PRACTICAL : 1

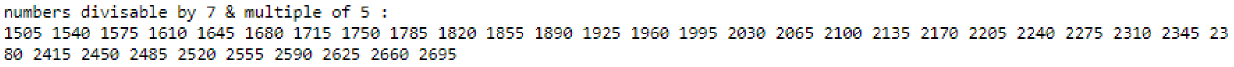
Aim : Develop Programs To Understand The Control Structures, Branching Programs , Strings and Input Of Python and functions.

1.1 : Write a Python Program to find those numbers which are divisible by 7 and multiple of 5, between 1500 and 2700.

Program:

```
print("numbers divisable by 7 & multiple of 5 :")
for i in range(1500,2701):
    if(i%7==0):
        if(i%5==0):
            print(i,end=" ")
```

Output:



```
numbers divisable by 7 & multiple of 5 :
1505 1540 1575 1610 1645 1680 1715 1750 1785 1820 1855 1890 1925 1960 1995 2030 2065 2100 2135 2170 2205 2240 2275 2310 2345 23
80 2415 2450 2485 2520 2555 2590 2625 2660 2695
```

1.2 : Write a Python Progran to construct the following pattern, using nested for loop.

```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

Program :

```
for i in range(1,6):
    for j in range(1,i+1):
        print("*",end=" ")
    print()
for i in range(4,0,-1):
    for j in range(1,i+1):
        print("*",end=" ")
    print()
```

Output:

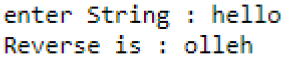
```
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

1.3 Write a Python program that accepts a word from user and reverse it (without using the reverse function)

Program:

```
str=input("enter String : ")
l=len(str)
print("Reverse is : ",end="")
for i in range(l-1,-1,-1):
    print(str[i],end="")
```

Output:



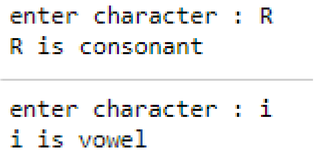
```
enter String : hello
Reverse is : olleh
```

1.4 : Write a Python program to check whether an alphabet is a vowel or consonant.

Program:

```
lst=['a','e','i','o','u']
ch=input("enter character : ")
char=ch.lower()
if char in lst:
    print(ch,"is vowel")
else:
    print(ch,"is consonant")
```

Output:



```
enter character : R
R is consonant
```

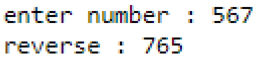
```
enter character : i
i is vowel
```

1.5 Write a Python program to find reverse of given number using user defined function.

Program:

```
def reverse(n):
    s=0
    while(n!=0):
        rem=n%10
        s=s*10+rem
        n=n//10
    print("reverse :",s)

n=int(input("enter number : "))
reverse(n)
```

Output:



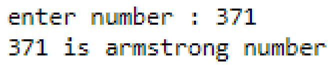
```
enter number : 567
reverse : 765
```

1.6 Write a Python program to check whether the given no is Armstrong or not using user defined function.

Program:

```
def armstrong(n):
    num=n
    s=0
    while(n>0):
        rem=n%10
        s=s+(rem*rem*rem)
        n=n//10
    if(num==s):
        print(num,"is armstrong number")
    else:
        print(num,"is not armstrong number")

a=int(input("enter number : "))
armstrong(a)
```

Output:



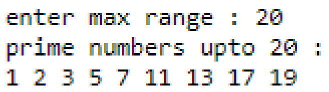
```
enter number : 371
371 is armstrong number
```

1.7 : To write a Python program to find first n prime numbers.

Program:

```
n=int(input("enter max range : "))
print("prime numbers upto 20 : ")
for i in range(1,n+1):
    for j in range(2,i):
        if((i%j)==0):
            break
    else:
        print(i,end=" ")
```

Output:



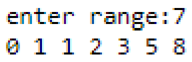
```
enter max range : 20
prime numbers upto 20 :
1 2 3 5 7 11 13 17 19
```

1.8 Write a Python program to print Fibonacci series upto n terms.

Program:

```
r=int(input("enter range:"))
a=0
b=1
print(a,end=" ")
print(b,end=" ")
for i in range(2,r):
	c=a+b
	print(c,end=" ")
	a=b
	b=c
```

Output:



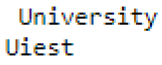
```
enter range:7
0 1 1 2 3 5 8
```

1.9 Give the output of following Python code:

a) myStr="GTU is the best University"

```
print(myStr[15::1])
print(myStr[-10:-1:2])
```

Output:

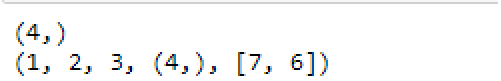


```
 University
Uiest
```

b) t=(1,2,3,(4,),[5,6])

```
print(t[3])
t[4][0]=7
print(t)
```

Output:

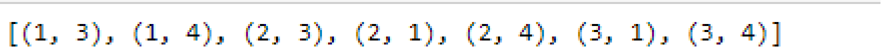


```
(4,)
(1, 2, 3, (4,), [7, 6])
```

c) I=[(x,y) for x in [1,2,3] for y in [3,1,4] if x != y]

print(I)

Output:

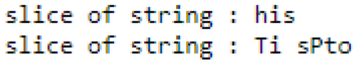


```
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
```

d) str1="This is Python"

```
print("slice of string :",str1[1:4:1])
print("slice of string :",str1[0:-1:2])
```

Output:



```
slice of string : his
slice of string : Ti sPto
```

PRACTICAL : 2

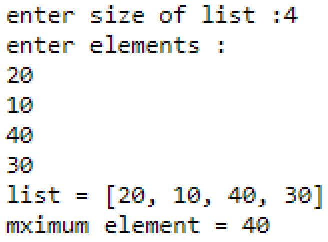
Aim : Develop programs to learn different types of structures (list, dictionary, tuples) in python.

2.1 : To write a Python Program to find the maximum from a list of numbers.

Program :

```
n=int(input("enter size of list :"))
lst=[]
print("enter elements :")
for i in range(0,n):
    num=int(input())
    lst.append(num)
print("list =",lst)

max=lst[0]
for i in lst:
    if max<i:
        max=i

print("mximum element =",max)
```

Output :



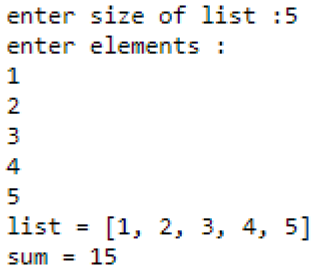
```
enter size of list :4
enter elements :
20
10
40
30
list = [20, 10, 40, 30]
mximum element = 40
```

2.2 : Write a Python program which will return the sum of the numbers in the array, returning 0 for an empty array. Except the number 13 is very unlucky, so it does not count and number that come immediately after 13 also do not count. Example : [1, 2, 3, 4] = 10 [1, 2, 3, 4, 13] = 10 [13, 1, 2, 3, 13] = 5

Program :

```
n=int(input("enter size of list :"))
lst=[]
print("enter elements :")
for i in range(0,n):
    num=int(input())
    lst.append(num)

print("list =",lst)
sum=0
for i in lst:
    if i==13:
        s=lst.remove(13);
    else:
        sum=sum+i
print('sum =',sum)
```

Output :



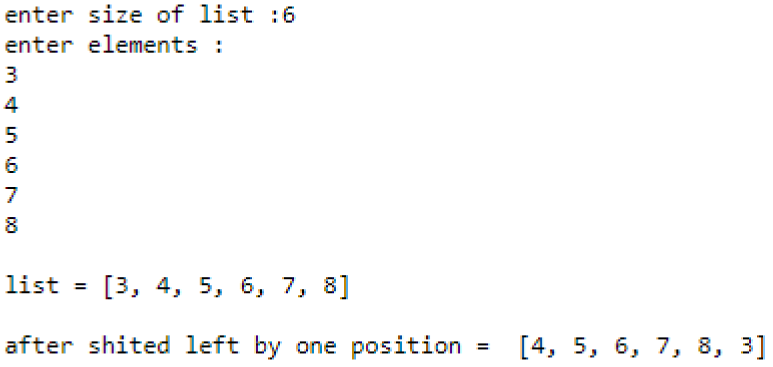
```
enter size of list :5
enter elements :
1
2
3
4
5
list = [1, 2, 3, 4, 5]
sum = 15
```

2.3 : Write a Python program which takes a list and returns a list with the elements "shifted left by one position" so [1, 2, 3] yields [2, 3, 1]. Example: [1, 2, 3] → [2, 3, 1] [11, 12, 13] → [12, 13, 11]

Program :

```
n=int(input("enter size of list :"))
lst=[]
print("enter elements : ")
for i in range(0,n):
   num=int(input())
   lst.append(num)
print()
print("list =",lst)

a=len(lst)
s=lst[0]

for i in range(0,len(lst)):
   lst[i]=lst[i]+1

lst.insert(a-1,s)
lst.pop(a)
print()
print('after shited left by one position = ',lst)
```

Output :



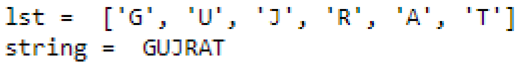
```
enter size of list :6
enter elements :
3
4
5
6
7
8

list = [3, 4, 5, 6, 7, 8]

after shited left by one position =  [4, 5, 6, 7, 8, 3]
```

2.4 : Write a program to convert a list of characters into a string

Program :

```
lst=['G','U','J','R','A','T']
print('lst = ',lst)
lst1=""
lst1=lst1.join(lst)
print('string = ',lst1)
```

Output :



```
lst =  ['G', 'U', 'J', 'R', 'A', 'T']
string =  GUJRAT
```

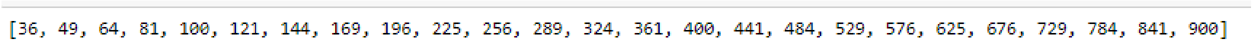
2.5 Write a Python program

1) To generate a list except for the first 5 elements, where the values are square of numbers between 1 and 30(both included)

Program :

```
lst=[i*i for i in range(1,31)]
print(lst[5:])
```

Output :



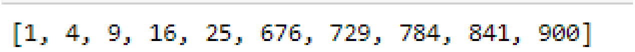
```
[36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400, 441, 484, 529, 576, 625, 676, 729, 784, 841, 900]
```

2) To generate a list of first and last 5 elements where the values are square of numbers btween 1 and 30.

Program :

```
lst=[i*i for i in range(1,31)]
lst1=lst[0:5]
lst2=lst[25:]
print(lst1+lst2)
```

Output :



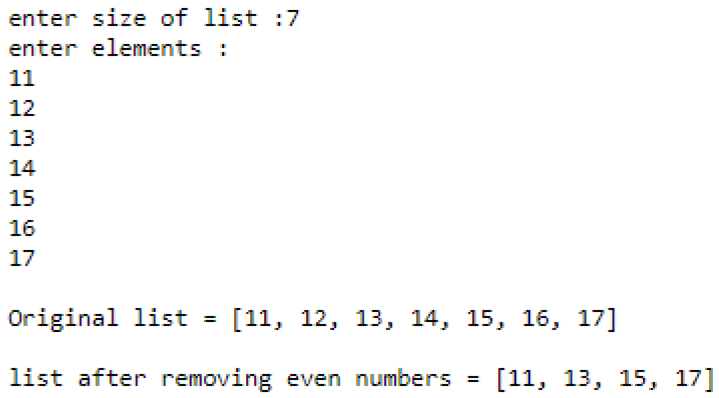
```
[1, 4, 9, 16, 25, 676, 729, 784, 841, 900]
```

2.6 Write a python program to print numbers given in the list after removing even numbers from it.

Program :

```
n=int(input("enter size of list :"))
lst=[]
print("enter elements :")
for i in range(0,n):
    num=int(input())
    lst.append(num)
print()
print('Original list =',lst)
print()

lst1=[]
for i in lst:
    if(i%2)==0:
        lst.remove(i)

print('list after removing even numbers =',lst)
```

Output :



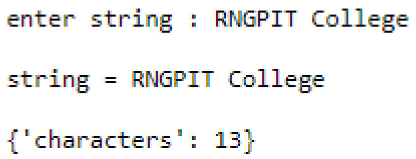
```
enter size of list :7
enter elements :
11
12
13
14
15
16
17

Original list = [11, 12, 13, 14, 15, 16, 17]

list after removing even numbers = [11, 13, 15, 17]
```

2.7 Write a program to count the numbers of characters in the string and store them in a dictionary data structure.

Program :

```
str=input("enter string : ")
print()
print('string =',str)
print()
c=0
for i in str:
    if i==' ':
        pass
    else:
        c=c+1

dict={'characters':0}
dict['characters']=c
print(dict)
```

Output :



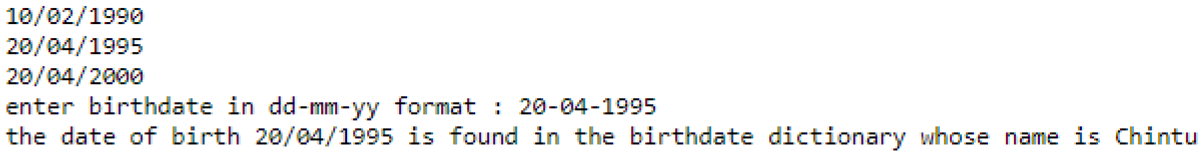
```
enter string : RNGPIT College

string = RNGPIT College

{'characters': 13}
```

2.8 Write a program to use split and join methods in the string and trace a birthday with a dictionary data structure.

Program :

```
birthdate={'Puja':'10/02/1990','Chintu':'20/04/1995','Banti':'20/04/2000'}
d=birthdate.values()
flag=0
for i in d:
    print(i)

choice=input("enter birthdate in dd-mm-yy format : ")
b=choice.split('-')
c='/'.join(b)

for name,birthdate in birthdate.items():
    if(birthdate==c):
        print("the date of birth %s is found in the birthdate dictionary whose name is %s"%(birthdate,name))
        break
else:
    print("not found")
```

Output :



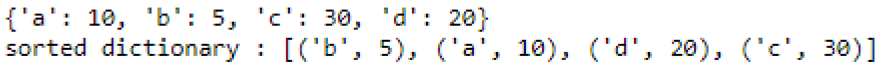
```
10/02/1990
20/04/1995
20/04/2000
enter birthdate in dd-mm-yy format : 20-04-1995
the date of birth 20/04/1995 is found in the birthdate dictionary whose name is Chintu
```

2.9 Write a python program to sort a dictionary by value.

Program :

```
dict={'a':10,'b':5,'c':30,'d':20}
print(dict)
dict_sort=sorted(dict.items(),key=lambda y: y[1])
print('sorted dictionary :',dict_sort)
```

Output :



```
{'a': 10, 'b': 5, 'c': 30, 'd': 20}
sorted dictionary : [('b', 5), ('a', 10), ('d', 20), ('c', 30)]
```

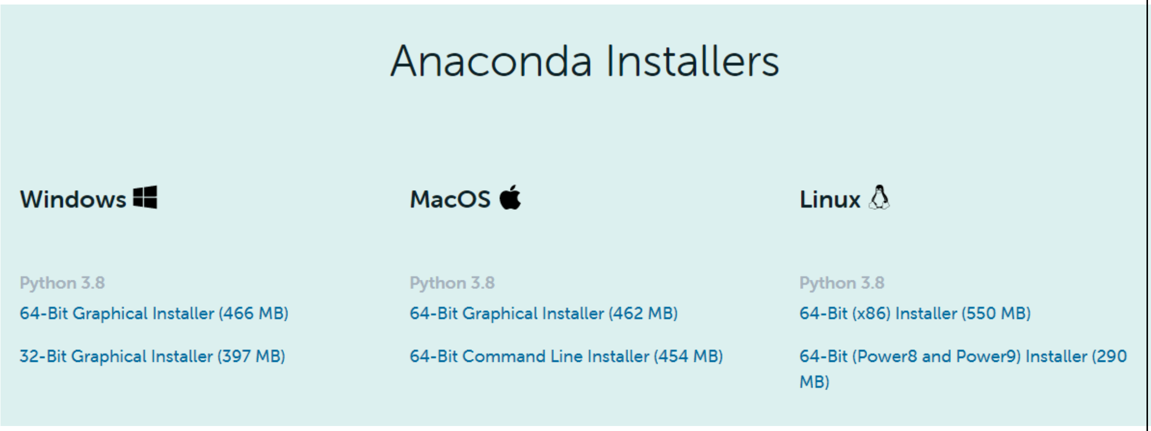
PRACTICAL: 3

Aim: Setting up Python for Data Science.

3.1 Installing Anaconda on Windows

Steps:

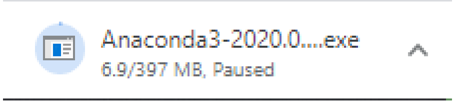
1. Go to the Anaconda website to download installer for the individual and go to ‘Anaconda Installers’.



Anaconda Installers

Windows	MacOS	Linux
Python 3.8	Python 3.8	Python 3.8
64-Bit Graphical Installer (466 MB)	64-Bit Graphical Installer (462 MB)	64-Bit (x86) Installer (550 MB)
32-Bit Graphical Installer (397 MB)	64-Bit Command Line Installer (454 MB)	64-Bit (Power8 and Power9) Installer (290 MB)

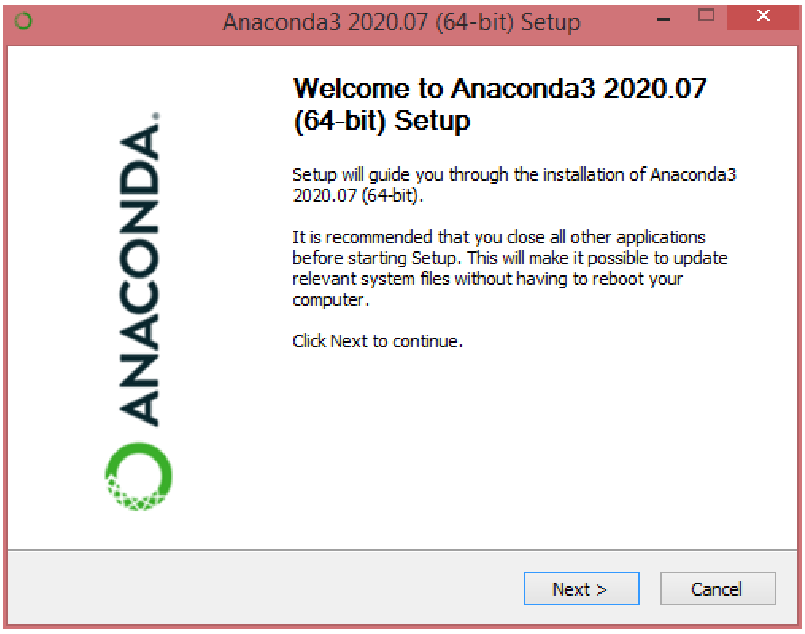
2. Select the operating system and choose the laptop/PC’s configuration (32 or 64 bit) and click on that. It will begins installation process.



Anaconda3-2020.0....exe

6.9/397 MB, Paused

3. Once download complete, open and run ‘.exe’ installer.
4. At the beginning click on ‘Next’ to confirm the installation.



Anaconda3 2020.07 (64-bit) Setup

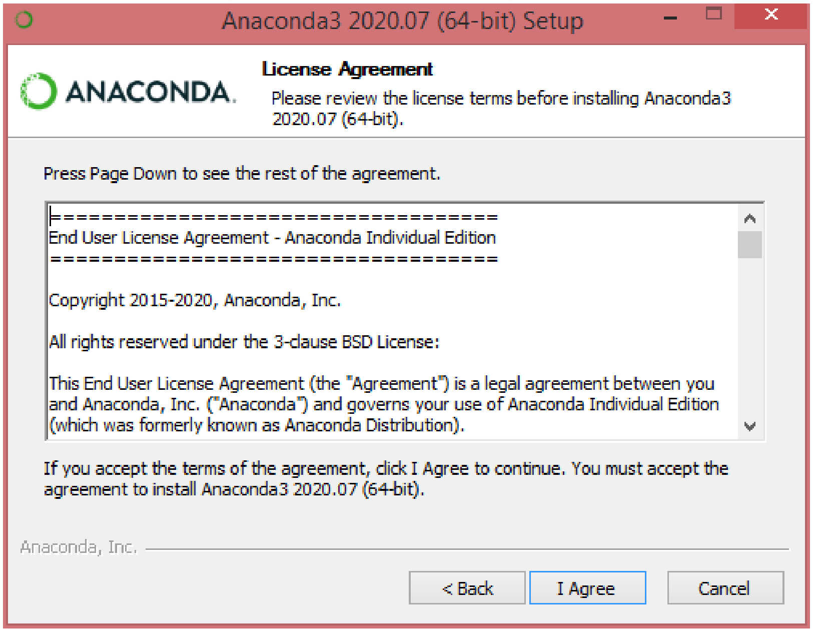
Welcome to Anaconda3 2020.07 (64-bit) Setup

Setup will guide you through the installation of Anaconda3 2020.07 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

5. Read the licensing terms and click ‘I Agree’.



License Agreement

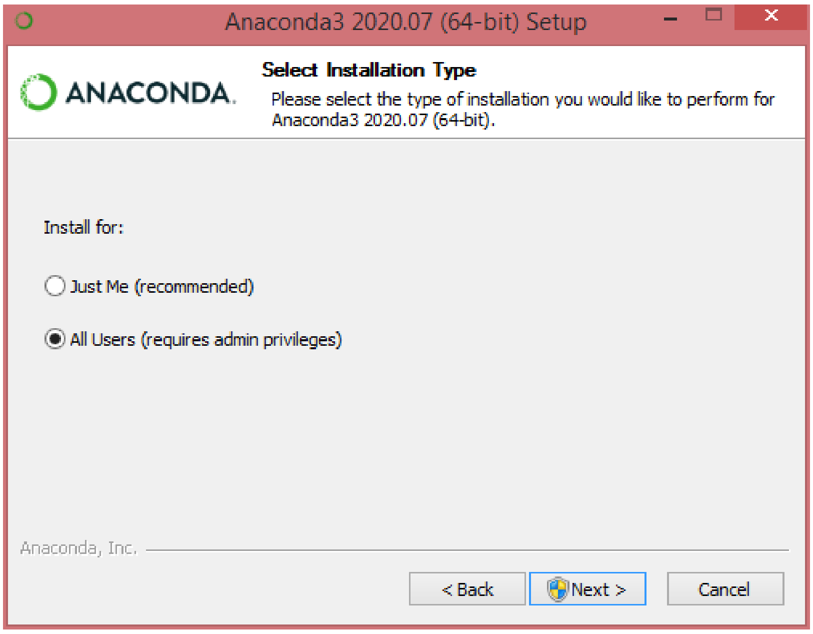
Please review the license terms before installing Anaconda3 2020.07 (64-bit).

