PRACTICAL-01

Develop programs to understand the control structures of python

Practical 1.1

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

Program :

```
n = []

for x in range(1500,2701):

    if(x%7==0)and(x%5==0):

        n.append(str(x))


print(n,end="")
```

OUTPUT :

```
>>>
========== RESTART: E:\Python Practical\prac_1.py ==========
['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', '2380', '2415', '2450', '2485', '2520',
 '2555', '2590', '2625', '2660', '2695']
>>>
```

Practical 1.2

AIM : Write a Python program to construct the following pattern, using nested for loop.

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

Program :

```
y=5
for i in range(0,y):
    for j in range(0,i+1):
            print("*",end="")
print("\r")
for i in range(y,0,-1):
    for j in range(0,i-1):
            print("*",end="")
print("\r")
```

OUTPUT :

```
==================== RESTART: E:\Python Practical\prac_2.py ====================
*
**
***
****
*****
****
***
**
*
```

Practical 1.3

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

Program :

```
str = input("Enter a word: ")

rev = " "

for i in str:

    rev = i +rev

print("Reverse word:",rev)
```

OUTPUT :

```
==================== RESTART: E:\Python Practical\prac_3.py ====================
Enter a word: PYTHON
Reverse word: NOHTYP
>>>
```

Practical 1.4

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

Program :

```
ch = input("Enter a Character : ")


if(ch=='A' or ch=='a' or ch=='E' or ch=='e' or ch=='I' or ch=='i' or ch=='O' or ch=='o' or ch=='U' or ch=='u'):
    print(ch,"is Wovel.")


else:
    print(ch,"is Constant.")
```

OUTPUT :

```
==================== RESTART: E:/Python Practical/prac_4.py =========
Enter a Character : a
a is Wovel.
>>>
```

```
==================== RESTART: E:/Python Practical/prac_4.py ===========
Enter a Character : D
D is Constant.
>>>
```

Practical 1.5

AIM : Write a Python program to find reverse of given number.

Program :

```
n=int(input("Enter number: "))
rev=0
while(n>0):
    dig=n%10
    rev=rev*10+dig
    n=n//10
print("Reverse of the number:",rev)
```

OUTPUT :

```
========== RESTART: E:/Python Practical/prac_5.py ==========
Enter number: 5281
Reverse of the number: 1825
>>>
```

```
========== RESTART: E:/Python Practical/prac_5.py ==========
Enter number: 8171
Reverse of the number: 1718
>>>
```

Practical 1.6

AIM : Write a Python program to check whether the given no is Armstrong or not using .

Program :

```
n = int(input("Enter a number : "))

sum = 0

temp = n

while temp > 0:

    d = temp % 10

    sum += d ** 3

    temp //= 10

if n == sum:

    print(n,"is an Amstrong Number.")

else:

    print(n,"is not a Amstrong Number.")
```

OUTPUT :

```
========== RESTART: E:/Python Practical/prac_6.py ==========
Enter a number : 153
153 is an Amstrong Number.
>>>
```

```
========== RESTART: E:/Python Practical/prac_6.py ==========
Enter a number : 663
663 is not a Amstrong Number.
>>>
```

Practical 1.7

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

Program :

```
n = int(input("Enter a number: "))
count = 1
s = 2
while count <=n-1:
    s += 1
    for i in range(2,s):
        if s%i==0:
            break
    else:
        if i==2:
            print(i,end=' ')
        i += 1
        print(i,end=' ')
        count += 1
```

OUTPUT :

```
========== RESTART: E:\Python Practical\prac_7.py ==========
Enter a number: 10
2 3 5 7 11 13 17 19 23 29
>>>
```

Practical 1.8

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

Program :

```
n = int(input("Enter the number of terms needed in fibonaci series: "))
f1, f2 = 0, 1
if n == 1:
   print(f1)
elif n == 2:
   print(f1,f2)
else:
   print(f1,f2,end=' ')
   for i in range(3,n+1):
      f3 = f1 + f2
      print(f3,end=' ')
      f1 = f2
      f2 = f3
```

OUTPUT :

```
========== RESTART: E:\Python Practical\prac_8.py ==========
Enter the number of terms needed in fibonaci series: 10
0 1 1 2 3 5 8 13 21 34
>>>
```

Practical 1.9

AIM : Give the output of following Python code:

a) myStr = 'GTU is the best University'

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

OUTPUT :

```
=================== RESTART: E:/Python Practical/prac_9_a.py ================
 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 :

```
>>>
=================== RESTART: E:/Python Practical/prac_9_c.py ==========
[(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 :

```
========== RESTART: E:/Python Practical/prac_9_d.py ==========
Slice of String :  his
Slice of String :  Ti sPto
>>>
```

PRACTICAL-02

Develop programs to learn different types of structures

(list, dictionary, tuples) in python.

