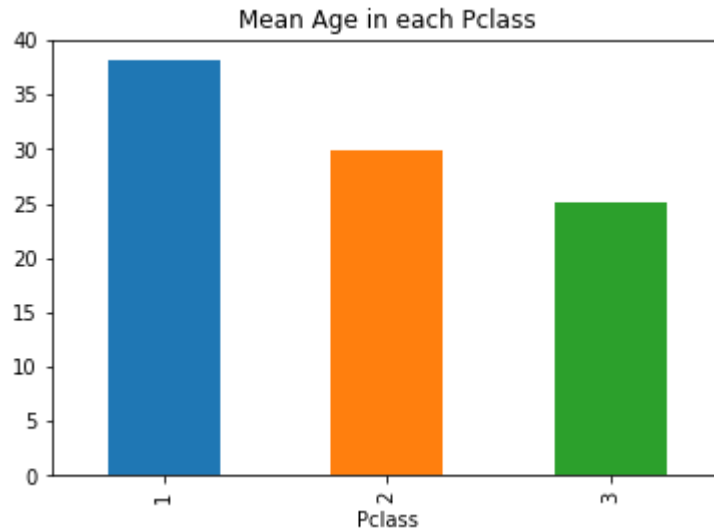
```
In [272]: df.groupby('Pclass')['Age'].mean()
```

Out[272]: Pclass
1 38.105543
2 29.877630
3 25.149492
Name: Age, dtype: float64

```
In [273]: # plot the mean age of each pclass

df.groupby('Pclass')['Age'].mean().plot(kind='bar',
                                         title='Mean Age in each Pclass',
                                         figsize=(6, 4))
```

Out[273]: <matplotlib.axes._subplots.AxesSubplot at 0x2427f620b70>



Plot the no. of survived and dead

```
In [274]: df['Survived'].value_counts()
```

Out[274]: 0 423
1 288
Name: Survived, dtype: int64

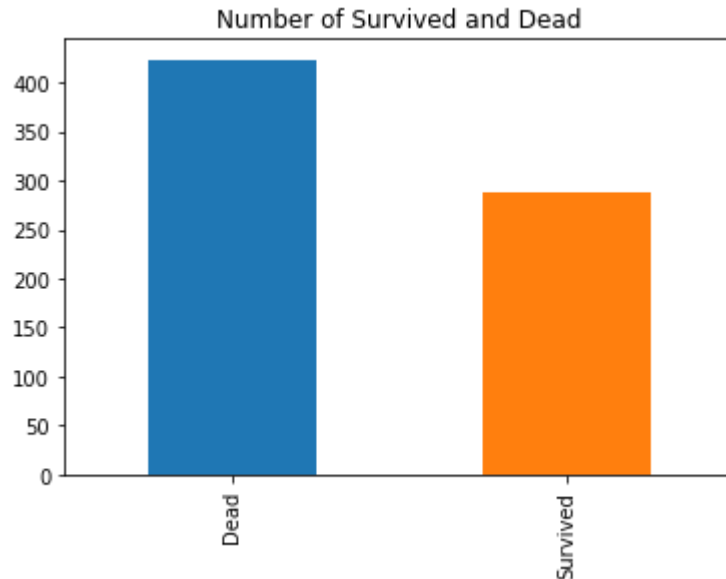
```
In [275]: new_df = df['Survived'] # create a new dataframe with the Survived column
new_df = new_df.replace({0: 'Dead', 1: 'Survived'}) # relace 0 with 'Dead', 1
with 'Survived'
new_df.value_counts()
```

Out[275]: Dead 423
Survived 288
Name: Survived, dtype: int64

```
In [276]: # plot the no. of Survived and Dead

new_df.value_counts().plot(kind = 'bar',
                             title='Number of Survived and Dead',
                             figsize=(6, 4))
```