Press Page Down to see the rest of the agreement.

```
==================================
End User License Agreement - Anaconda Individual Edition
==================================

Copyright 2015-2020, Anaconda, Inc.

All rights reserved under the 3-clause BSD License:

This End User License Agreement (the "Agreement") is a legal agreement between you
and Anaconda, Inc. ("Anaconda") and governs your use of Anaconda Individual Edition
(which was formerly known as Anaconda Distribution).
```

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Anaconda3 2020.07 (64-bit).

6. Select an install for “Just Me” unless installing for all users (which requires Windows Administrator privileges) and click ‘Next’.



Select Installation Type

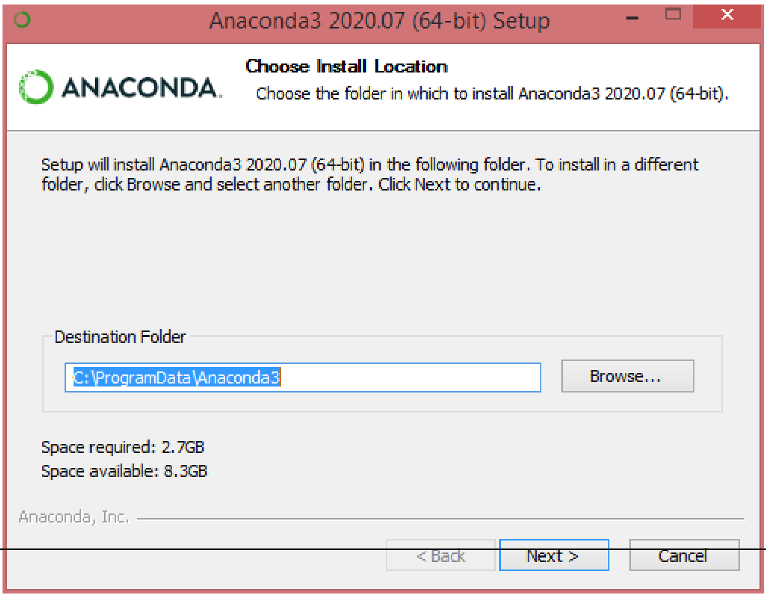
Please select the type of installation you would like to perform for Anaconda3 2020.07 (64-bit).

Install for:

○ Just Me (recommended)

● All Users (requires admin privileges)

7. Select a destination folder to install Anaconda and click the Next button.



Choose Install Location

Choose the folder in which to install Anaconda3 2020.07 (64-bit).

Setup will install Anaconda3 2020.07 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

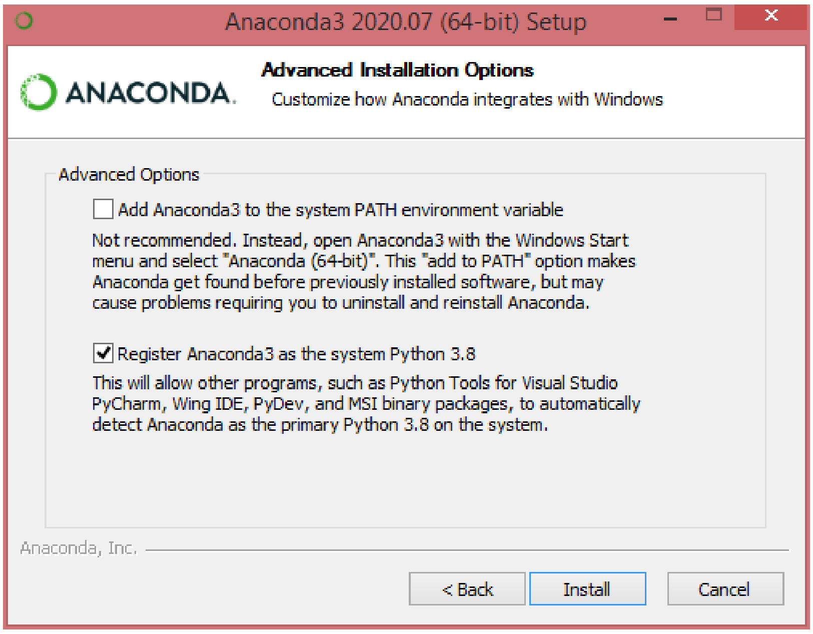
Destination Folder

C:\ProgramData\Anaconda3

Space required: 2.7GB

Space available: 8.3GB

8. Choose whether to add Anaconda to the PATH environment variable.
9. Choose whether to register Anaconda as default Python. Accept the default and leave this box checked and click on 'Install'.



Anaconda3 2020.07 (64-bit) Setup

Advanced Installation Options

Customize how Anaconda integrates with Windows

Advanced Options

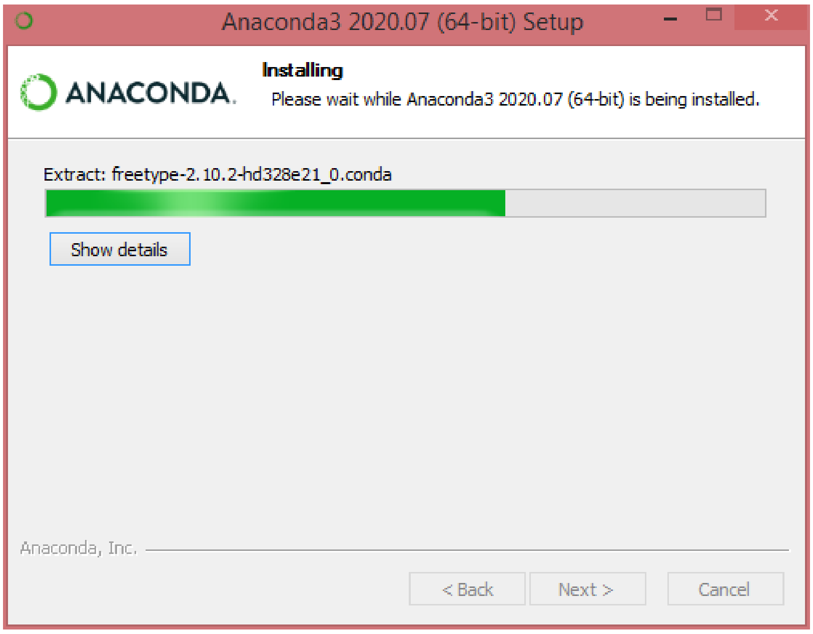
□ Add Anaconda3 to the system PATH environment variable

Not recommended. Instead, open Anaconda3 with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

☑ Register Anaconda3 as the system Python 3.8

This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.8 on the system.

Installing…..



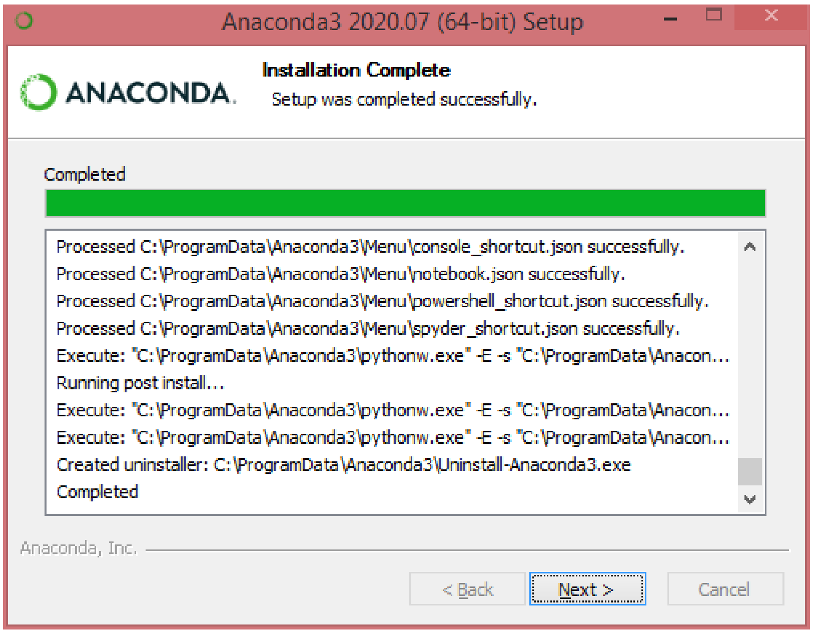
Anaconda3 2020.07 (64-bit) Setup

Installing

Please wait while Anaconda3 2020.07 (64-bit) is being installed.

Extract: freetype-2.10.2-hd328e21_0.conda

10. Click ‘next’.



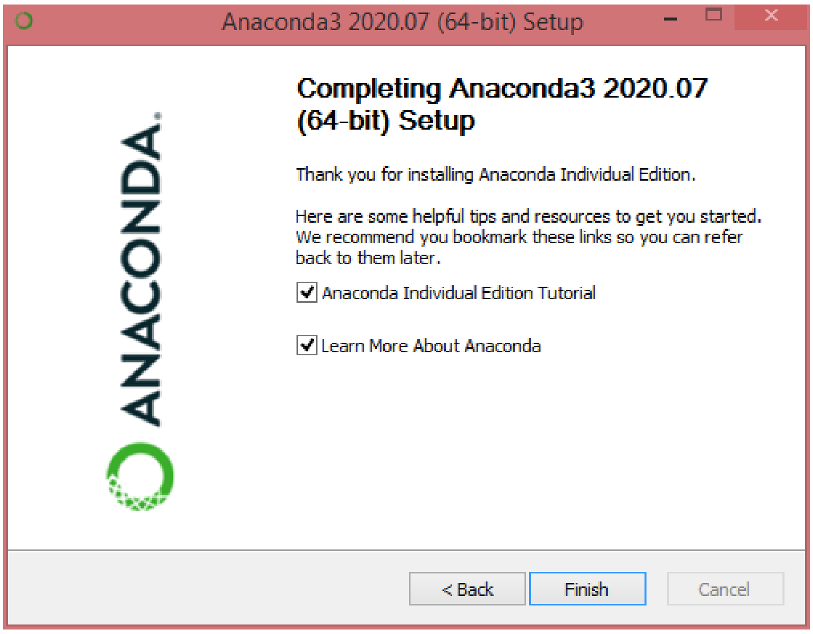
Installation Complete

Setup was completed successfully.

Completed

```
Processed C:\ProgramData\Anaconda3\Menu\console_shortcut.json successfully.
Processed C:\ProgramData\Anaconda3\Menu\notebook.json successfully.
Processed C:\ProgramData\Anaconda3\Menu\powershell_shortcut.json successfully.
Processed C:\ProgramData\Anaconda3\Menu\spyder_shortcut.json successfully.
Execute: "C:\ProgramData\Anaconda3\pythonw.exe" -E -s "C:\ProgramData\Anacon...
Running post install...
Execute: "C:\ProgramData\Anaconda3\pythonw.exe" -E -s "C:\ProgramData\Anacon...
Execute: "C:\ProgramData\Anaconda3\pythonw.exe" -E -s "C:\ProgramData\Anacon...
Created uninstaller: C:\ProgramData\Anaconda3\Uninstall-Anaconda3.exe
Completed
```

11. After a successful installation “Thanks for installing Anaconda” dialog box appear. Click on ‘Finish’.



Completing Anaconda3 2020.07 (64-bit) Setup

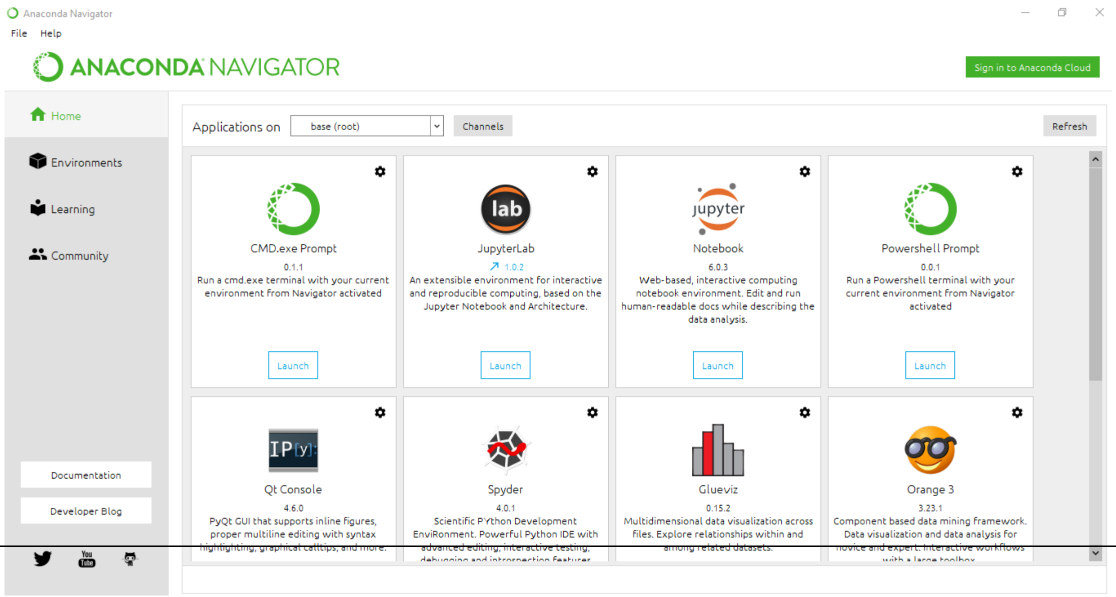
Thank you for installing Anaconda Individual Edition.

Here are some helpful tips and resources to get you started. We recommend you bookmark these links so you can refer back to them later.

☑ Anaconda Individual Edition Tutorial

☑ Learn More About Anaconda

12. After this the ‘Anaconda Navigator’ look like this.



CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated

JupyterLab
↗ 1.0.2
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
6.0.3
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated

Qt Console
4.6.0
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
4.0.1
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Glueviz
0.15.2
Multidimensional data visualization across files. Explore relationships within and among related datasets.

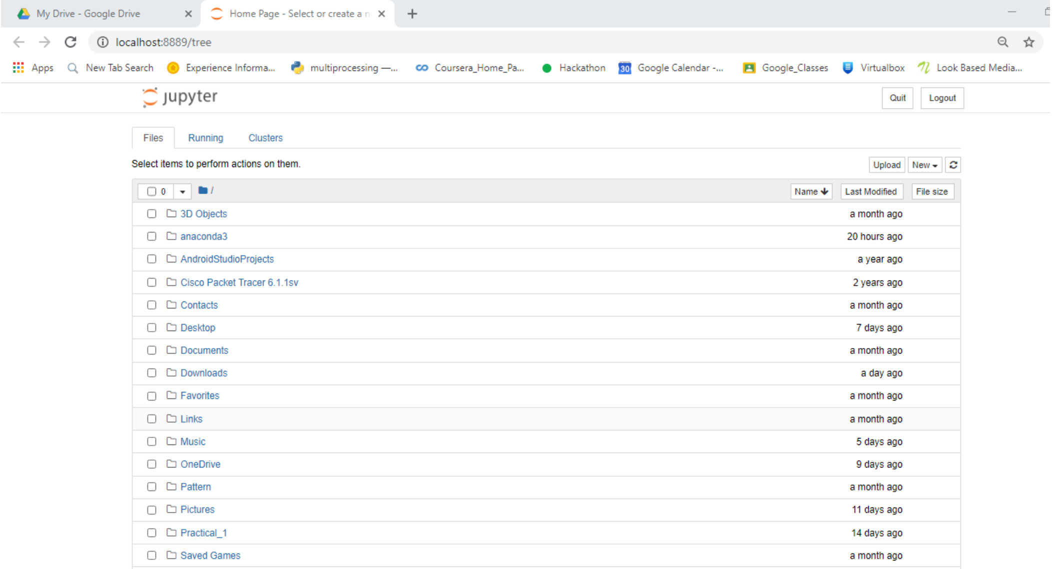
Orange 3
3.23.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

3.2 Working with Jupyter Notebook

- Creating a new notebook

  Steps:

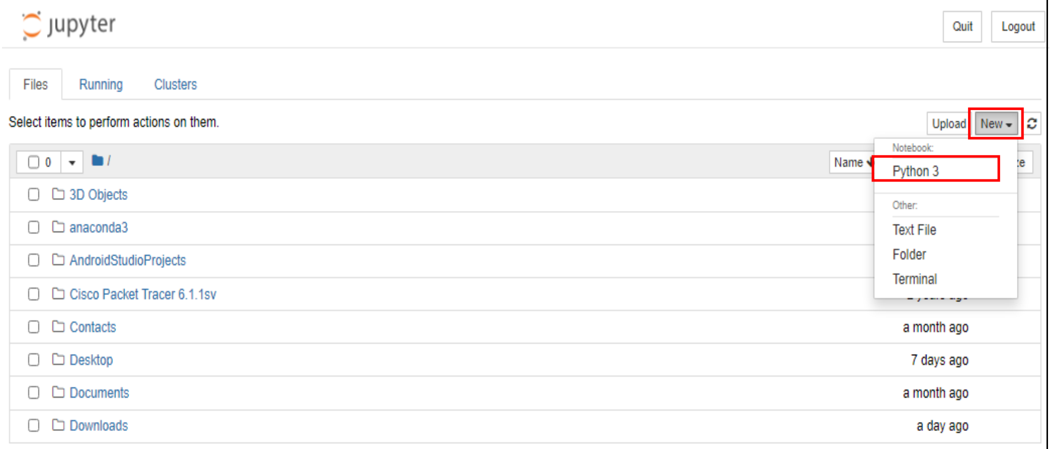
  1. Launch the Jupiter notebook from anaconda prompt. It will open the home page as below.



Select items to perform actions on them.

Name	Last Modified
3D Objects	a month ago
anaconda3	20 hours ago
AndroidStudioProjects	a year ago
Cisco Packet Tracer 6.1.1sv	2 years ago
Contacts	a month ago
Desktop	7 days ago
Documents	a month ago
Downloads	a day ago
Favorites	a month ago
Links	a month ago
Music	5 days ago
OneDrive	9 days ago
Pattern	a month ago
Pictures	11 days ago
Practical_1	14 days ago
Saved Games	a month ago

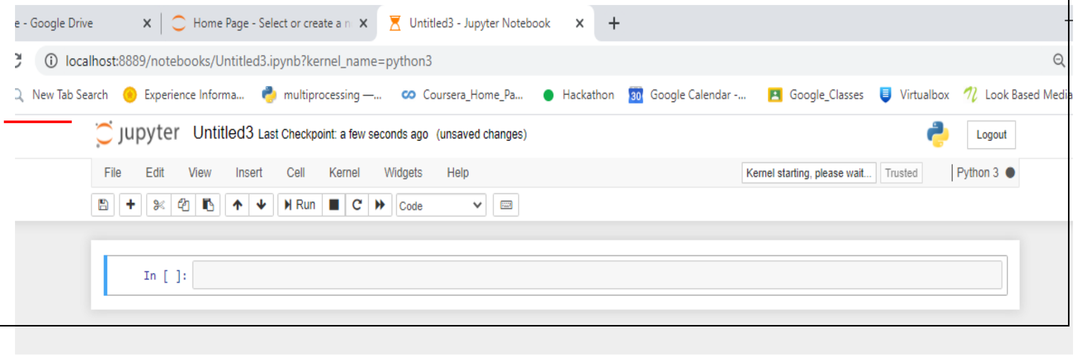
  2. Choose New => Python 3



Select items to perform actions on them.

Name	Last Modified
3D Objects	
anaconda3	
AndroidStudioProjects	
Cisco Packet Tracer 6.1.1sv	
Contacts	a month ago
Desktop	7 days ago
Documents	a month ago
Downloads	a day ago

It will open a new tab in browser with new notebook containing the highlighted cell in which we can type code. The notebook is untitled right now.