Practical 2.1

Aim : 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 a 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("maximum element=",max)
```

Output :

```
========== RESTART: E:\Python Practical\prac_2_1.py ==========
Enter size of list:5
Enter a elements:
10
20
85
45
100
list= [10, 20, 85, 45, 100]
maximum element= 100
>>>
```

Practical 2.2

Aim : 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 after13 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 element:"))
lst=[]
print("Enter a 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 :

```
...
========== RESTART: E:\Python Practical\prac_2_2.py ==========
Enter size of element:5
Enter a elements:
1
2
3
4
5
list= [1, 2, 3, 4, 5]
Sum = 15
>>>
```

```
========== RESTART: E:\Python Practical\prac_2_2.py ==========
Enter size of element:5
Enter a elements:
1
2
3
4
13
list= [1, 2, 3, 4, 13]
Sum = 10
>>>
```

Practical 2.3

Aim : 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 f element:"))

lst=[]

print("Enter elements:")

for i in range(0,n):

    num=int(input())

    lst.append(num)

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 shfted left by one position=",lst)
```

Output :

```
========== RESTART: E:\Python Practical\prac_2_3.py ==========
Enter size f element:4
Enter elements:
11
12
13
14
list= [11, 12, 13, 14]

After shfted left by one position= [12, 13, 14, 11]
>>>
```

Practical 2.4

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

Program :

```
lst = ['R','N','G','P','I','T']

print("list :",lst)

lst1 = "".join(lst)

print("After convert a list of character into string ->",lst1)
```

Output :

```
>>>
========== RESTART: E:\Python Practical\prac_2_4.py ==========
list : ['R', 'N', 'G', 'P', 'I', 'T']
After convert a list of character into string -> RNGPIT
>>>
```

Practical 2.5

Aim : Write a Python program

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

Program :

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

print(lst[5:])
```

Output :

```
>>>
================== RESTART: E:\Python Practical\prac_2_5_1.py ==================
[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 between 1 and 30.**

Program :

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

lst1=lst[0:5]

lst2=lst[25:]

lst3=lst1 + lst2

print(lst3)
```

Output :

```
>>>
================== RESTART: E:\Python Practical\prac_2_5_2.py ==============
[1, 4, 9, 16, 25, 676, 729, 784, 841, 900]
>>>
```

Practical 2.6

Aim : 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 the element:")
for i in range(0,n):
    num=int(input())
    lst.append(num)
print("Original list= ",lst)


lst1=[]
for i in lst:
    if(i%2 == 0):
        lst.remove(i)
print("list after removing even numbers=",lst)
```

Output :

```
>>>
========== RESTART: E:\Python Practical\prac_2_6.py ==========
Enter size of list:5
Enter the element:
10
11
15
18
20
Original list=  [10, 11, 15, 18, 20]
list after removing even numbers= [11, 15, 20]
>>>
```

Practical 2.7

Aim : 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 : ")
d = {}
for i in str:
    if i in d:
        d[i] += 1
    else:
        d[i] = 1

print("count the numbers of characters in the string and store them in a dictionary data structure :",d)
```

Output :

```
>>>
==================== RESTART: E:/Python Practical/prac_2_7.py ====================
enter string : MATPLOTLIB
count the numbers of characters in the string and store them in a dictionary dat
a structure : {'M': 1, 'A': 1, 'T': 2, 'P': 1, 'L': 2, 'O': 1, 'I': 1, 'B': 1}
>>>
```

Practical 2.8

Aim : Write a program to use split and join methods in the string and trace a birthday with a dictionary datastructure.

Program :

```
birthdays = {'Dharmik': '18 11 2001','Sunny': '28 01 2001','Shuchi': '17 01 2000','Neel':'15 02 2000'}

name=input("Enter a name:")

if name in birthdays:

 s=birthdays[name];

 L=s.split()

 k="/"

 L=k.join(L)

 print(L + ' is the birthday of ' + name)

else:

 print('I do not have birthday information for ' + name)
```

Output :

```
========== RESTART: E:\Python Practical\prac_2_8.py ==========
Enter a name :Dharmik
18/11/2001 is the birthday of Dharmik
>>>

========== RESTART: E:\Python Practical\prac_2_8.py ==========
Enter a name :rushabh
I do not have birthday information for rushabh
>>>
```

Practical 2.9

Aim : Write a python program to sort a dictionary by value

Program :