Out[276]: <matplotlib.axes._subplots.AxesSubplot at 0x2427f62f320>



Line charts

Plot the average age vs no. of siblings and spouses traveling with the passenger

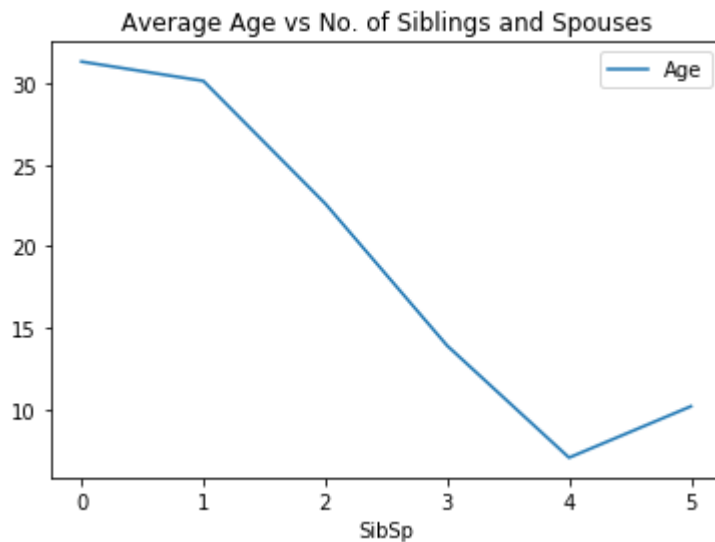
```
In [277]: df[['Age', 'SibSp']].groupby(['SibSp']).mean()
```

Out[277]:

Age	
SibSp	
0	31.318230
1	30.134176
2	22.620000
3	13.916667
4	7.055556
5	10.200000


```
In [278]: # plot the average age vs no. of siblings and spouses traveling with the passenger
df[['Age', 'SibSp']].groupby(['SibSp']).mean().plot(kind = 'line',
                                                    title='Average Age vs No.
                                                    of Siblings and Spouses',
                                                    figsize=(6, 4))
```

Out[278]: <matplotlib.axes._subplots.AxesSubplot at 0x2427f6be198>



Plot more than one line in the chart

```
In [279]: # average age and average fare for each no. of siblings and spouses
df[['Age', 'SibSp', 'Fare']].groupby(['SibSp']).mean()
```

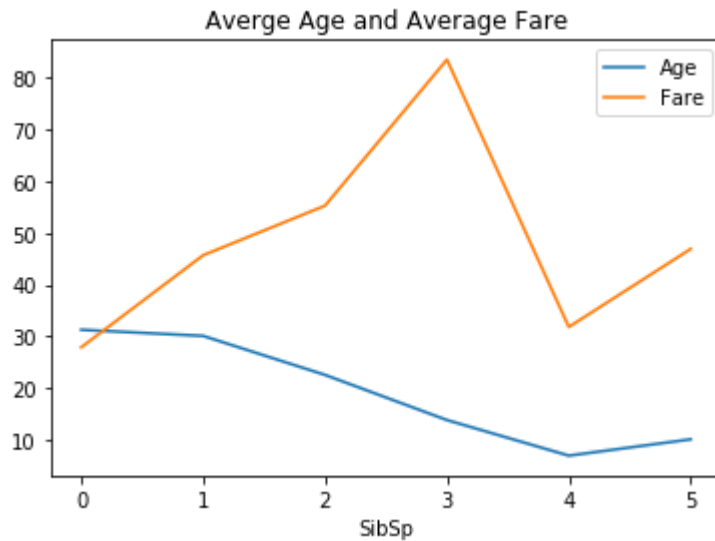
Out[279]:

	Age	Fare
SibSp		
0	31.318230	27.934620
1	30.134176	45.680199
2	22.620000	55.236996
3	13.916667	83.389583
4	7.055556	31.855556
5	10.200000	46.900000

```
In [280]: # plot the
# - average age vs no. of siblings and spouses traveling with the passenger
# - average fare vs no. of siblings and spouses traveling with the passenger

df[['Age', 'SibSp', 'Fare']].groupby(['SibSp']).mean().plot(kind = 'line',
                                                             title='Average Age
and Average Fare',
                                                             figsize=(6, 4))
```