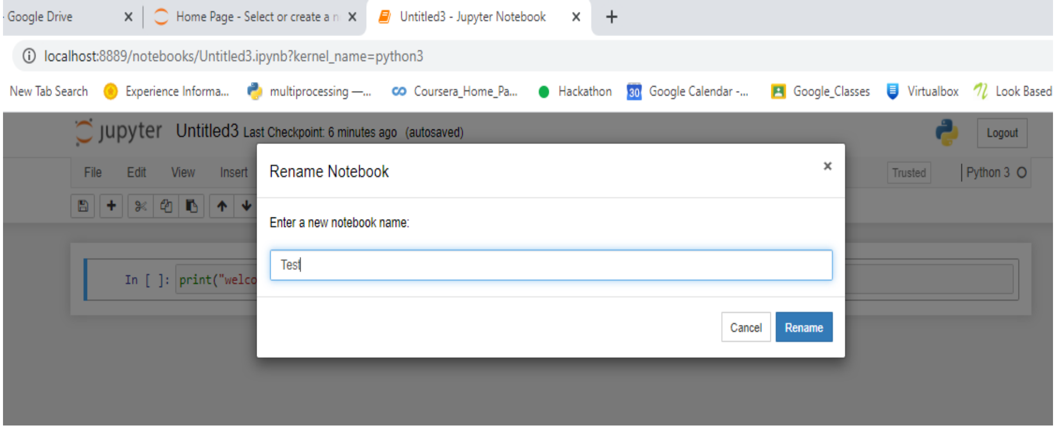


Untitled3 Last Checkpoint: a few seconds ago (unsaved changes)

Kernel starting, please wait...

```
In [ ]:
```

3. Click on 'Untitled' on the page.
   It will ask to enter new name of notebook. Just enter name and click on 'rename'.



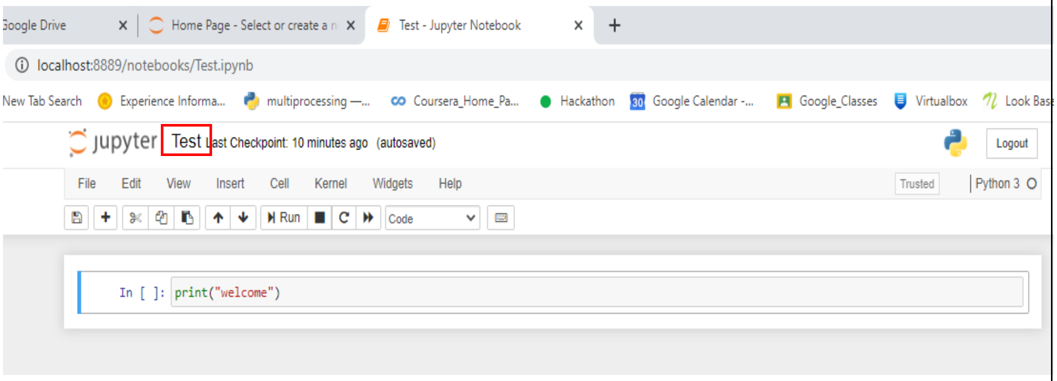
Rename Notebook

Enter a new notebook name:

Test

Untitled3 - Jupyter Notebook
localhost:8889/notebooks/Untitled3.ipynb?kernel_name=python3
jupyter Untitled3 Last Checkpoint: 6 minutes ago (autosaved)
Cancel
Rename

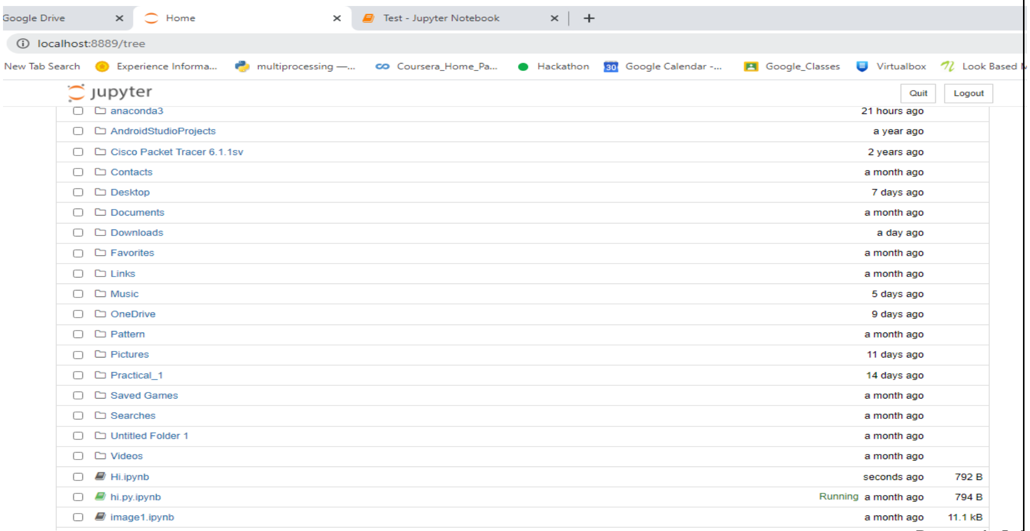
Now, notebook is saved with new name.



```
In [ ]: print("welcome")
```

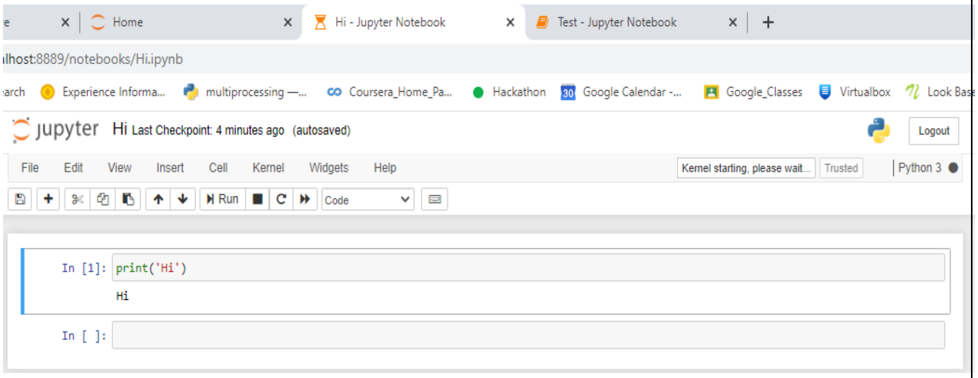
Test - Jupyter Notebook
localhost:8889/notebooks/Test.ipynb
jupyter Test Last Checkpoint: 10 minutes ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Run
Code
Trusted
Python 3
Logout

- Opening existing notebooks
  Steps:
  1. Go to home page.
     Start=>Anaconda Navigator=>Launch jupyter notebook



☐ anaconda3		21 hours ago	
☐ AndroidStudioProjects		a year ago	
☐ Cisco Packet Tracer 6.1.1sv		2 years ago	
☐ Contacts		a month ago	
☐ Desktop		7 days ago	
☐ Documents		a month ago	
☐ Downloads		a day ago	
☐ Favorites		a month ago	
☐ Links		a month ago	
☐ Music		5 days ago	
☐ OneDrive		9 days ago	
☐ Pattern		a month ago	
☐ Pictures		11 days ago	
☐ Practical_1		14 days ago	
☐ Saved Games		a month ago	
☐ Searches		a month ago	
☐ Untitled Folder 1		a month ago	
☐ Videos		a month ago	
☐ Hi.ipynb		seconds ago	792 B
☐ hi.py.ipynb	Running	a month ago	794 B
☐ image1.ipynb		a month ago	11.1 kB

2. Click on existing notebook (for example, Hi.ipynb)

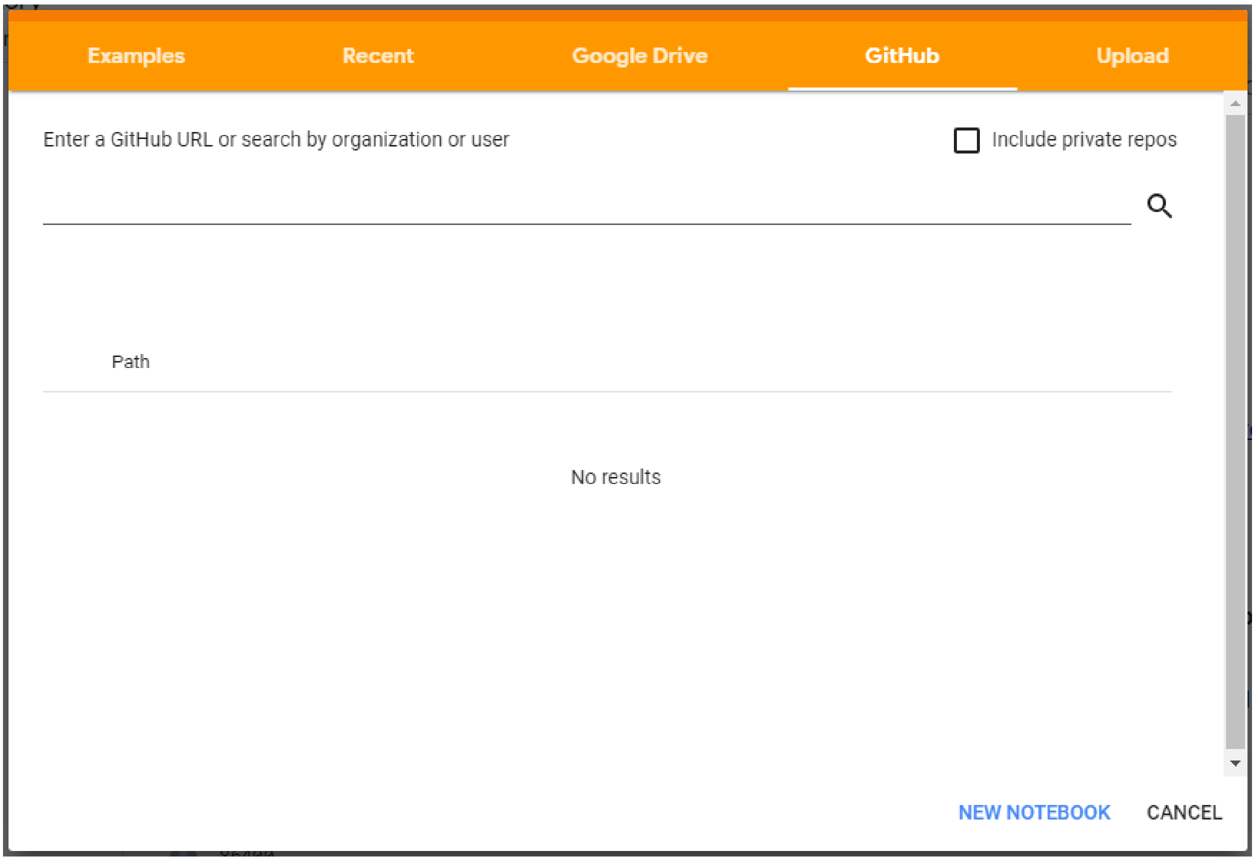


```
In [1]: print('Hi')
        Hi

In [ ]:
```

Now, Notebook is open.

- Using GitHub for existing notebooks

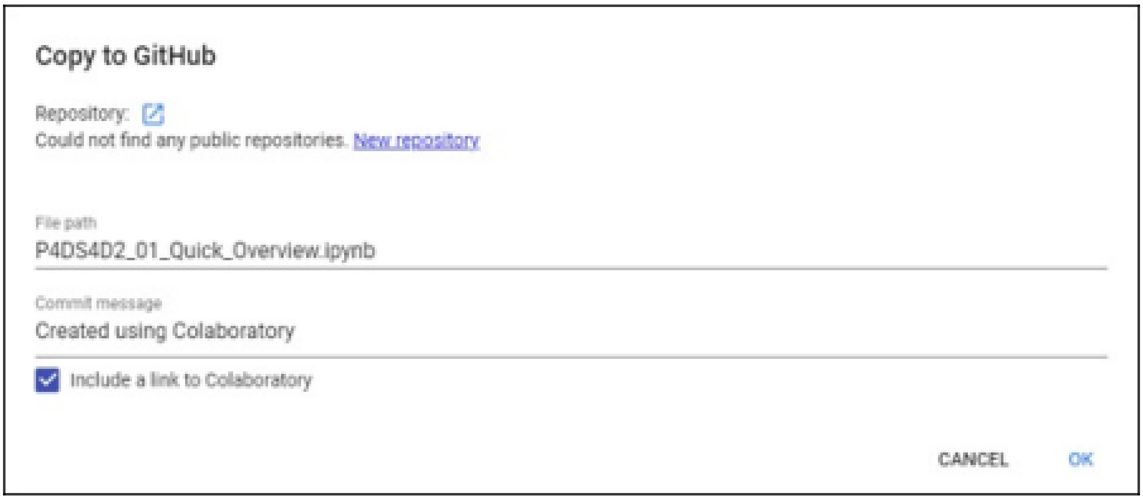
When working with GitHub, initially need to provide the location of the source code online, as shown below. The location must point to a public project, can't use Colab to access private projects.



After making the connection to GitHub, two lists appears: repositories, which are containers for code related to a particular project; and branches, a particular implementation of the code. Selecting a repository and branch displays a list of notebook files that you can load into Colab.

- Using GitHub to save notebooks

GitHub also supports private repositories. To save a file to GitHub, choose File ⇒ Save a Copy in GitHub. After you sign in, a dialog box will be shown as below.



Copy to GitHub

Repository:
Could not find any public repositories. New repository

File path
P4DS4D2_01_Quick_Overview.ipynb

Commit message
Created using Colaboratory

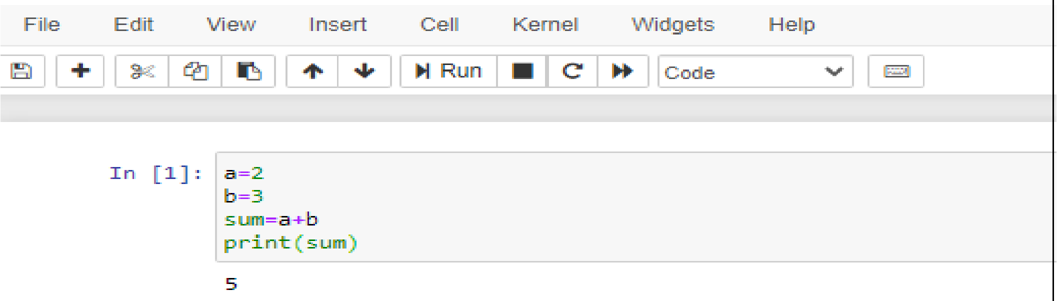
☑ Include a link to Colaboratory

After you save the file, it will appear in GitHub repository of your choice. The repository will include a link to open the data in Colab by default, unless you choose not to include this feature.

3.3 Performing Common Tasks

- Creating code cells

To create code cell it is necessary to select 'code' from the dropdown list. So that the code written in the cell will execute successfully.

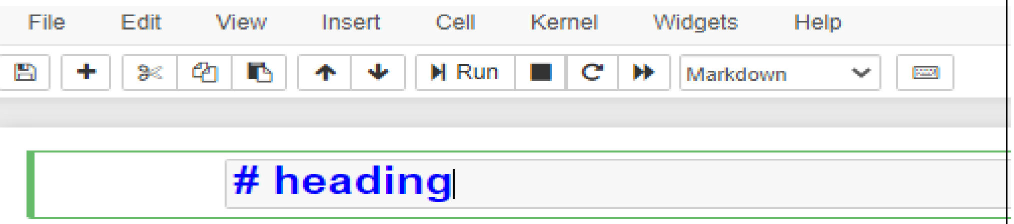


Code

```
In [1]: a=2
        b=3
        sum=a+b
        print(sum)

        5
```

- Creating text cells

To create text select 'markdown' form dropdown list and add text in the cell. Text must be start with '#'



Markdown

```
# heading
```

- Creating special cells Executing the Code

  This type of cells used for the special purpose.

  i.e. Raw NBConvert format will be converted in a way specific to the output (such as HTML or Latex).

- Viewing Your Notebook

  To view exist notebook simply click on that notebook. The content of notebook will be display on the new tab. This content can be modified as per need if required.

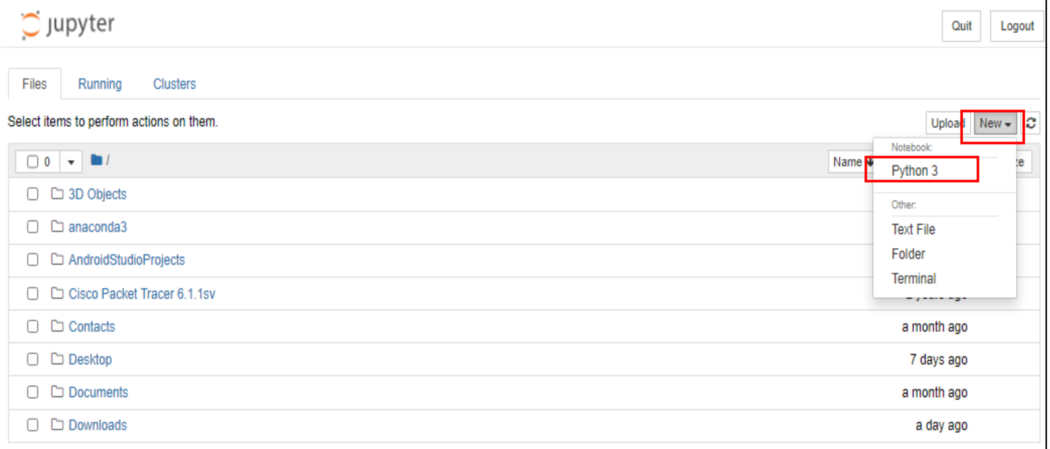
3.4 Defining the code repository

- Defining a new folder

Steps :

1. Choose 'New =>Folder' from homepage.

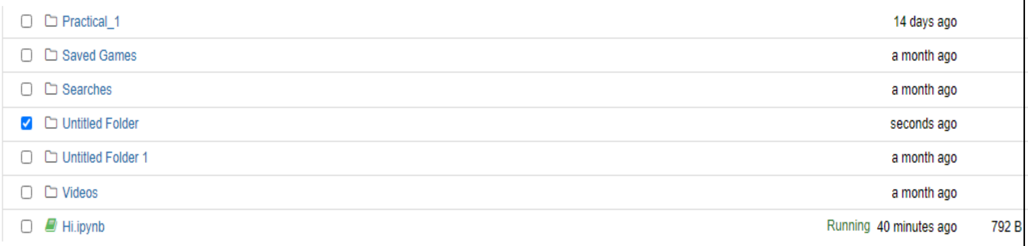
   Notebook will create new folder. The name of folder can vary, by default is 'Untitled Folder'.



Select items to perform actions on them.

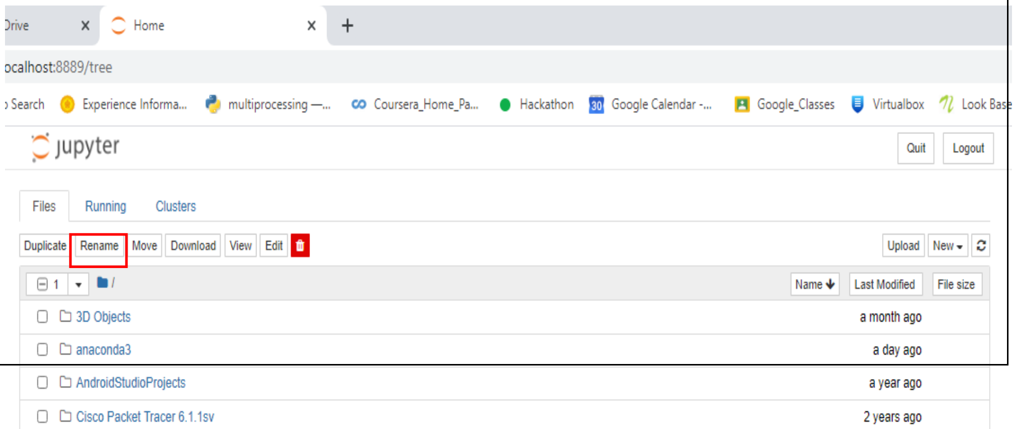
3D Objects	
anaconda3	
AndroidStudioProjects	
Cisco Packet Tracer 6.1.1sv	
Contacts	a month ago
Desktop	7 days ago
Documents	a month ago
Downloads	a day ago

2. Place a check in the box next to the 'Untitled Folder'.



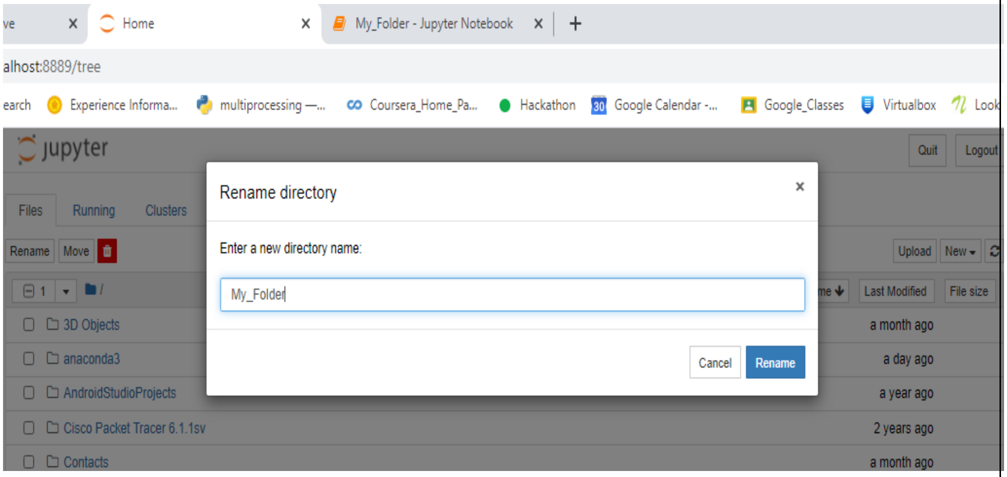
□ Practical_1		14 days ago	
□ Saved Games		a month ago	
□ Searches		a month ago	
☑ Untitled Folder		seconds ago	
□ Untitled Folder 1		a month ago	
□ Videos		a month ago	
□ Hi.ipynb	Running	40 minutes ago	792 B

3. Click rename at the top of the page.



3D Objects	a month ago
anaconda3	a day ago
AndroidStudioProjects	a year ago
Cisco Packet Tracer 6.1.1sv	2 years ago

4. Enter name of the folder you want to give and click on ‘rename’. Notebook will renames the folder.



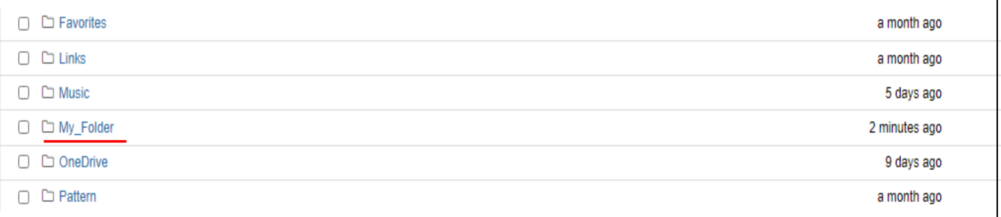
Rename directory

Enter a new directory name:

My_Folder

3D Objects	a month ago
anaconda3	a day ago
AndroidStudioProjects	a year ago
Cisco Packet Tracer 6.1.1sv	2 years ago
Contacts	a month ago

Now new folder will be created and shown in the list.