Out[280]: <matplotlib.axes._subplots.AxesSubplot at 0x2427f739518>



Scatterplots

Scatterplot of Fare vs Age

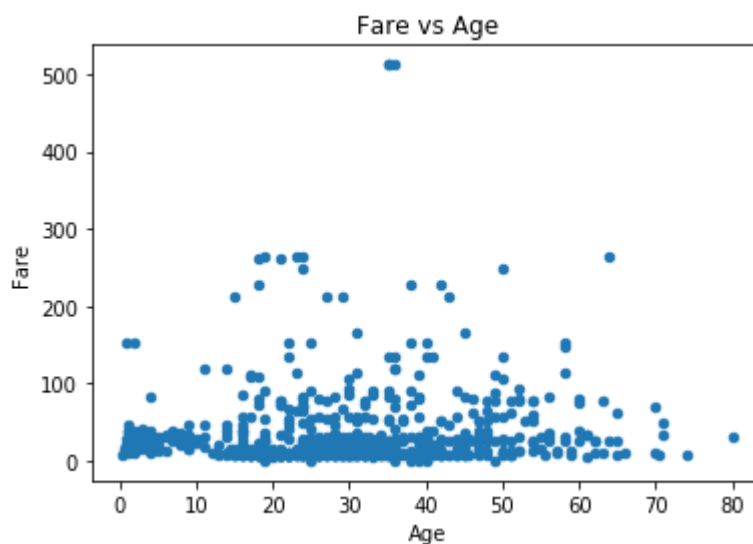
In [281]: `df.head()`

Out[281]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked
PassengerId										
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	
7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	

In [282]: `# Scatterplot of Fare vs Age`
`df.plot(x='Age',`
`y='Fare',`
`kind = 'scatter',`
`title='Fare vs Age',`
`figsize=(6, 4))`

Out[282]: `<matplotlib.axes._subplots.AxesSubplot at 0x2427f78b198>`

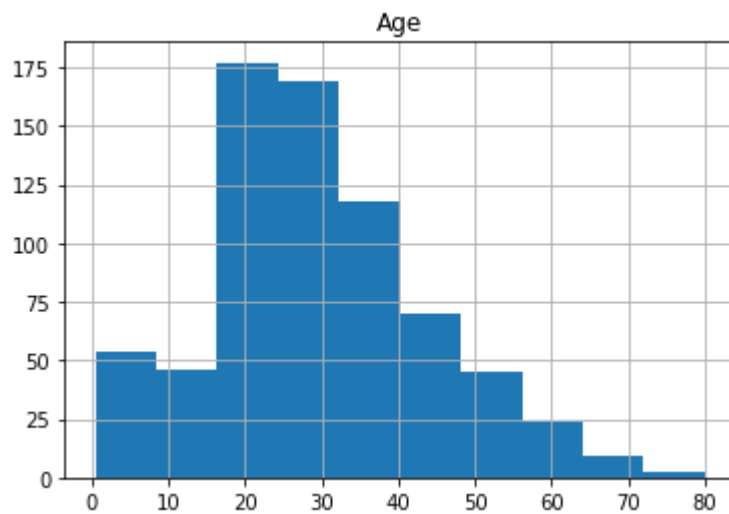


Histogram

Plot the age distribution of all the passengers

```
In [56]: df.hist(column='Age', sharex=True, sharey=True, bins=10)
```