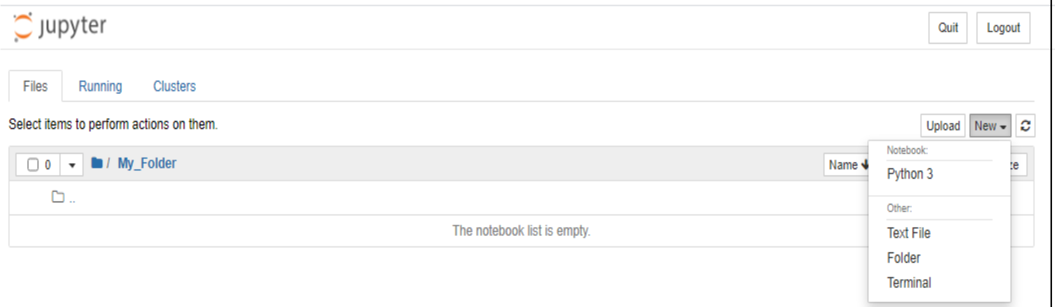


Favorites	a month ago
Links	a month ago
Music	5 days ago
My_Folder	2 minutes ago
OneDrive	9 days ago
Pattern	a month ago

- Creating a new notebook

  Go to newly created folder and create new notebook.

  Root folder->My_Folder->New->Python 3



Select items to perform actions on them.

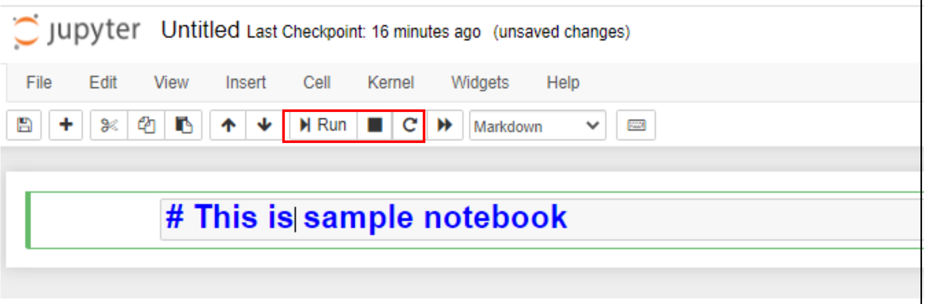
The notebook list is empty.

This will create a new notebook named ‘Untitled’ as same as other notebook.

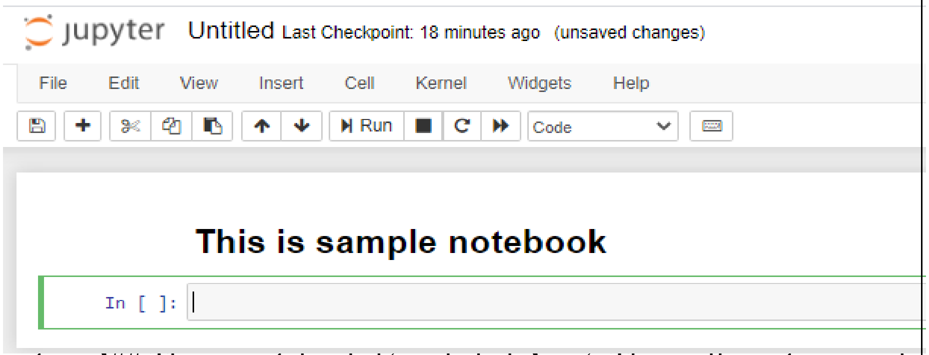
- Adding notebook content

  Steps:

  1. Choose Markdown from the drop-down list that currently contains the word Code.
  2. Type ‘# This is sample notebook’ and click ‘run’. The ‘#’ marks creates a first level heading.


jupyter
Untitled Last Checkpoint: 16 minutes ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Run
Markdown

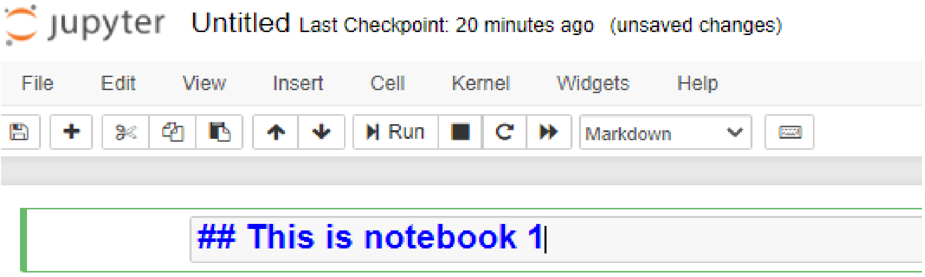
```
# This is sample notebook
```


jupyter
Untitled Last Checkpoint: 18 minutes ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Run
Code

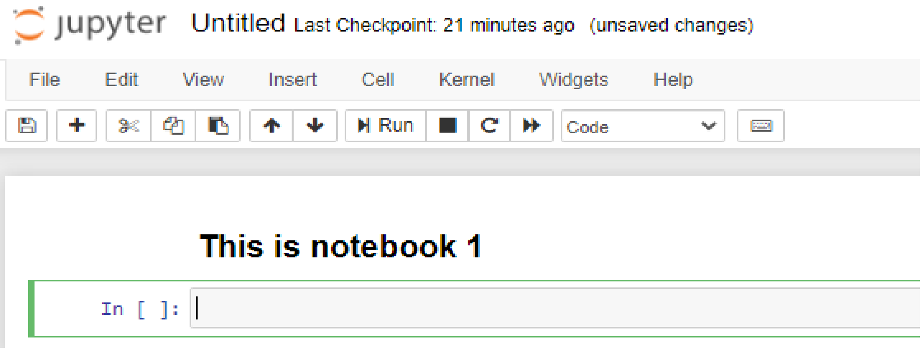
This is sample notebook

```
In [ ]:
```

3. Choose markdown, type ‘## This is notebook 1’ and click ‘run’. This will create second-level heading.


jupyter
Untitled Last Checkpoint: 20 minutes ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Run
Markdown

```
## This is notebook 1
```

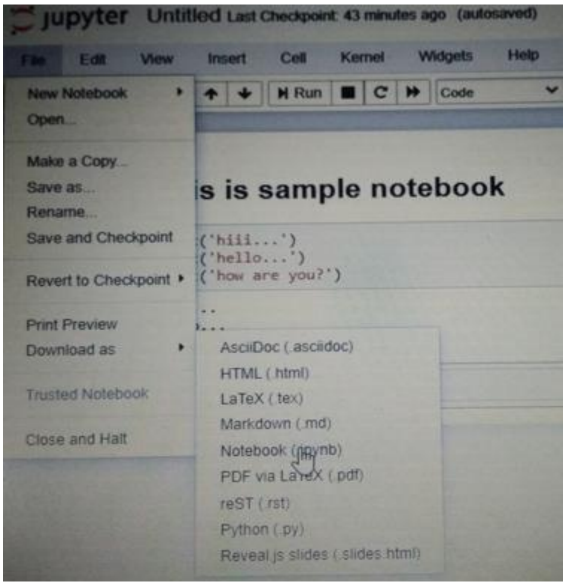

jupyter
Untitled Last Checkpoint: 21 minutes ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Run
Code

This is notebook 1

```
In [ ]:
```

- Exporting a notebook

  To share this file to someone, it is needed to export that file.
  Follow the following steps to export the notebook.
  File->Download as->Notebook (or choose html, pdf, python etc.)



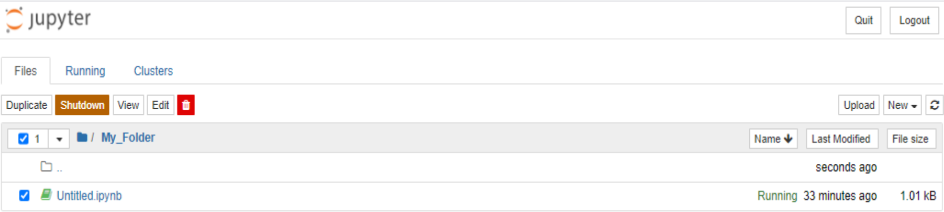
s is sample notebook

```
('hiii...')
('hello...')
('how are you?')
```

- Removing a notebook

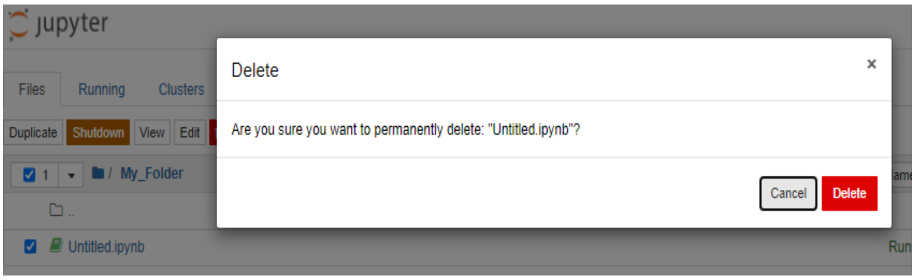
Steps:

1. Select the check box next to the file name.



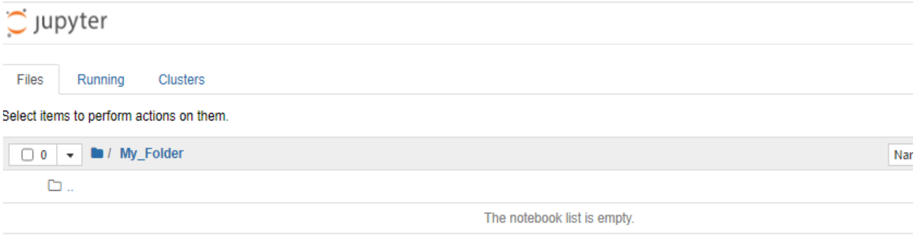
2. Click on delete icon.

It will display warning message like this-



Are you sure you want to permanently delete: "Untitled.ipynb"?

3.



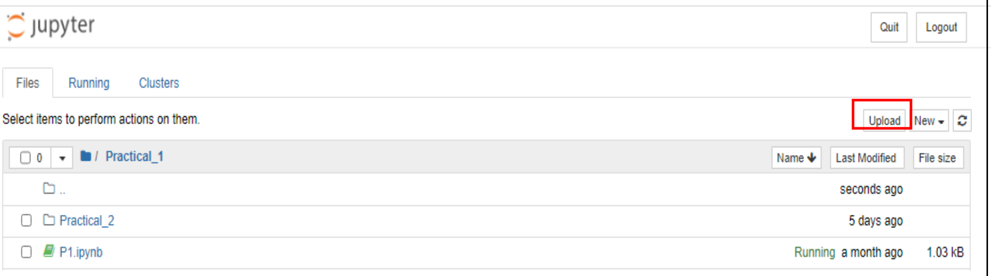
Select items to perform actions on them.

The notebook list is empty.

- Importing a notebook

Steps:

1. Click on ‘upload’ on the notebook.

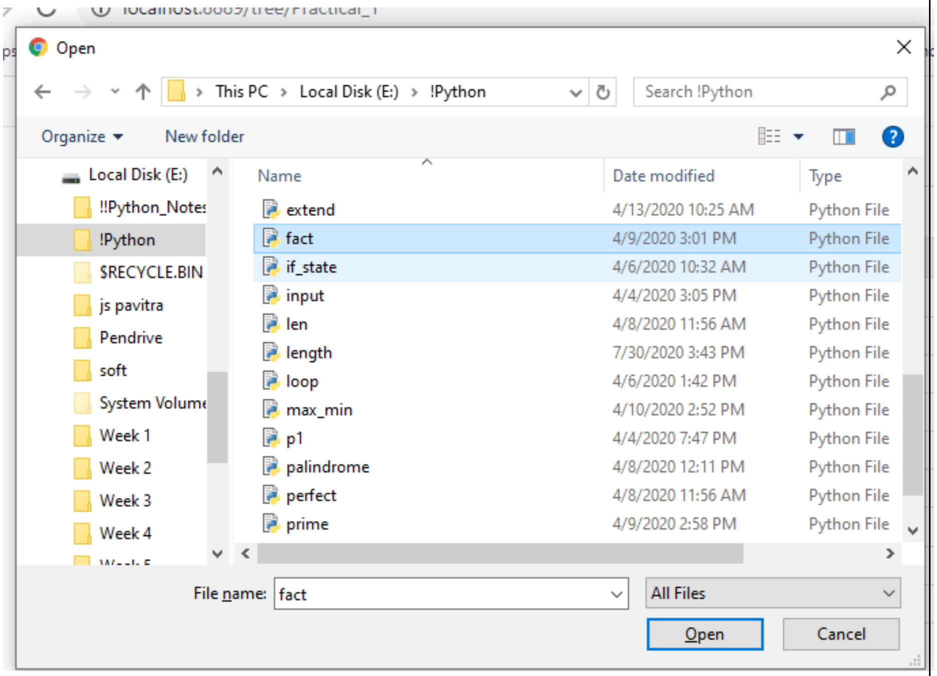


Select items to perform actions on them.

/ Practical_1		
..	seconds ago	
Practical_2	5 days ago	
P1.ipynb (Running)	a month ago	1.03 kB

2. Navigate to the directory containing the files that want to import.

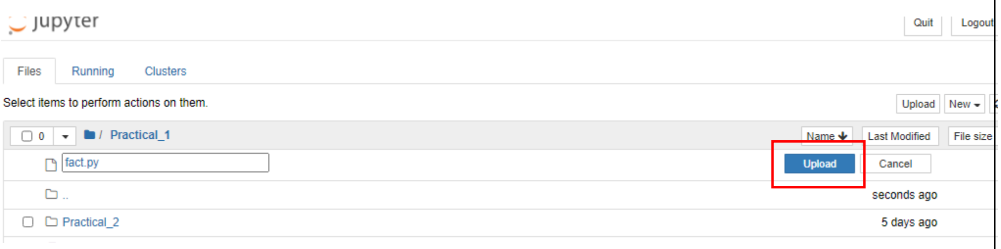
3. Select one or more files, click on ‘open’ to upload the files.



Name	Date modified	Type
extend	4/13/2020 10:25 AM	Python File
fact	4/9/2020 3:01 PM	Python File
if_state	4/6/2020 10:32 AM	Python File
input	4/4/2020 3:05 PM	Python File
len	4/8/2020 11:56 AM	Python File
length	7/30/2020 3:43 PM	Python File
loop	4/6/2020 1:42 PM	Python File
max_min	4/10/2020 2:52 PM	Python File
p1	4/4/2020 7:47 PM	Python File
palindrome	4/8/2020 12:11 PM	Python File
perfect	4/8/2020 11:56 AM	Python File
prime	4/9/2020 2:58 PM	Python File

File name: fact

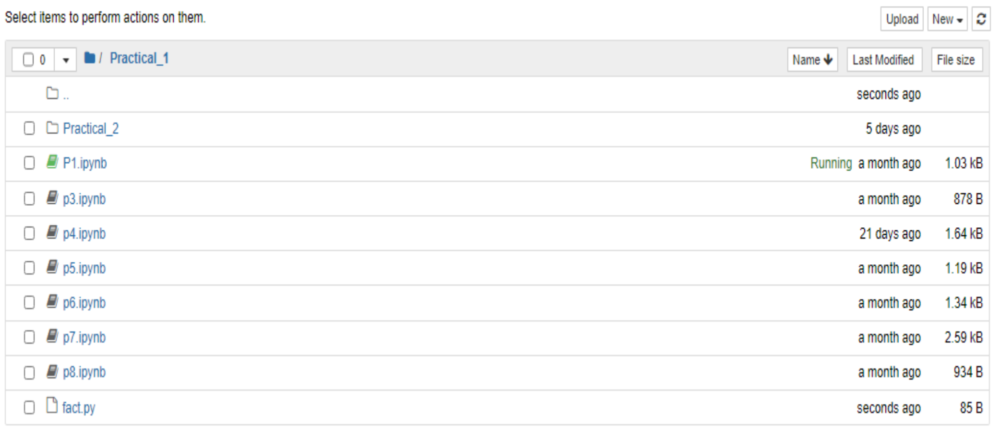
4. Click upload.



Select items to perform actions on them.

/ Practical_1	
fact.py	
..	seconds ago
Practical_2	5 days ago

Now, the file is imported as shown below.

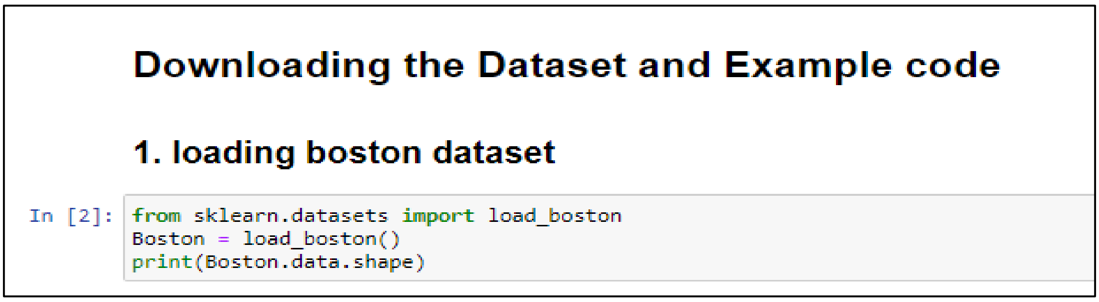


/ Practical_1

□ ..		seconds ago	
□ Practical_2		5 days ago	
□ P1.ipynb	Running	a month ago	1.03 kB
□ p3.ipynb		a month ago	878 B
□ p4.ipynb		21 days ago	1.64 kB
□ p5.ipynb		a month ago	1.19 kB
□ p6.ipynb		a month ago	1.34 kB
□ p7.ipynb		a month ago	2.59 kB
□ p8.ipynb		a month ago	934 B
□ fact.py		seconds ago	85 B

3.5 Understating the following dataset with its code.

- load_boston(): Regression analysis with the Boston house-prices dataset



Downloading the Dataset and Example code

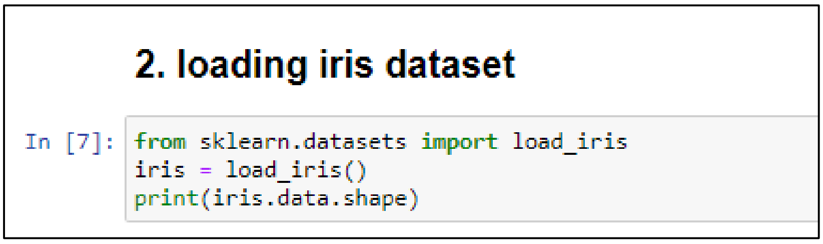
1. loading boston dataset

```
In [2]: from sklearn.datasets import load_boston
        Boston = load_boston()
        print(Boston.data.shape)
```

Output:

```
(506, 13)
```

- load_iris(): Classification with the Iris dataset



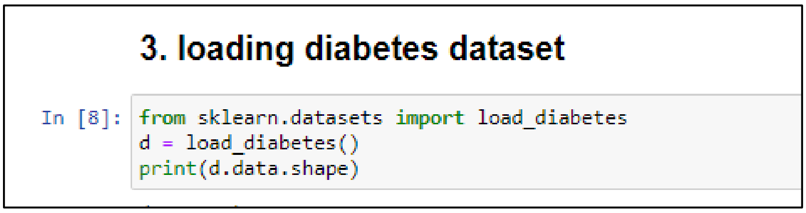
2. loading iris dataset

```
In [7]: from sklearn.datasets import load_iris
        iris = load_iris()
        print(iris.data.shape)
```

Output:

```
(150, 4)
```

- load_diabetes(): Regression with the diabetes dataset



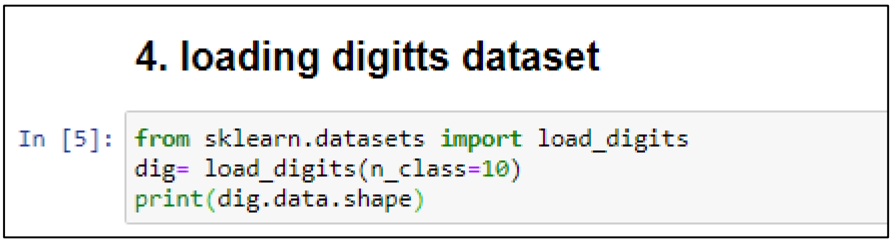
3. loading diabetes dataset

```
In [8]: from sklearn.datasets import load_diabetes
        d = load_diabetes()
        print(d.data.shape)
```

Output:

```
(442, 10)
```

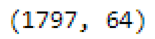
- load_digits([n_class]): Classification with the digits dataset



4. loading digitts dataset

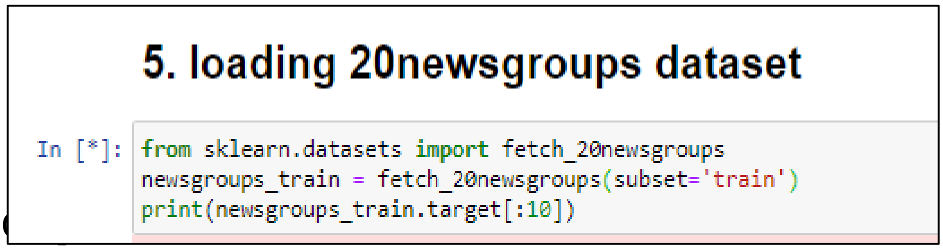
```
In [5]: from sklearn.datasets import load_digits
        dig= load_digits(n_class=10)
        print(dig.data.shape)
```

Output:



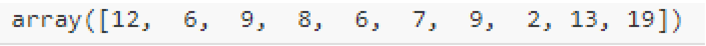
```
(1797, 64)
```

- fetch_20newsgroups(subset='train'): Data from 20 newsgroups



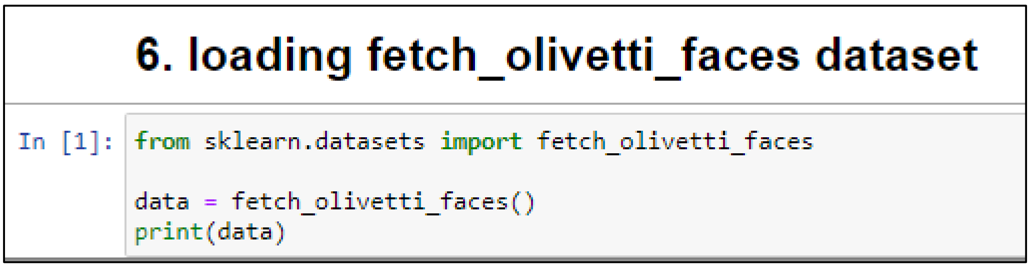
5. loading 20newsgroups dataset

```
In [*]: from sklearn.datasets import fetch_20newsgroups
        newsgroups_train = fetch_20newsgroups(subset='train')
        print(newsgroups_train.target[:10])
```



```
array([12,  6,  9,  8,  6,  7,  9,  2, 13, 19])
```

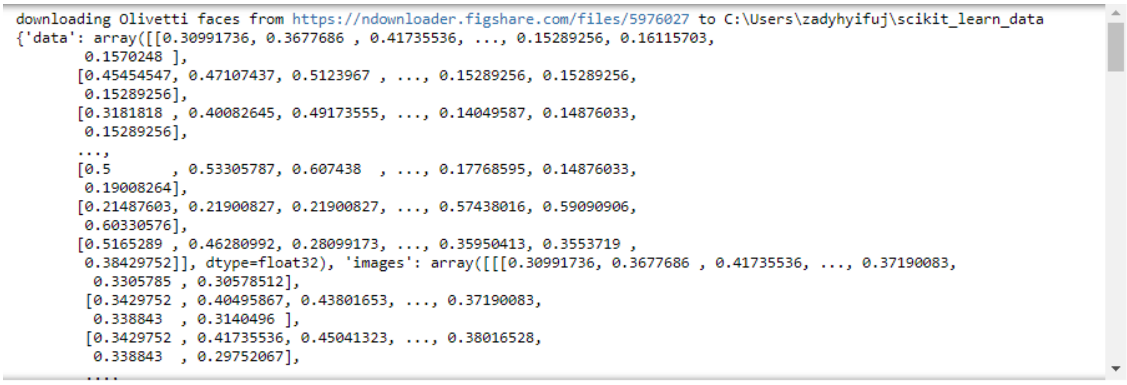
- fetch_olivetti_faces(): Olivetti faces dataset from AT&T



6. loading fetch_olivetti_faces dataset

```
In [1]: from sklearn.datasets import fetch_olivetti_faces

        data = fetch_olivetti_faces()
        print(data)
```

Output:



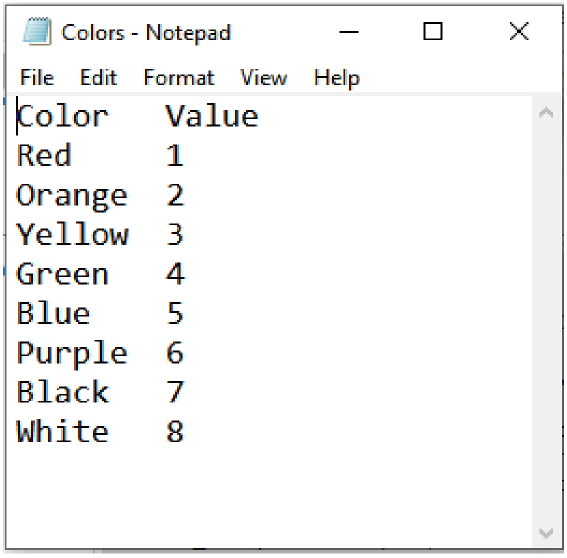
```
downloading Olivetti faces from https://ndownloader.figshare.com/files/5976027 to C:\Users\zadyhyifuj\scikit_learn_data
{'data': array([[0.30991736, 0.3677686 , 0.41735536, ..., 0.15289256, 0.16115703,
        0.1570248 ],
       [0.45454547, 0.47107437, 0.5123967 , ..., 0.15289256, 0.15289256,
        0.15289256],
       [0.3181818 , 0.40082645, 0.49173555, ..., 0.14049587, 0.14876033,
        0.15289256],
       ...,
       [0.5       , 0.53305787, 0.607438  , ..., 0.17768595, 0.14876033,
        0.19008264],
       [0.21487603, 0.21900827, 0.21900827, ..., 0.57438016, 0.59090906,
        0.60330576],
       [0.5165289 , 0.46280992, 0.28099173, ..., 0.35950413, 0.3553719 ,
        0.38429752]], dtype=float32), 'images': array([[[0.30991736, 0.3677686 , 0.41735536, ..., 0.37190083,
         0.3305785 , 0.30578512],
        [0.3429752 , 0.40495867, 0.43801653, ..., 0.37190083,
         0.338843  , 0.3140496 ],
        [0.3429752 , 0.41735536, 0.45041323, ..., 0.38016528,
         0.338843  , 0.29752067],
        ....
```

PRACTICAL : 4

Aim : Uploading, Streaming, and Sampling Data using Pandas.

4.1 : Uploading small amounts of data in Colors.txt into memory.

Colors.txt

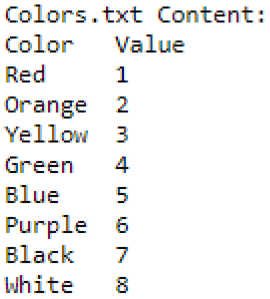


```
Color    Value
Red      1
Orange   2
Yellow   3
Green    4
Blue     5
Purple   6
Black    7
White    8
```

Program:

```
with open("res/Colors.txt",'r') as open_file:
    print('Colors.txt Content:\n' + open_file.read())
```

Output:



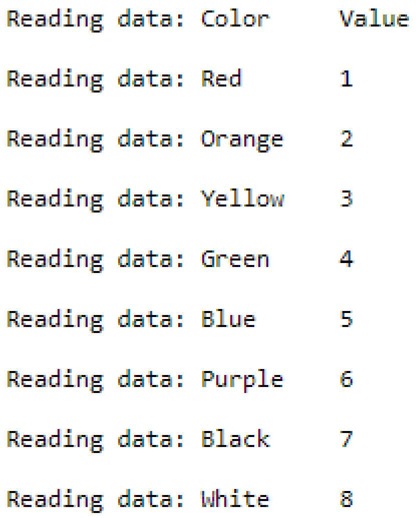
```
Colors.txt Content:
Color    Value
Red      1
Orange   2
Yellow   3
Green    4
Blue     5
Purple   6
Black    7
White    8
```

4.2 : Streaming large amounts of data in Colors.txt into memory.

Program:

```
with open("res/Colors.txt",'r') as open_file:
   for ob in open_file:
      print('Reading data: ' + ob)
```

Output:



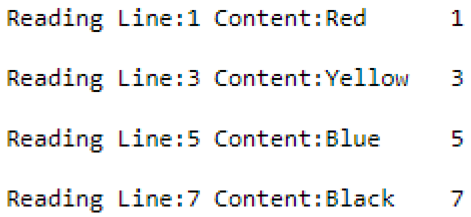
```
Reading data: Color     Value

Reading data: Red       1

Reading data: Orange    2

Reading data: Yellow    3

Reading data: Green     4

Reading data: Blue      5

Reading data: Purple    6

Reading data: Black     7

Reading data: White     8
```

4.3 : Retrieve every odd number record from the file Colors.txt (Data Sampling).

Program:

```
n = 2
with open("res/Colors.txt",'r') as open_file:
   for j, ob in enumerate(open_file):
      if j%2!=0 :
         print('Reading Line:'+str(j)+' Content:'+ob)
```

Output:



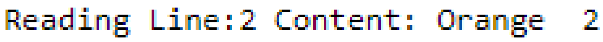
```
Reading Line:1 Content:Red       1

Reading Line:3 Content:Yellow    3

Reading Line:5 Content:Blue      5

Reading Line:7 Content:Black     7
```

4.4 : Select random samples from the file Colors.txt

Program:

```
from random import random
sample_size=0.25
with open("res/Colors.txt",'r') as open_file:
    for j, ob in enumerate(open_file):
        if random()<=sample_size :
            print('Reading Line:'+str(j)+' Content: '+ob)
```

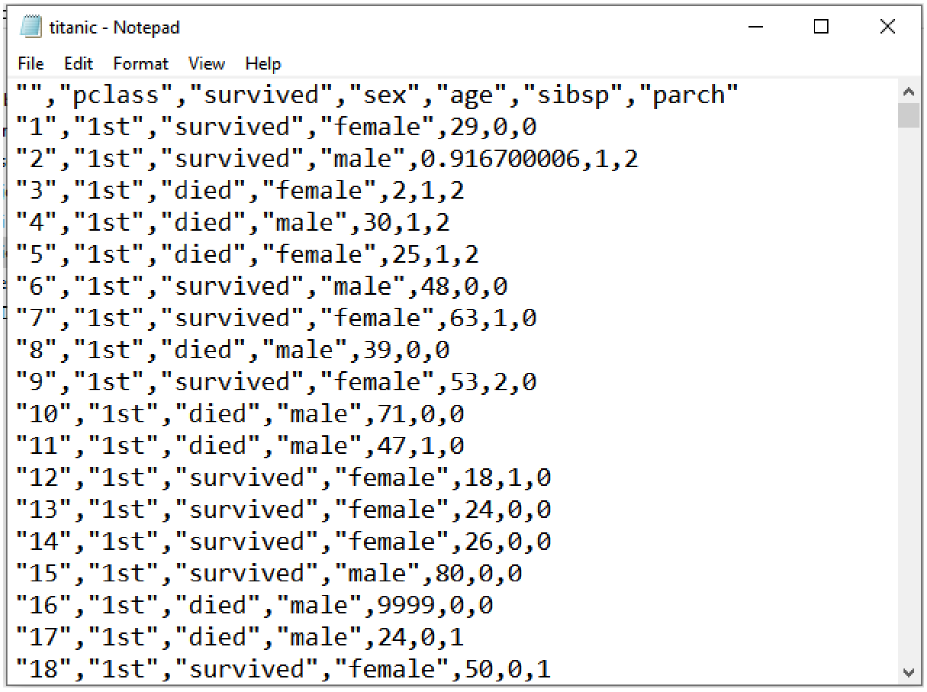
Output:



```
Reading Line:2 Content: Orange  2
```

4.5 : Read the csv file named titanic.csv and print the values.

titanic.csv



```
"","pclass","survived","sex","age","sibsp","parch"
"1","1st","survived","female",29,0,0
"2","1st","survived","male",0.916700006,1,2
"3","1st","died","female",2,1,2
"4","1st","died","male",30,1,2
"5","1st","died","female",25,1,2
"6","1st","survived","male",48,0,0
"7","1st","survived","female",63,1,0
"8","1st","died","male",39,0,0
"9","1st","survived","female",53,2,0
"10","1st","died","male",71,0,0
"11","1st","died","male",47,1,0
"12","1st","survived","female",18,1,0
"13","1st","survived","female",24,0,0
"14","1st","survived","female",26,0,0
"15","1st","survived","male",80,0,0
"16","1st","died","male",9999,0,0
"17","1st","died","male",24,0,1
"18","1st","survived","female",50,0,1
```

Program:

```
import pandas as pd
titanic = pd.io.parsers.read_csv("res/titanic.csv")
data= titanic[['age']]
print(data)
```

Output:

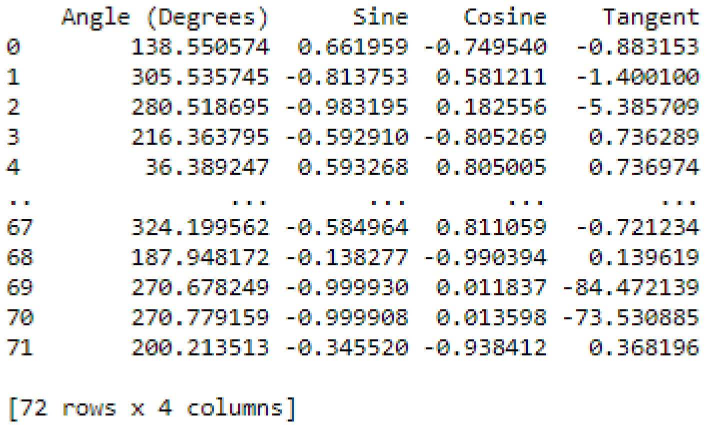
```
            age
0       29.0000
1        0.9167
2        2.0000
3       30.0000
4       25.0000
...         ...
1304    14.5000
1305  9999.0000
1306    26.5000
1307    27.0000
1308    29.0000

[1309 rows x 1 columns]
```

4.6 : Read the Excel file named values.xls file and parse the values and print it.

Program:

```
import pandas as pd
excel=pd.ExcelFile("res/Values.xls")
data=excel.parse('Sheet1',index_col=None,na_values=['NA'])
print(data)
```

Output:



```
    Angle (Degrees)      Sine    Cosine    Tangent
0        138.550574  0.661959 -0.749540  -0.883153
1        305.535745 -0.813753  0.581211  -1.400100
2        280.518695 -0.983195  0.182556  -5.385709
3        216.363795 -0.592910 -0.805269   0.736289
4         36.389247  0.593268  0.805005   0.736974
..              ...       ...       ...        ...
67       324.199562 -0.584964  0.811059  -0.721234
68       187.948172 -0.138277 -0.990394   0.139619
69       270.678249 -0.999930  0.011837 -84.472139
70       270.779159 -0.999908  0.013598 -73.530885
71       200.213513 -0.345520 -0.938412   0.368196

[72 rows x 4 columns]
```

4.7 : Write a python script to read the image stored in local storage as well as on the specific URL. Also perform the following operations on image.

a) **Displaying the image information**

b) **Cropping the image**

c) **Resizing the image**

d) **Flatening the image**

I. Reading image stored in local storage

Program:

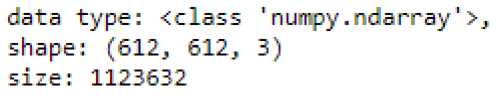
```
from skimage.io import imread
from skimage.transform import resize
from matplotlib import pyplot as pl
import matplotlib.cm as cm

ex_file=("res/purvi.jpg")
image=imread(ex_file, as_gray=False)
pl.imshow(image,cmap=cm.gray)
pl.show()
```

Output:

a) **Displaying the image information**

```
print("data type: %s,"%(type(image)))
print("shape:",image.shape)
print("size:",image.size)
```

Output:

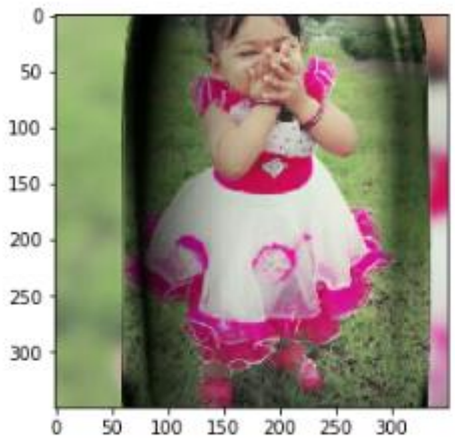


```
data type: <class 'numpy.ndarray'>,
shape: (612, 612, 3)
size: 1123632
```

b) Cropping the image

```
image2 = image[200:550,0:350]
pl.imshow(image2, cmap=cm.gray)
pl.show()
```

Output:

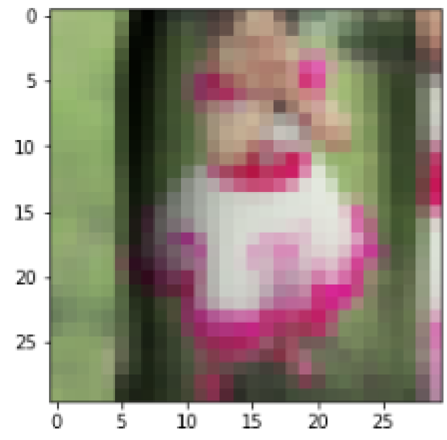


c) Resizing the image

```
image3 = resize(image2, (30, 30), mode='symmetric')
pl.imshow(image3, cmap=cm.gray)
print("data type: %s, shape: %s" %
(type(image3), image3.shape))
```

Output:

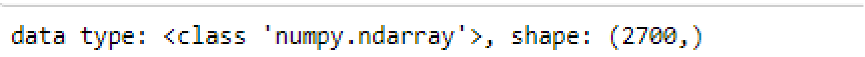
```
data type: <class 'numpy.ndarray'>, shape: (30, 30, 3)
```



d) Flatening the image

```
image_row = image3.flatten()
print("data type: %s, shape: %s" %
(type(image_row), image_row.shape))
```

Output:



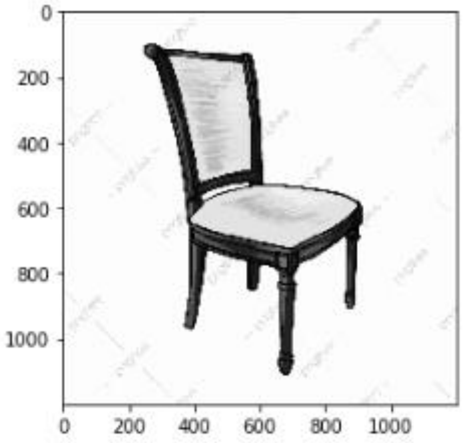
```
data type: <class 'numpy.ndarray'>, shape: (2700,)
```

II. Reading image from specific URL

Program:

```
from skimage.io import imread
from skimage.transform import resize
from matplotlib import pyplot as pl
import matplotlib.cm as cm

ex_file=("https://png.pngtree.com/png-clipart/20190617/original/pngtree-restaurant-chair-wooden-chair-beautiful-chair-exquisite-chair-png-image_3872153.jpg")
im=imread(ex_file, as_gray=True)
pl.imshow(im,cmap=cm.gray)
pl.show()
```

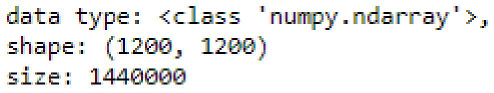
Output:



a) Displaying the image information

```
print("data type: %s,"%(type(image)))
print("shape:",im.shape)
print("size:",im.size)
```

Output:

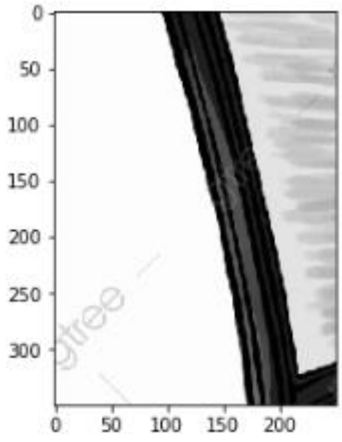


```
data type: <class 'numpy.ndarray'>,
shape: (1200, 1200)
size: 1440000
```

b) Cropping the image

image2 = im[200:550,200:450]
pl.imshow(image2, cmap=cm.gray)
pl.show()

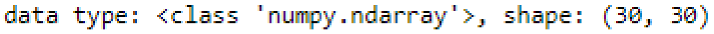
Output:



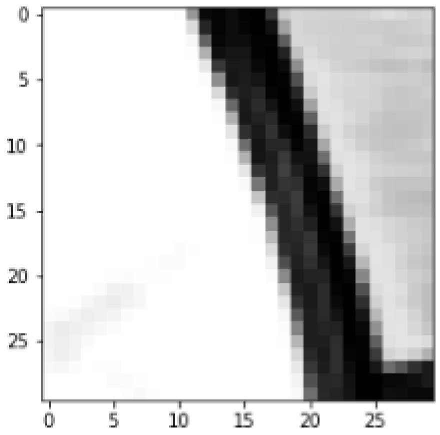
c) Resizing the image

image3 = resize(image2, (30, 30), mode='symmetric')
pl.imshow(image3, cmap=cm.gray)
print("data type: %s, shape: %s" %
(type(image3), image3.shape))

Output:



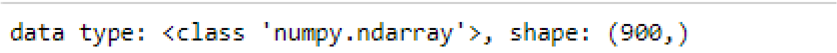
```
data type: <class 'numpy.ndarray'>, shape: (30, 30)
```



d) Flatening the image

```
image_row = image3.flatten()
print("data type: %s, shape: %s" %
(type(image_row), image_row.shape))
```

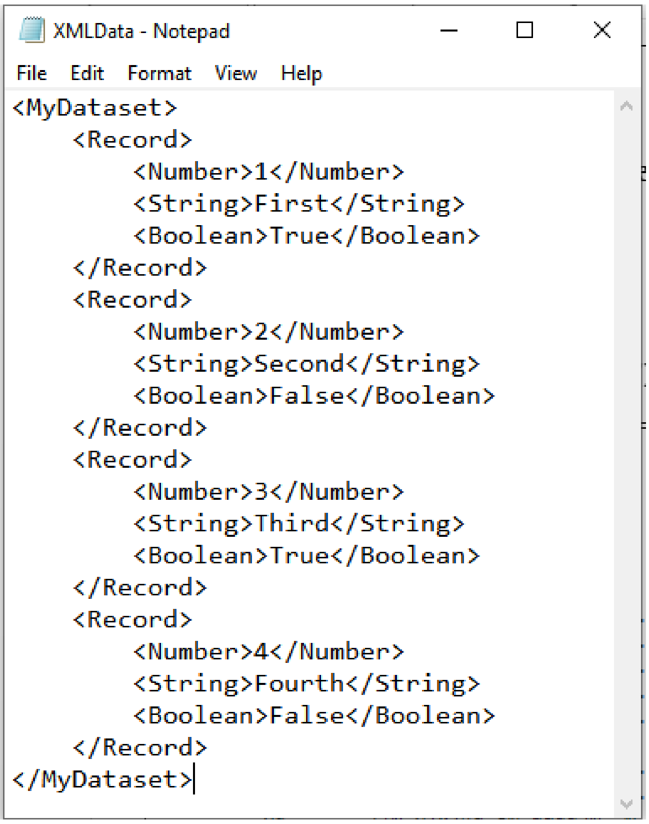
Output:



```
data type: <class 'numpy.ndarray'>, shape: (900,)
```

4.8 : Read the data from the given XMLData.xml file

XMLData.xml

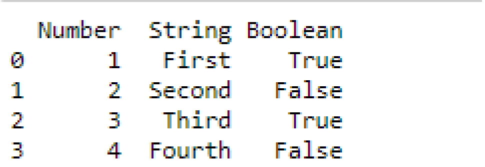


```
<MyDataset>
    <Record>
        <Number>1</Number>
        <String>First</String>
        <Boolean>True</Boolean>
    </Record>
    <Record>
        <Number>2</Number>
        <String>Second</String>
        <Boolean>False</Boolean>
    </Record>
    <Record>
        <Number>3</Number>
        <String>Third</String>
        <Boolean>True</Boolean>
    </Record>
    <Record>
        <Number>4</Number>
        <String>Fourth</String>
        <Boolean>False</Boolean>
    </Record>
</MyDataset>
```