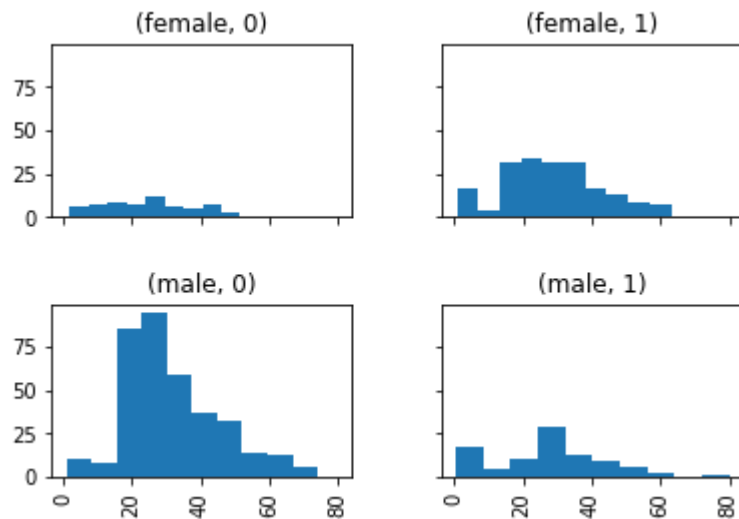
```
Out[56]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000015461513CA0  
>]],  
          dtype=object)
```



Plot the age distribution of different groups of passengers (Sex and Survived)

```
In [62]: df.hist(column='Age', by=['Sex', 'Survived'], sharex=True, sharey=True, bins=10)
```

```
Out[62]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000015462E5D970>,  
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000015462E82D00  
  >],  
  [<matplotlib.axes._subplots.AxesSubplot object at 0x0000015462EA6D00>,  
  <matplotlib.axes._subplots.AxesSubplot object at 0x0000015462ED2BB0  
  >]],  
  dtype=object)
```



References

Dolan, H. (2019, March 17). A Guide to Pandas and Matplotlib for Data Exploration. Retrieved from <https://towardsdatascience.com/a-guide-to-pandas-and-matplotlib-for-data-exploration-56fad95f951c>
(<https://towardsdatascience.com/a-guide-to-pandas-and-matplotlib-for-data-exploration-56fad95f951c>)

Essential basic functionality — pandas 0.25.0 documentation. (n.d.). Retrieved from https://pandas.pydata.org/pandas-docs/stable/getting_started/basics.html (https://pandas.pydata.org/pandas-docs/stable/getting_started/basics.html)

Jupyter Notebook Keyboard Shortcuts. (2016, March 21). Retrieved from <https://www.cheatography.com/weidadeyue/cheat-sheets/jupyter-notebook/>
(<https://www.cheatography.com/weidadeyue/cheat-sheets/jupyter-notebook/>)

Missing Data In pandas Dataframes. (2017, December 20). Retrieved from https://chrisalbon.com/python/data_wrangling/pandas_missing_data/
(https://chrisalbon.com/python/data_wrangling/pandas_missing_data/)

Moffitt, C. (n.d.). Effectively Using Matplotlib - Practical Business Python. Retrieved from <https://pbpython.com/effective-matplotlib.html> (<https://pbpython.com/effective-matplotlib.html>)

Notebook Basics — Jupyter Notebook 6.0.0 documentation. (n.d.). Retrieved from <https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Notebook%20Basics.html> (<https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Notebook%20Basics.html>)

Sharma, M. (2018, November 3). Data Visualization using Seaborn. Retrieved from <https://towardsdatascience.com/data-visualization-using-seaborn-fc24db95a850>
(<https://towardsdatascience.com/data-visualization-using-seaborn-fc24db95a850>)

Understand df.plot in pandas. (n.d.). Retrieved from <http://jonathansoma.com/lede/algorithms-2017/classes/fuzziness-matplotlib/understand-df-plot-in-pandas/> (<http://jonathansoma.com/lede/algorithms-2017/classes/fuzziness-matplotlib/understand-df-plot-in-pandas/>)

Using iloc, loc, & ix to select rows and columns in Pandas DataFrames. (2017, July 9). Retrieved from <https://www.shanelynn.ie/select-pandas-dataframe-rows-and-columns-using-iloc-loc-and-ix/>
(<https://www.shanelynn.ie/select-pandas-dataframe-rows-and-columns-using-iloc-loc-and-ix/>)