Program:

```
from lxml import objectify
import pandas as pd
xml = objectify.parse(open('res/XMLData.xml'))
root = xml.getroot()
df = pd.DataFrame(columns=('Number', 'String', 'Boolean'))
for i in range(0,4):
   obj = root.getchildren()[i].getchildren()
   row = dict(zip(['Number', 'String', 'Boolean'],
               [obj[0].text, obj[1].text,
               obj[2].text]))

   row_s = pd.Series(row)
   row_s.name = i
   df = df.append(row_s)
print(df)
```

Output:

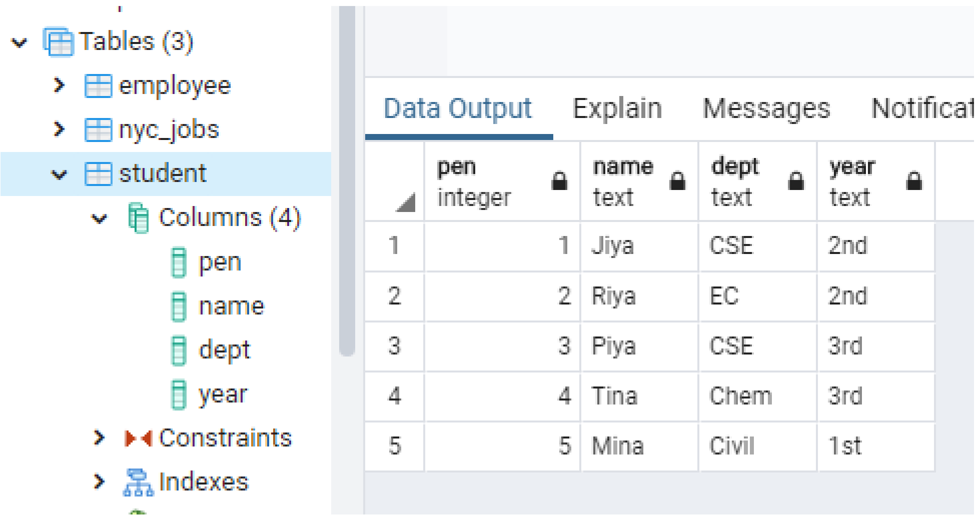


```
  Number  String Boolean
0      1   First    True
1      2  Second   False
2      3   Third    True
3      4  Fourth   False
```

PRACTICAL : 5

Aim : Connecting Pandas to a PostgreSQL Database with SQLAlchemy

5.1 Create a SQLAlchemy Connection with Postgres Database and retrieve data from the table in Postgresql

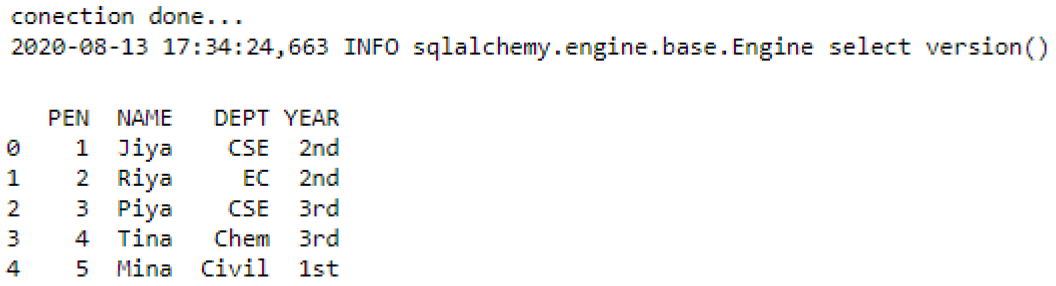


	pen integer	name text	dept text	year text
1	1	Jiya	CSE	2nd
2	2	Riya	EC	2nd
3	3	Piya	CSE	3rd
4	4	Tina	Chem	3rd
5	5	Mina	Civil	1st

Program:

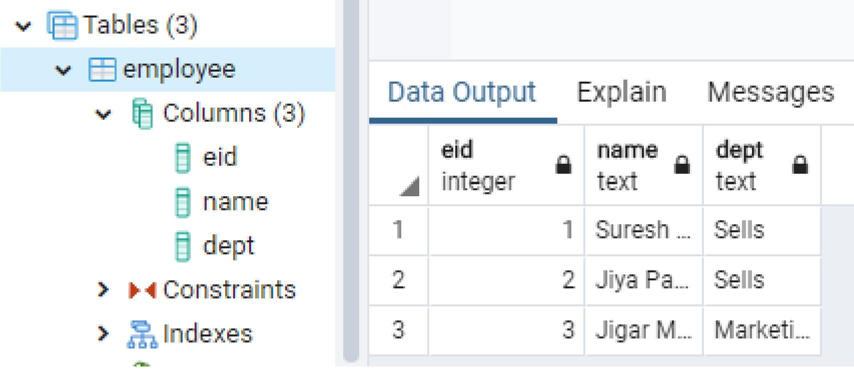
```
from sqlalchemy import create_engine
import pandas as pd

uri = "postgresql://postgres:root@localhost:5432/postgres"
db=create_engine(uri,echo=True)
print('conection done...')
result = db.execute("select * from student;")
data=pd.DataFrame(result,columns=['PEN','NAME','DEPT','YEAR'])
print()
print(data)
```

Output:



```
conection done...
2020-08-13 17:34:24,663 INFO sqlalchemy.engine.base.Engine select version()

   PEN  NAME   DEPT YEAR
0    1  Jiya    CSE  2nd
1    2  Riya     EC  2nd
2    3  Piya    CSE  3rd
3    4  Tina   Chem  3rd
4    5  Mina  Civil  1st
```

5.2 Create a table in PostgreSQL and retrieve the data using read_sql_query().

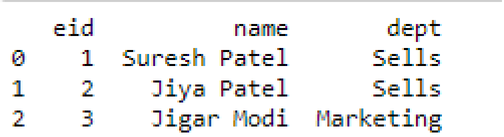


	eid integer	name text	dept text
1	1	Suresh ...	Sells
2	2	Jiya Pa...	Sells
3	3	Jigar M...	Marketi...

Program:

```
import psycopg2 as pg
import pandas as pd

connection = pg.connect(user = "postgres",password = "root",host = "127.0.0.1",port = "5432",database =
"postgres")
df=pd.read_sql_query('select * from employee',con=connection)
print(df)
```

Output:



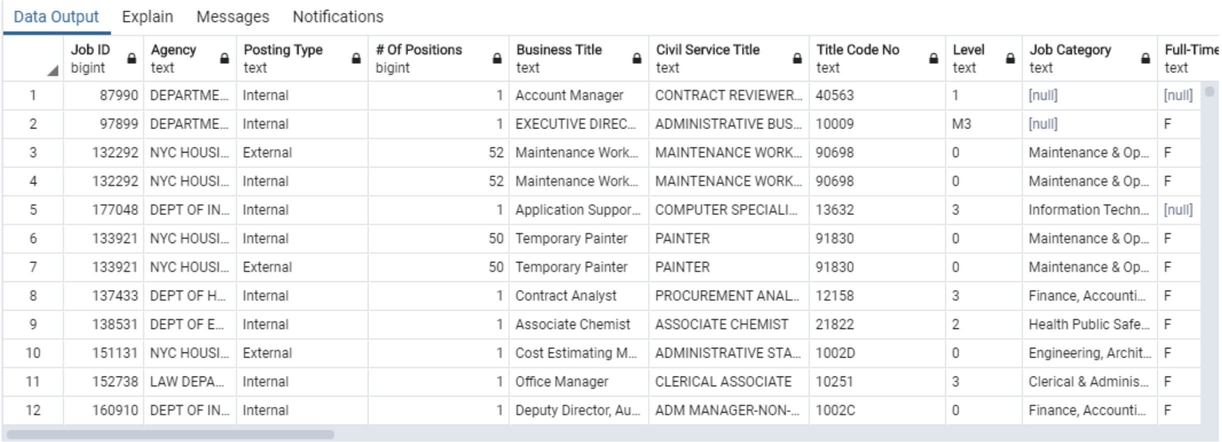
```
   eid         name       dept
0    1  Suresh Patel      Sells
1    2    Jiya Patel      Sells
2    3    Jigar Modi  Marketing
```

5.3 Create a SQL table from data in a CSV. The CSV containing NYC job data

Program:

```
import pandas as pd
jobs_df=pd.read_csv('res/nyc-jobs.csv')

from sqlalchemy.types import Integer, Text, String, DateTime
from sqlalchemy import create_engine
table_name = 'nyc_jobs'
engine=create_engine("postgresql://postgres:root@localhost:5432/postgres",echo=True)
jobs_df.to_sql(
    table_name,
    engine,
    if_exists='replace',
    index=False,
    chunksize=500,
    dtype={
        "job_id": Integer,
        "agency": Text,
        "business_title": Text,
        "job_category":  Text,
        "salary_range_from": Integer,
        "salary_range_to": Integer,
        "salary_frequency": String(50),
        "work_location": Text,
        "division/work_unit": Text,
        "job_description": Text,
        "posting_date": DateTime,
        "posting_updated": DateTime
    }
)
table_df=pd.read_sql_table(table_name,con=engine)
print(table_df.info())
```

Output:



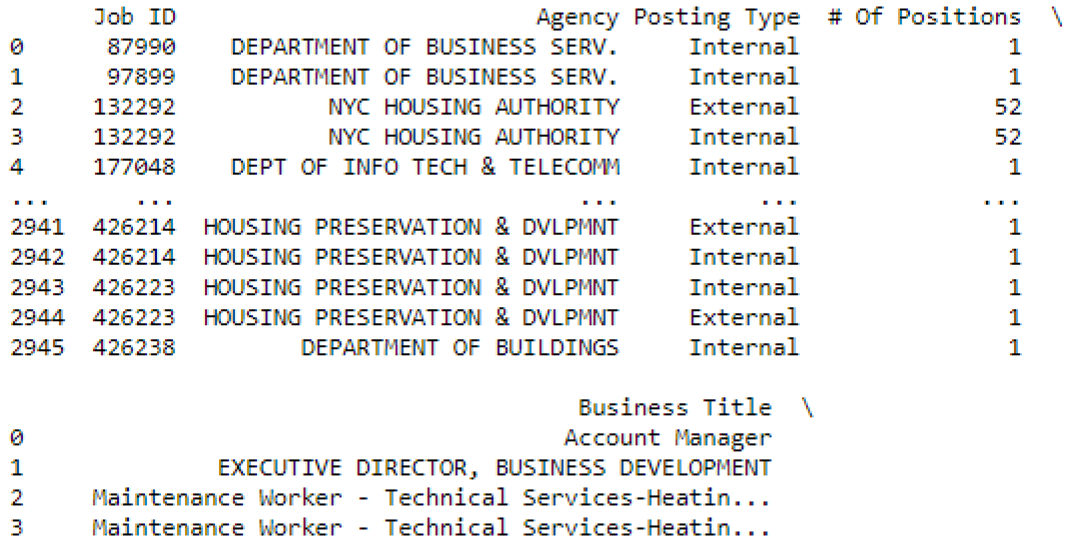
	Job ID bigint	Agency text	Posting Type text	# Of Positions bigint	Business Title text	Civil Service Title text	Title Code No text	Level text	Job Category text	Full-Time text
1	87990	DEPARTME...	Internal	1	Account Manager	CONTRACT REVIEWER...	40563	1	[null]	[null]
2	97899	DEPARTME...	Internal	1	EXECUTIVE DIREC...	ADMINISTRATIVE BUS...	10009	M3	[null]	F
3	132292	NYC HOUSI...	External	52	Maintenance Work...	MAINTENANCE WORK...	90698	0	Maintenance & Op...	F
4	132292	NYC HOUSI...	Internal	52	Maintenance Work...	MAINTENANCE WORK...	90698	0	Maintenance & Op...	F
5	177048	DEPT OF IN...	Internal	1	Application Suppor...	COMPUTER SPECIALI...	13632	3	Information Techn...	[null]
6	133921	NYC HOUSI...	Internal	50	Temporary Painter	PAINTER	91830	0	Maintenance & Op...	F
7	133921	NYC HOUSI...	External	50	Temporary Painter	PAINTER	91830	0	Maintenance & Op...	F
8	137433	DEPT OF H...	Internal	1	Contract Analyst	PROCUREMENT ANAL...	12158	3	Finance, Accounti...	F
9	138531	DEPT OF E...	Internal	1	Associate Chemist	ASSOCIATE CHEMIST	21822	2	Health Public Safe...	F
10	151131	NYC HOUSI...	External	1	Cost Estimating M...	ADMINISTRATIVE STA...	1002D	0	Engineering, Archit...	F
11	152738	LAW DEPA...	Internal	1	Office Manager	CLERICAL ASSOCIATE	10251	3	Clerical & Adminis...	F
12	160910	DEPT OF IN...	Internal	1	Deputy Director, Au...	ADM MANAGER-NON-...	1002C	0	Finance, Accounti...	F

5.4 Create DataFrame from SQL Table and read the data using read_sql().

Program:

```
sql_df = pd.read_sql(
    "SELECT * FROM nyc_jobs",
    con=engine,
    parse_dates=[
        'created_at',
        'updated_at']
)
print(sql_df)
```

Output:



```
      Job ID                           Agency Posting Type  # Of Positions  \
0      87990    DEPARTMENT OF BUSINESS SERV.     Internal               1
1      97899    DEPARTMENT OF BUSINESS SERV.     Internal               1
2     132292           NYC HOUSING AUTHORITY     External              52
3     132292           NYC HOUSING AUTHORITY     Internal              52
4     177048    DEPT OF INFO TECH & TELECOMM     Internal               1
...      ...                             ...          ...             ...
2941  426214  HOUSING PRESERVATION & DVLPMNT     External               1
2942  426214  HOUSING PRESERVATION & DVLPMNT     Internal               1
2943  426223  HOUSING PRESERVATION & DVLPMNT     Internal               1
2944  426223  HOUSING PRESERVATION & DVLPMNT     External               1
2945  426238         DEPARTMENT OF BUILDINGS     Internal               1

                                         Business Title  \
0                                       Account Manager
1               EXECUTIVE DIRECTOR, BUSINESS DEVELOPMENT
2     Maintenance Worker - Technical Services-Heatin...
3     Maintenance Worker - Technical Services-Heatin...
```

PRACTICAL : 6

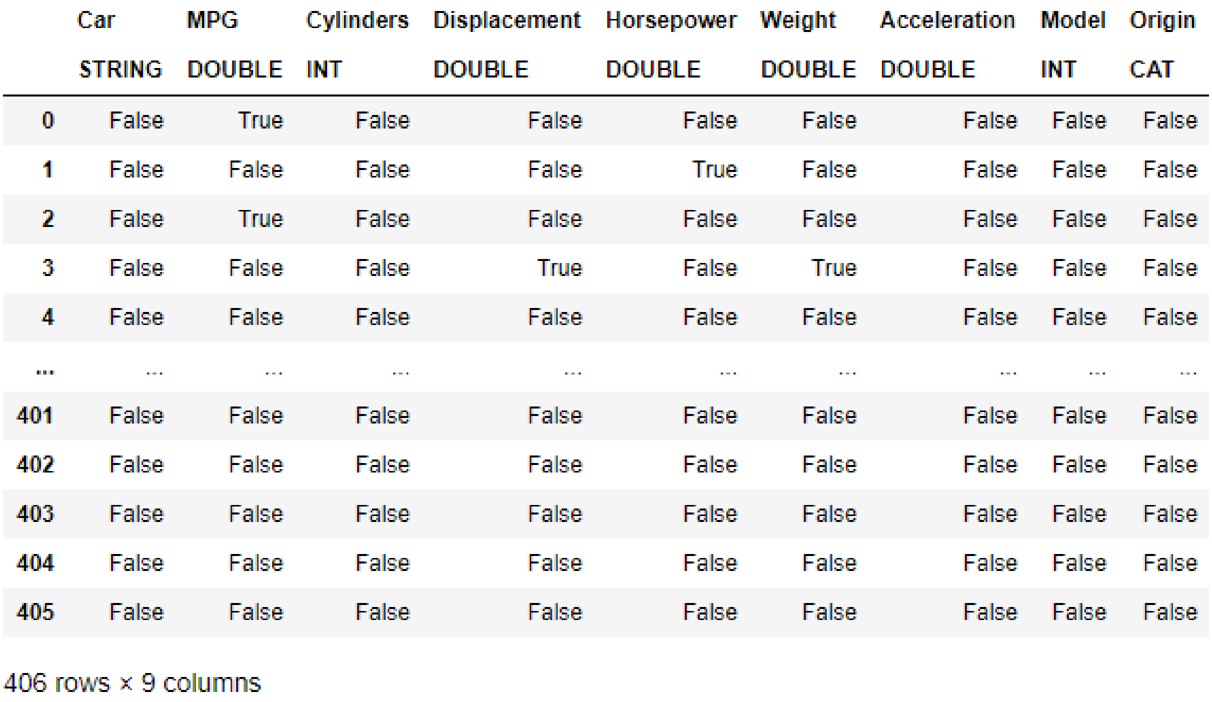
Aim : Handling Missing Data.

6.1 : Find the Missing Data in the given file cars.csv and print it.

Program:

```
import pandas as pd
df=pd.read_csv("res/cars.csv",sep=';',header=[0,1])
df.isnull()
```

Output:



	Car	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model	Origin
	STRING	DOUBLE	INT	DOUBLE	DOUBLE	DOUBLE	DOUBLE	INT	CAT
0	False	True	False	False	False	False	False	False	False
1	False	False	False	False	True	False	False	False	False
2	False	True	False	False	False	False	False	False	False
3	False	False	False	True	False	True	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
401	False	False	False	False	False	False	False	False	False
402	False	False	False	False	False	False	False	False	False
403	False	False	False	False	False	False	False	False	False
404	False	False	False	False	False	False	False	False	False
405	False	False	False	False	False	False	False	False	False

406 rows × 9 columns

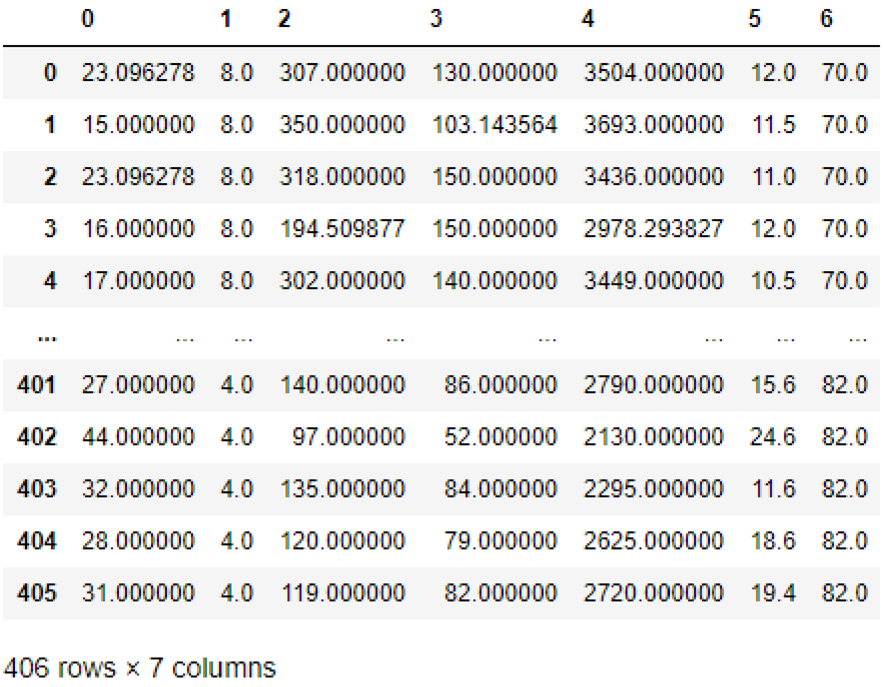
6.2 : Impute the missing data with all the methods (mean, median and most_frequent)

i) using mean() method:

Program:

```
import pandas as pd
from sklearn.impute import SimpleImputer

df=pd.read_csv("res/cars.csv",sep=';',header=[0,1])
df.drop(['Car','Origin'],axis='columns',inplace=True)
imp=SimpleImputer(strategy='mean')
imp.fit(df)
data=pd.DataFrame(imp.transform(df))
data
```

Output:



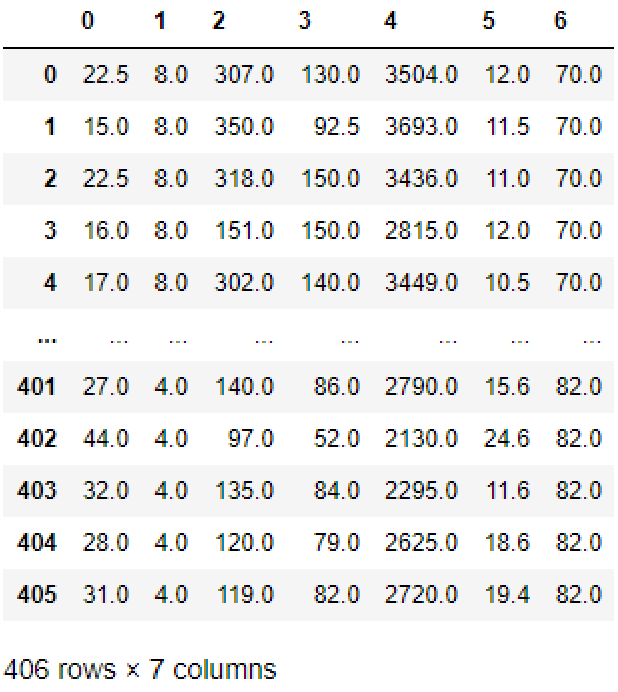
```
             0    1           2           3            4     5     6
  0  23.096278  8.0  307.000000  130.000000  3504.000000  12.0  70.0
  1  15.000000  8.0  350.000000  103.143564  3693.000000  11.5  70.0
  2  23.096278  8.0  318.000000  150.000000  3436.000000  11.0  70.0
  3  16.000000  8.0  194.509877  150.000000  2978.293827  12.0  70.0
  4  17.000000  8.0  302.000000  140.000000  3449.000000  10.5  70.0
...        ...  ...         ...         ...          ...   ...   ...
401  27.000000  4.0  140.000000   86.000000  2790.000000  15.6  82.0
402  44.000000  4.0   97.000000   52.000000  2130.000000  24.6  82.0
403  32.000000  4.0  135.000000   84.000000  2295.000000  11.6  82.0
404  28.000000  4.0  120.000000   79.000000  2625.000000  18.6  82.0
405  31.000000  4.0  119.000000   82.000000  2720.000000  19.4  82.0

406 rows × 7 columns
```

ii) using median() method:

Program:

```
import pandas as pd
from sklearn.impute import SimpleImputer

df=pd.read_csv("res/cars.csv",sep=';',header=[0,1])
df.drop(['Car','Origin'],axis='columns',inplace=True)
imp=SimpleImputer(strategy='median')
imp.fit(df)
data=pd.DataFrame(imp.transform(df))
data
```

Output:



	0	1	2	3	4	5	6
0	22.5	8.0	307.0	130.0	3504.0	12.0	70.0
1	15.0	8.0	350.0	92.5	3693.0	11.5	70.0
2	22.5	8.0	318.0	150.0	3436.0	11.0	70.0
3	16.0	8.0	151.0	150.0	2815.0	12.0	70.0
4	17.0	8.0	302.0	140.0	3449.0	10.5	70.0
...	...	...	...	...	...	...	...
401	27.0	4.0	140.0	86.0	2790.0	15.6	82.0
402	44.0	4.0	97.0	52.0	2130.0	24.6	82.0
403	32.0	4.0	135.0	84.0	2295.0	11.6	82.0
404	28.0	4.0	120.0	79.0	2625.0	18.6	82.0
405	31.0	4.0	119.0	82.0	2720.0	19.4	82.0

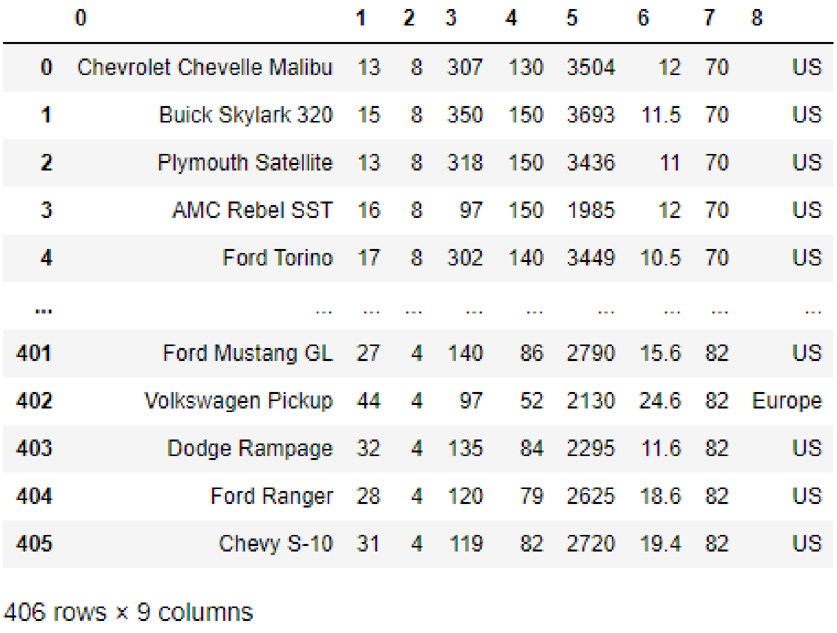
406 rows × 7 columns

iii) using most_frequent() method:

Program:

```
import pandas as pd
from sklearn.impute import SimpleImputer
df=pd.read_csv("res/cars.csv",sep=';',header=[0,1])
imp=SimpleImputer(strategy='most_frequent')
imp.fit(df)
data=pd.DataFrame(imp.transform(df))
data
```

Output:



	0	1	2	3	4	5	6	7	8
0	Chevrolet Chevelle Malibu	13	8	307	130	3504	12	70	US
1	Buick Skylark 320	15	8	350	150	3693	11.5	70	US
2	Plymouth Satellite	13	8	318	150	3436	11	70	US
3	AMC Rebel SST	16	8	97	150	1985	12	70	US
4	Ford Torino	17	8	302	140	3449	10.5	70	US
...	...	...	...	...	...	...	...	...	...
401	Ford Mustang GL	27	4	140	86	2790	15.6	82	US
402	Volkswagen Pickup	44	4	97	52	2130	24.6	82	Europe
403	Dodge Rampage	32	4	135	84	2295	11.6	82	US
404	Ford Ranger	28	4	120	79	2625	18.6	82	US
405	Chevy S-10	31	4	119	82	2720	19.4	82	US

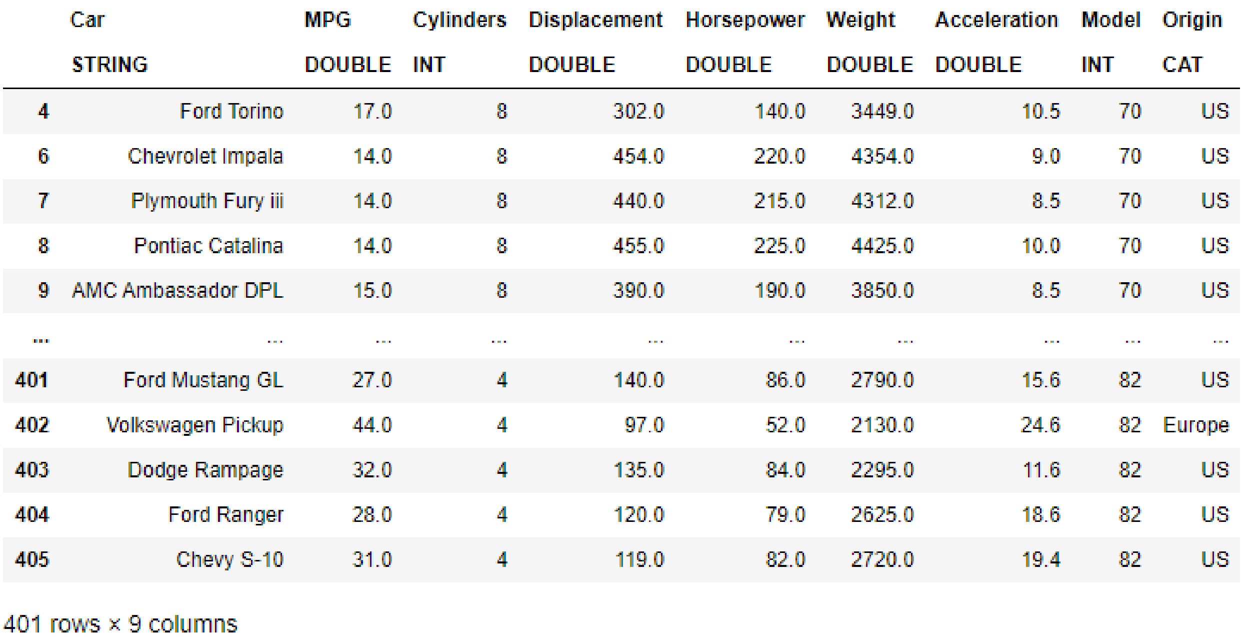
406 rows × 9 columns

6.3 : Delete all the missing data entry in cars.csv

Program:

```
import pandas as pd
df=pd.read_csv("res/cars.csv",sep=';',header=[0,1])
df.dropna()
```

Output:



	Car	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model	Origin
	STRING	DOUBLE	INT	DOUBLE	DOUBLE	DOUBLE	DOUBLE	INT	CAT
4	Ford Torino	17.0	8	302.0	140.0	3449.0	10.5	70	US
6	Chevrolet Impala	14.0	8	454.0	220.0	4354.0	9.0	70	US
7	Plymouth Fury iii	14.0	8	440.0	215.0	4312.0	8.5	70	US
8	Pontiac Catalina	14.0	8	455.0	225.0	4425.0	10.0	70	US
9	AMC Ambassador DPL	15.0	8	390.0	190.0	3850.0	8.5	70	US
...	...	...	...	...	...	...	...	...	...
401	Ford Mustang GL	27.0	4	140.0	86.0	2790.0	15.6	82	US
402	Volkswagen Pickup	44.0	4	97.0	52.0	2130.0	24.6	82	Europe
403	Dodge Rampage	32.0	4	135.0	84.0	2295.0	11.6	82	US
404	Ford Ranger	28.0	4	120.0	79.0	2625.0	18.6	82	US
405	Chevy S-10	31.0	4	119.0	82.0	2720.0	19.4	82	US

401 rows × 9 columns

PRACTICAL : 7

Aim: Working with Data Shaping

7.1 Apply bags of words model on the following statements.

- **Review 1: This movie is very scary and long**
- **Review 2: This movie is not scary and is slow**
- **Review 3: This movie is spooky and good**

What is the Bag of Words (BoW) model?

The Bag of Words (BoW) model is the simplest form of text representation in numbers. Like the term itself, we can represent a sentence as a bag of words vector (a string of numbers).
For the following reviews:
Review 1: This movie is very scary and long
Review 2: This movie is not scary and is slow
Review 3: This movie is spooky and good
Vocabulary is:
'This', 'movie', 'is', 'very', 'scary', 'and', 'long', 'not', 'slow', 'spooky', 'good'.
We can now take each of these words and mark their occurrence in the three movie reviews above with 1s and 0s. This will give us 3 vectors for 3 reviews:

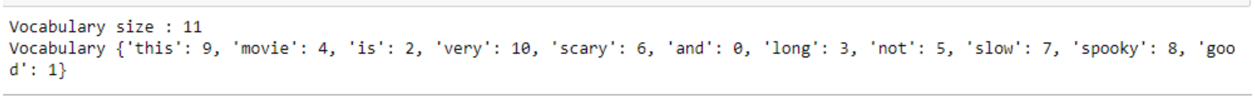
	1 **This**	**2** **movie**	**3** **is**	**4** **very**	**5** **scary**	**6** **and**	**7** **long**	**8** **not**	**9** **slow**	**10** **spooky**	**11** **good**	**TOTAL (in words)**
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	1	1	0	1	1	0	0	8
Review 3	1	1	1	0	0	1	0	0	0	1	1	6

Review 1 Vector: [1,1,1,1,1,1,1,0,0,0,0]
Review 2 Vector: [1,1,2,0,1,1,0,1,1,0,0]
Review 3 Vector: [1,1,1,0,0,1,0,0,0,1,1]

Program:

```
from sklearn.feature_extraction.text import *
import pandas as pd

phrase=["This movie is very scary and long","This movie is not scary and is slow","This movie is spooky and good"]
c_vect=CountVectorizer()
c_vect.fit_transform(phrase)
print('Vocabulary size :',len(c_vect.vocabulary_))
print('Vocabulary',c_vect.vocabulary_)
```

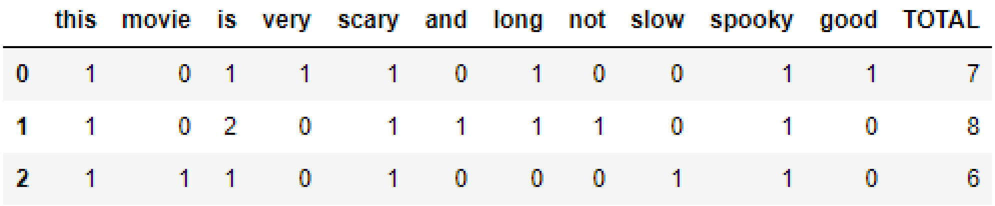
Output:



```
Vocabulary size : 11
Vocabulary {'this': 9, 'movie': 4, 'is': 2, 'very': 10, 'scary': 6, 'and': 0, 'long': 3, 'not': 5, 'slow': 7, 'spooky': 8, 'goo
d': 1}
```

```
import pandas as pd
import numpy as np

bag_of_words=pd.DataFrame(c_vect.transform(phrase).toarray(),columns=c_vect.vocabulary_)
bag_of_words.insert(11,'TOTAL','')
bag_of_words['TOTAL']=np.sum(bag_of_words,axis=1)
bag_of_words
```

Output:



	this	movie	is	very	scary	and	long	not	slow	spooky	good	TOTAL
0	1	0	1	1	1	0	1	0	0	1	1	7
1	1	0	2	0	1	1	1	1	0	1	0	8
2	1	1	1	0	1	0	0	0	1	1	0	6

7.2 Apply TF-IDF Model on the following statements.

- **Review 1: This movie is very scary and long**
- **Review 2: This movie is not scary and is slow**
- **Review 3: This movie is spooky and good**

<u>What is TF-IDF Model?</u>

The Term Frequency times Inverse Document Frequency (TF-IDF) transformation is a technique used to help compensate for words found relatively often in different documents, which makes it hard to distinguish between the documents because they are too common (stop words are a good example).

The greater the frequency of a word in a document, the more important it is to that document. However, the measurement is offset by the document size — the total number of words the document contains — and by how often the word appears in other documents.

We need the IDF value because computing just the TF alone is not sufficient to understand the importance of words.

$$TF = \frac{\text{(Number of repetitions of word in a document)}}{\text{(\# of words in a document)}}$$

1. **We will first calculate the TF values for all words,**

	TF		
words/document	**Review 1**	**Review 2**	**Review 3**
This	0.14	0.12	0.16
movie	0.14	0.12	0.16
Is	0.14	0.25	0.16
very	0.14	0.0	0.0
scary	0.14	0.12	0.0
and	0.14	0.12	0.16
long	0.14	0.0	0.0
not	0.0	0.12	0.0
slow	0.0	0.12	0.0
spooky	0.0	0.0	0.16
good	0.0	0.0	0.16

2. **<u>Calculating IDF values for all words</u>**

$$IDF = \frac{\text{Log[(\# Number of documents)}}{\text{(Number of documents containing the word)]}}$$

words/document	**IDF**
This	log(3/3) = 0.00
movie	log(3/3) = 0.00
is	log(3/3) = 0.00
very	log(3/1) = 0.48
scary	log(3/2) = 0.18
and	log(3/3) = 0.00
long	log(3/1) = 0.48
not	log(3/1) = 0.48
slow	log(3/1) = 0.48
spooky	log(3/1) = 0.48
good	log(3/1) = 0.48

Hence, we see that words like “is”, “this”, “and”, etc., are reduced to 0 and have little importance; while words like “scary”, “long”, “good”, etc. are words with more importance and thus have a higher value.

We can now compute the **TF-IDF score** for each word in the reviews. Words with a higher score are more important, and those with a lower score are less important:

TF-IDF = TF * IDF

First calculate Review 1:

TF-TDF(‘this’) = TF(‘this’, Review 1) * IDF(‘this’) = 0.14 * 0.00 = 0
TF-TDF(‘movie’) = TF(‘movie’, Review 1) * IDF(‘movie’) = 0.14 * 0.00 = 0
TF-TDF(‘is’) = TF(‘is’, Review 1) * IDF(‘is’) = 0.14 * 0.00 = 0.
TF-TDF(‘very’) = TF(‘very’, Review 1) * IDF(‘very’) = 0.14 * 0.48 = 0.068
TF-TDF(‘scary’) = TF(‘scary’, Review 1) * IDF(‘scary’) = 0.14 * 0.18 = 0.025
TF-TDF(‘and’) = TF(‘and’, Review 1) * IDF(‘and’) = 0.14 * 0.00 = 0
TF-TDF(‘long’) = TF(‘long’, Review 1) * IDF(‘long’) =0.14 * 0.48 = 0.067
TF-TDF(‘not’) = TF(‘not’, Review 1) * IDF(‘not’) = 0.00 * 0.48 = 0
TF-TDF(‘slow’) = TF(‘slow’, Review 1) * IDF(‘slow’) = 0.00 * 0.48 = 0
TF-TDF(‘spooky’) = TF(‘spooky’, Review 1) * IDF(‘spooky’) = 0.00 * 0.48 = 0
TF-TDF(‘good’) = TF(‘good’, Review 1) * IDF(‘good’) = 0.00 * 0.48 = 0

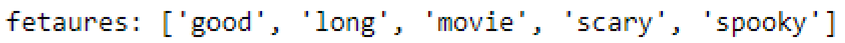
Similarly, we are calculating TF-IDF for all the words of all the reviews:

words	**Review 1**	**Review 2**	**Review 3**	**IDF**	**TF-IDF (Review 1)**	**TF-IDF (Review 2)**	**TF-IDF (Review 3)**
This	1	1	1	0.00	0.000	0.000	0.000
movie	1	1	1	0.00	0.000	0.000	0.000
is	1	2	1	0.00	0.000	0.000	0.000
very	1	0	0	0.48	0.068	0.000	0.000
Scary	1	1	0	0.18	0.025	0.021	0.000
And	1	1	1	0.00	0.000	0.000	0.000
Long	1	0	0	0.48	0.067	0.000	0.000
Not	0	1	0	0.48	0.000	0.060	0.000
Slow	0	1	0	0.48	0.000	0.060	0.000
spooky	0	0	1	0.48	0.000	0.000	0.080
Good	0	0	1	0.48	0.000	0.000	0.080

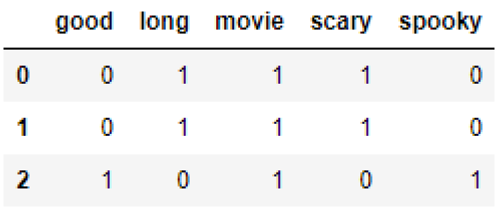
TF-IDF also gives larger values for less frequent words and is high when both IDF and TF values are high i.e. the word is rare in all the documents combined but frequent in a single document.

Program:

```
from sklearn.feature_extraction.text import *
import pandas as pd

phrases=["This movie is very scary and long","This movie is not scary and is long","This movie is spooky and good"]
c_vect=CountVectorizer(stop_words='english')
vect_data=c_vect.fit_transform(phrases)
print('fetaures:',c_vect.get_feature_names())
cv_df=pd.DataFrame(vect_data.toarray(),columns=c_vect.get_feature_names())
cv_df
```



```
fetaures: ['good', 'long', 'movie', 'scary', 'spooky']
```

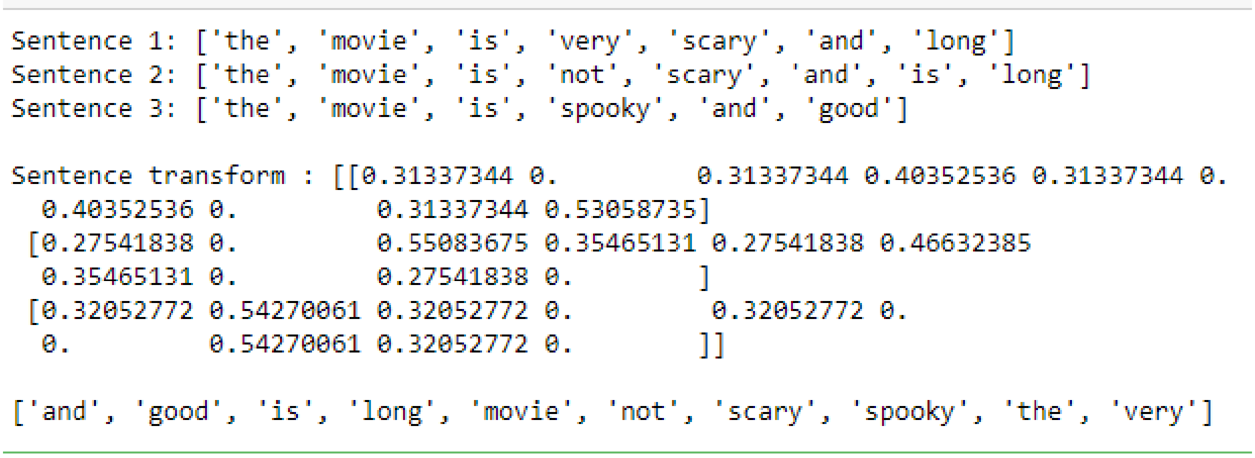


```
   good  long  movie  scary  spooky
0     0     1      1      1       0
1     0     1      1      1       0
2     1     0      1      0       1
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

sentence1="The movie is very scary and long"
sentence2="The movie is not scary and is long"
sentence3="The movie is spooky and good"
```

```
Doc = [sentence1 ,
       sentence2 ,
       sentence3]
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(Doc)
analyze = vectorizer.build_analyzer()
print('Sentence 1:',analyze(sentence1))
print('Sentence 2:',analyze(sentence2))
print('Sentence 3:',analyze(sentence3))
print()
print('Sentence transform :',X.toarray())
print()
print(vectorizer.get_feature_names())
```

Output:



```
Sentence 1: ['the', 'movie', 'is', 'very', 'scary', 'and', 'long']
Sentence 2: ['the', 'movie', 'is', 'not', 'scary', 'and', 'is', 'long']
Sentence 3: ['the', 'movie', 'is', 'spooky', 'and', 'good']

Sentence transform : [[0.31337344 0.         0.31337344 0.40352536 0.31337344 0.
  0.40352536 0.         0.31337344 0.53058735]
 [0.27541838 0.         0.55083675 0.35465131 0.27541838 0.46632385
  0.35465131 0.         0.27541838 0.        ]
 [0.32052772 0.54270061 0.32052772 0.         0.32052772 0.
  0.         0.54270061 0.32052772 0.        ]]

['and', 'good', 'is', 'long', 'movie', 'not', 'scary', 'spooky', 'the', 'very']
```

PRACTICAL : 8

Aim: Working with Data Visualization

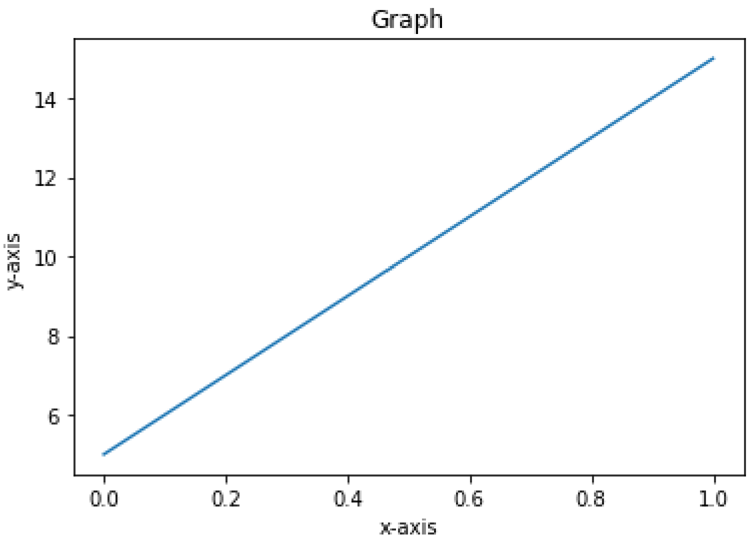
8.1: Plot a line graph:

You have to pass only one list of two points, which will be taken as y axis co-ordinates. For x axis it takes the default values in the range of 0 to 1, 2 being the length of the list [5, 15] and plot the graph.

Program:

```
import matplotlib.pyplot as plt
%matplotlib inline

plt.plot([5,15])
plt.title('Graph')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.show()
```

Output:



8.2: Plot line graph with multiple lines with label and legend for the following values:

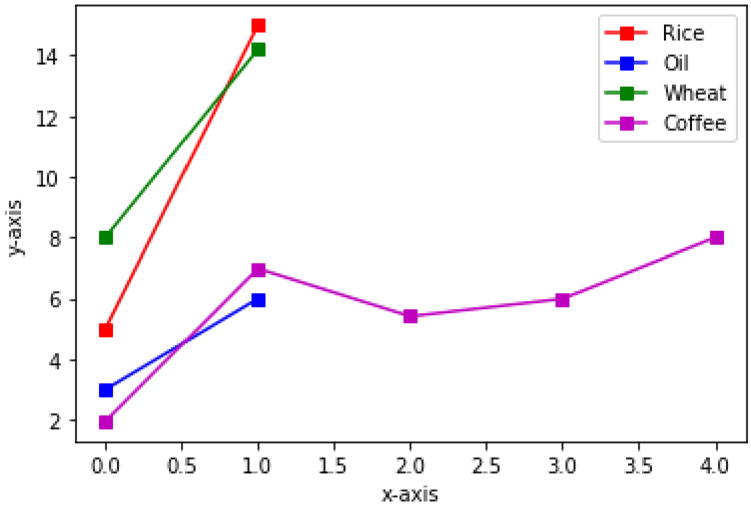
Rice	[5, 15]
Oil	[3, 6]
Wheat	[8.0010, 14.2]
Coffee	[1.95412, 6.98547, 5.41411, 5.99, 7.9999]

Also Mark the Line graph with Marker

Program:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot([5,15],'rs-',label='Rice')
plt.plot([3,6],'bs-',label='Oil')
plt.plot([8.0010,14.2],'gs-',label='Wheat')
plt.plot([1.95412,6.98547,5.41411,5.99,7.9999],'ms-',label='Coffee')
plt.legend()
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.show()
```

Output:

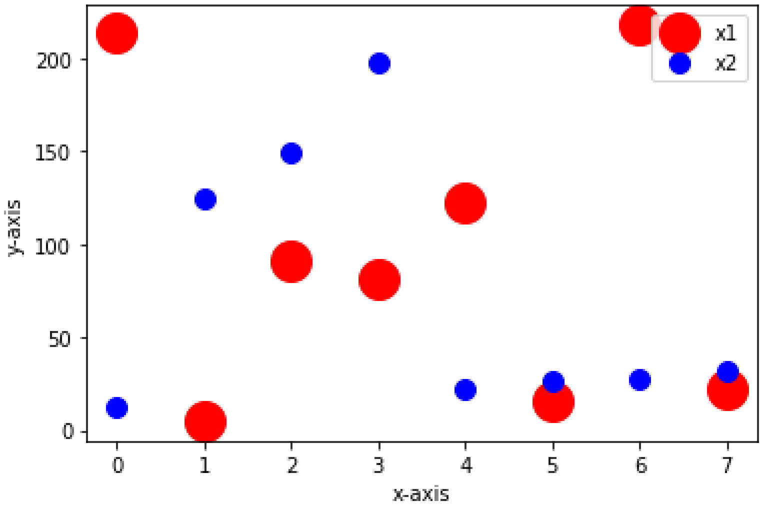


8.3: Plot scatter with marker size of 20 and 10 for x1 and x2 for the following data:
x1 = [214, 5, 91, 81, 122, 16, 218, 22]
x2 = [12, 125, 149, 198, 22, 26, 28, 32]

Program:

```
import matplotlib.pyplot as plt
%matplotlib inline

x1 = [214, 5, 91, 81, 122, 16, 218, 22]
x2 = [12, 125, 149, 198, 22, 26, 28, 32]

plt.plot(x1,'ro',markersize=20,label='x1')
plt.plot(x2,'bo',markersize=10,label='x2')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.legend()
plt.show()
```

Output:



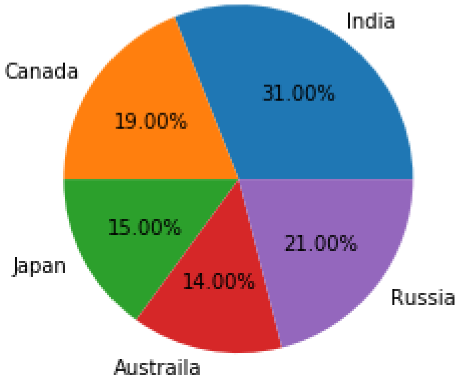
8.4: Plot pie graph for the following values and also turn on the axis of graph.

India	31%
Canada	19%
Japan	15%
Australia	14%
Russia	21%

Program:

```
import matplotlib.pyplot as plt
%matplotlib inline

labels=['India','Canada','Japan','Austraila','Russia']
size=[31,19,15,14,21]

plt.pie(size,labels=labels,autopct='%1.2f%%')
plt.show()
```

Output:

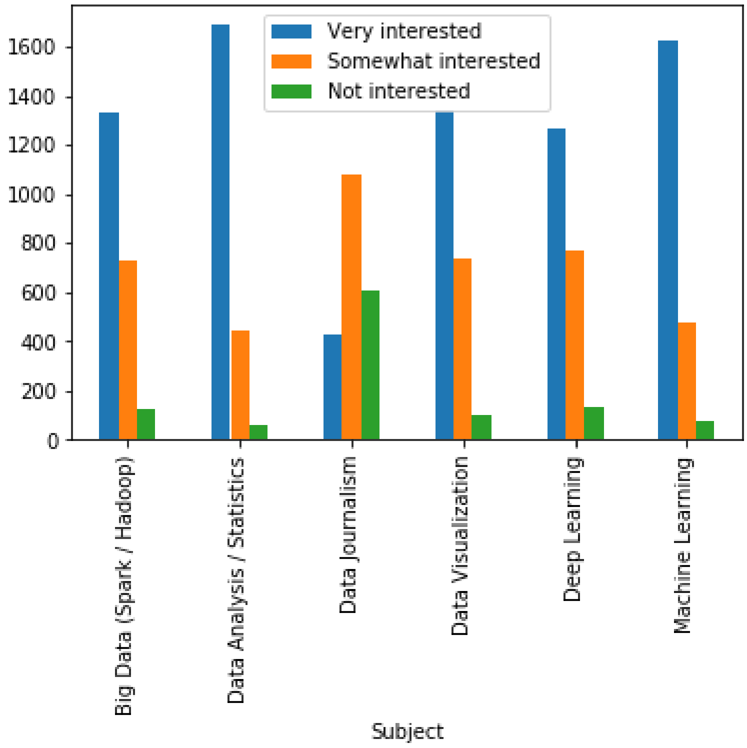


8.5: Read the data from Topic_Survey_Assignment.csv file and Plot the bar graph. Also Put the percentage values on top of each bar

Program:

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

df = pd.read_csv('res/Topic_Survey_Assignment.csv')
df.rename(columns={'Unnamed: 0': 'Subject'}, inplace=True)

df.plot.bar(x='Subject', y=['Very interested', 'Somewhat interested','Not interested'])
```

Output:

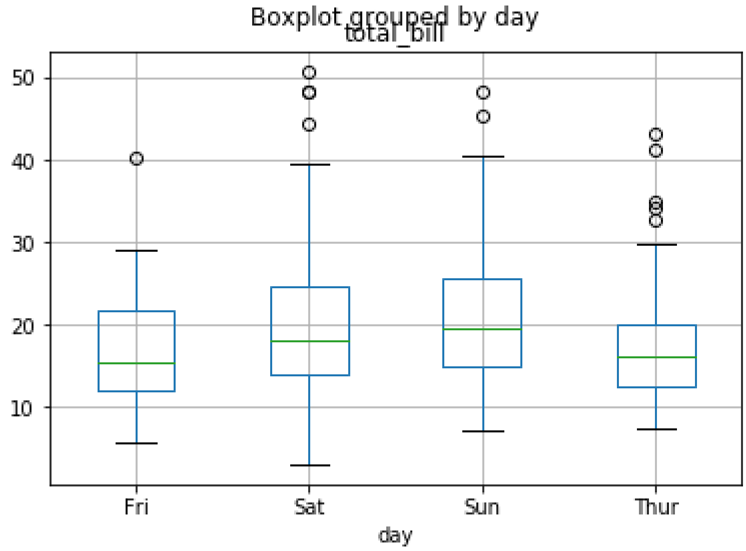


8.6: Read the data from tips.csv file and Plot the box graph. plot day on X-Axis and ploy total_bill on Y-Axis

Program:

```
import matplotlib.pyplot as plt
import pandas as pd
%matplotlib inline

tips=pd.read_csv('res/tips.csv')

tips.boxplot(by='day',column=['total_bill'])
```

Output:

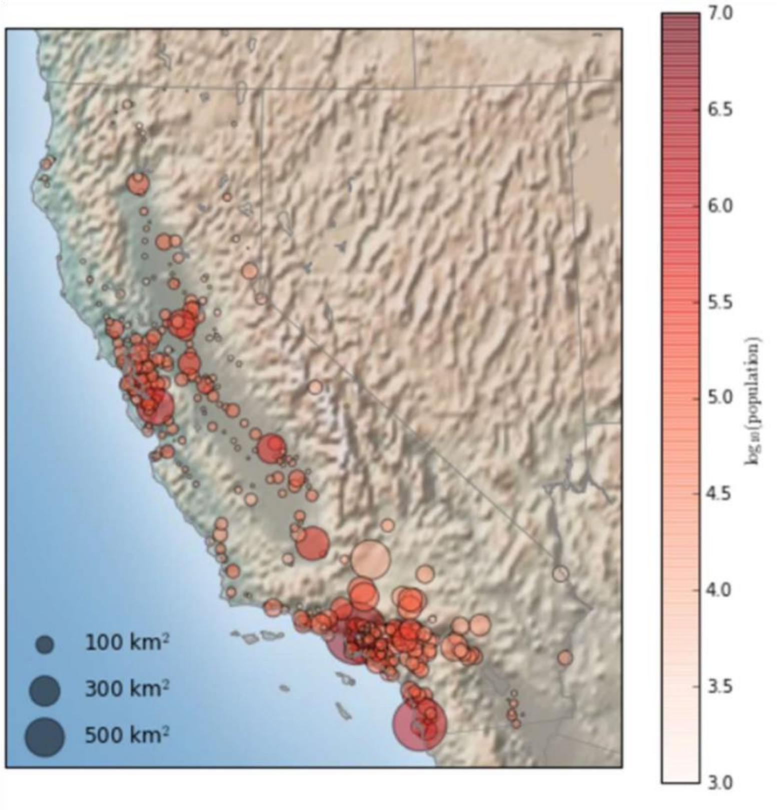


8.7: Plot the Geographic Data from california_cities.csv file using basemap.

Program:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap

cities = pd.read_csv('res/california_cities.csv')
lat = cities['latd'].values
lon = cities['longd'].values
population = cities['population_total'].values
area = cities['area_total_km2'].values

fig = plt.figure(figsize=(8, 8))
m = Basemap(projection='lcc',resolution='h',lat_0=37.5, lon_0=-119,width=1E6,height=1.2E6)
m.shadedrelief()
m.drawcoastlines(color='gray')
m.drawcountries(color='gray')
m.drawstates(color='gray')
m.scatter(lon, lat, latlon=True,c=np.log10(population),s=area,cmap='Reds', alpha=0.5)

plt.colorbar(label=r'$\log_{10}({\rm population})$')
plt.clim(3, 7)
for a in [100, 300, 500]:
    plt.scatter([], [], c='k', alpha=0.5, s=a,label=str(a) + ' km$^2$')
plt.legend(scatterpoints=1, frameon=False,labelspacing=1, loc='lower left')
plt.show()
```

Output:

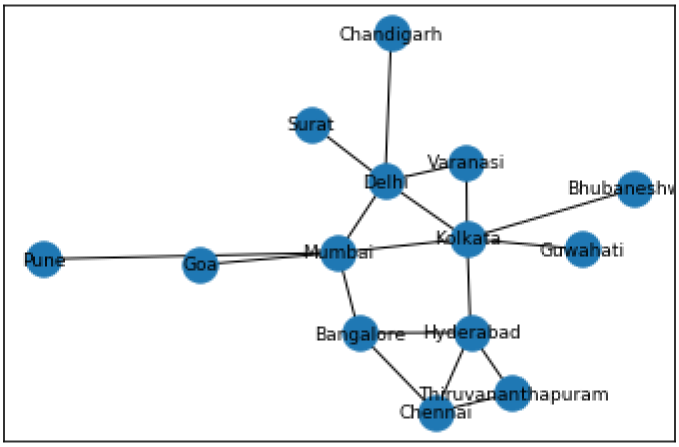

100 km^2
300 km^2
500 km^2
7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
$\log_{10}$(population)

8.8: Here is a Dataset of various Indian cities and the distances between them in edge_list.txt. Draw the Graph for the data.

Program:

```
import networkx as nx
import pandas as pd
%matplotlib inline

df=pd.read_csv('res/edge_list.txt',delim_whitespace=True,header=None,names=['n1','n2','weight'])
graph=nx.read_weighted_edgelist('res/edge_list.txt',delimiter=" ")
nx.draw_networkx(graph,with_label=True,font_size=9)
```

Output:



PRACTICAL : 9

Aim: Load the Boston dataset from sklearn library concerns the housing prices in housing city of Boston. The dataset provided has 506 instances with 13 features. Split the data into training and testing sets. Train the model with 80% of the samples and test with the remaining 20%.

Predict the house prices for testing dataset.

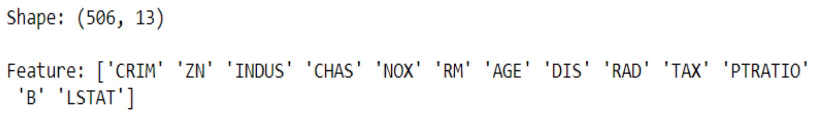
- **Importing Libraries and Loading boston dataset**

```
import numpy as np
import pandas as pd
#Visualization Libraries
import matplotlib.pyplot as plt
#To plot the graph embedded in the notebook
%matplotlib inline
# loading dataset
from sklearn.datasets import load_boston
boston = load_boston()
```

- **Boston dataset Shape and Features:**

```
print("Shape:",boston.data.shape )
print("Feature:",boston.feature_names )
```

Output:

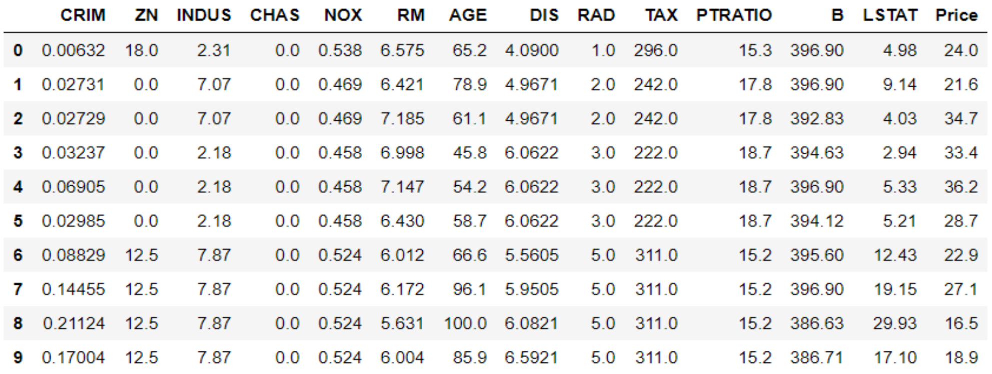


```
Shape: (506, 13)

Feature: ['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

- **Converting data from nd-array to dataframe and adding feature names and 'Price' column to the dataset**

```
data = pd.DataFrame(boston.data)
data.columns = boston.feature_names
data['Price'] = boston.target
data.head(10)
```

Output:

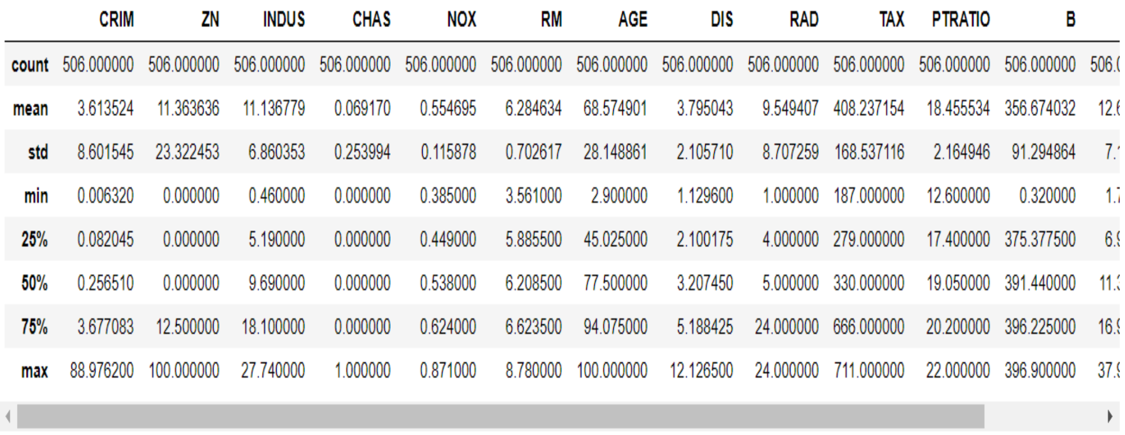


	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2
5	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3.0	222.0	18.7	394.12	5.21	28.7
6	0.08829	12.5	7.87	0.0	0.524	6.012	66.6	5.5605	5.0	311.0	15.2	395.60	12.43	22.9
7	0.14455	12.5	7.87	0.0	0.524	6.172	96.1	5.9505	5.0	311.0	15.2	396.90	19.15	27.1
8	0.21124	12.5	7.87	0.0	0.524	5.631	100.0	6.0821	5.0	311.0	15.2	386.63	29.93	16.5
9	0.17004	12.5	7.87	0.0	0.524	6.004	85.9	6.5921	5.0	311.0	15.2	386.71	17.10	18.9

➢ Description of Boston dataset

data.describe()

Output:

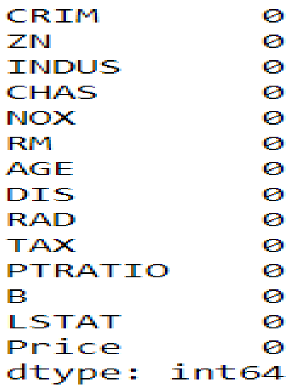


	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.0
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.6
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.1
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.7
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.9
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.3
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.9
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.9

➢ Data Preprocessing

data.isnull().sum()

Output:

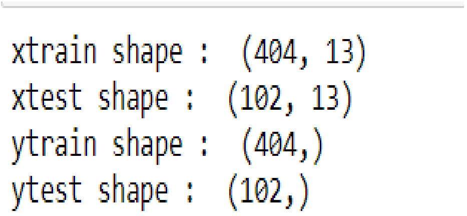


```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
Price      0
dtype: int64
```

➢ Splitting data to training and testing dataset.

```
# Input Data
x = boston.data
# Output Data
y = boston.target
# splitting data to training and testing dataset.
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size =0.2, random_state = 0)
print("xtrain shape : ", xtrain.shape)
print("xtest shape : ", xtest.shape)
print("ytrain shape : ", ytrain.shape)
print("ytest shape : ", ytest.shape)
```

Output:



```
xtrain shape :  (404, 13)
xtest shape :  (102, 13)
ytrain shape :  (404,)
ytest shape :  (102,)
```

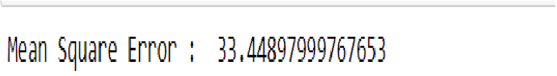
- **Training Model**

```
# Fitting Multi Linear regression model to training model
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(xtrain, ytrain)
# predicting the test set results
y_pred = regressor.predict(xtest)
```

- **Checking Model Accuracy**

```
# Results of Linear Regression.
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(ytest, y_pred)
print("Mean Square Error : ", mse)
```

Output:

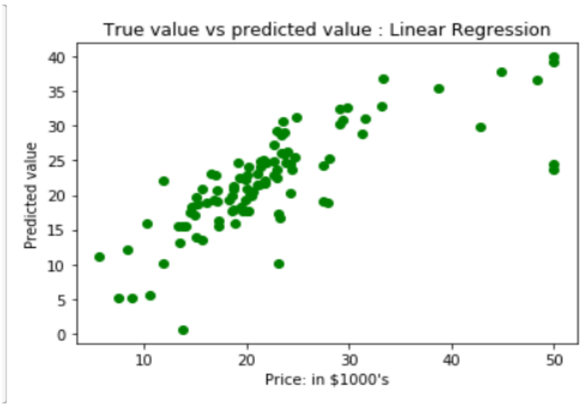


```
Mean Square Error :  33.448979997676533
```

- **Showing the prediction of house prices**

```
# Plotting Scatter graph to show the prediction
# results - 'ytrue' value vs 'y_pred' value
plt.scatter(ytest, y_pred, c = 'green')
plt.xlabel("Price: in $1000's")
plt.ylabel("Predicted value")
plt.title("True value vs predicted value : Linear Regression")
plt.show()
```

Output:


True value vs predicted value : Linear Regression
Predicted value
Price: in $1000's