```
a = {13:2 ,22:1 ,54:3 ,43:4 ,86:5 ,95:6 }

b=dict(sorted(a.items(),key=lambda a:a[1]))

print("dict is sorted by values:\n",b)
```

Output:

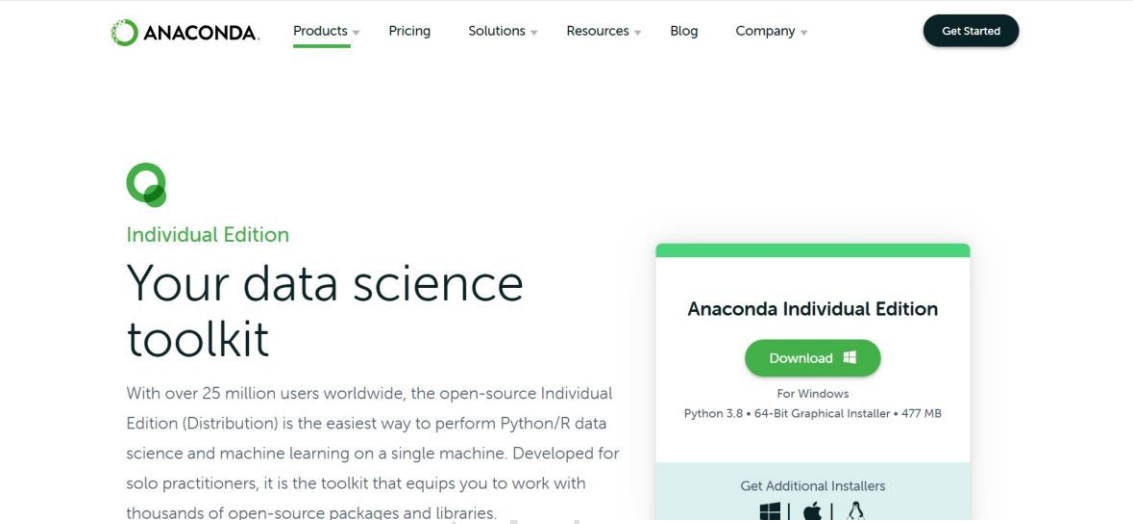
```
========== RESTART: E:/Python Practical/prac_2_9.py ========
dict is sorted by values:
 {22: 1, 13: 2, 54: 3, 43: 4, 86: 5, 95: 6}
>>>
```

PRACTICAl-03

Aim :- Setting up Python for Data Science.

Practical 3.1 :- Installing Anaconda on Windows.

1. **Visit the Anaconda downloads page.**
   Go to the following link: Anaconda.com/downloads
   The Anaconda Downloads Page will look something like this:

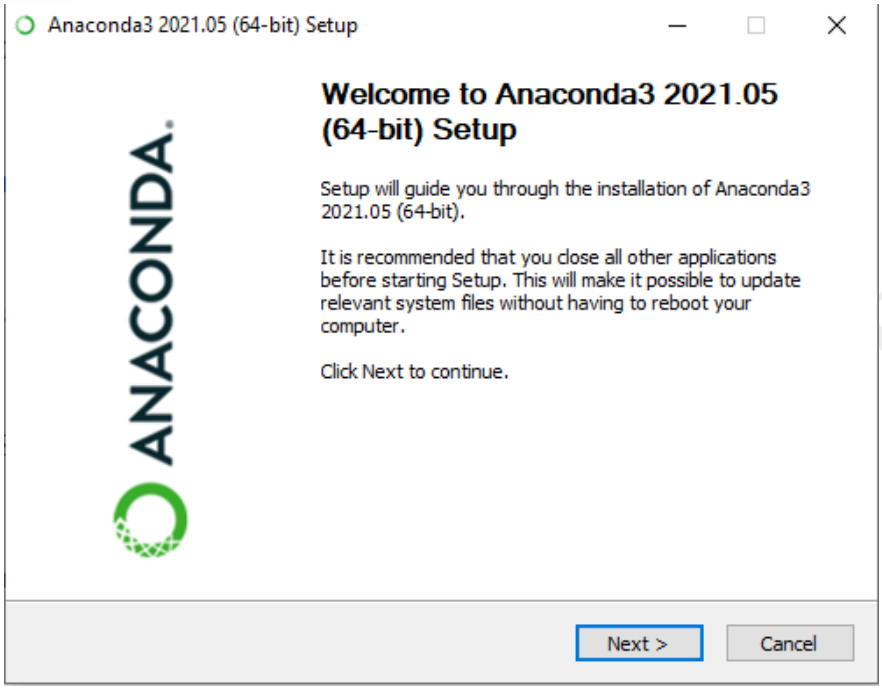


2. **Select Windows.**
   Select Windows where the three operating systems are listed.

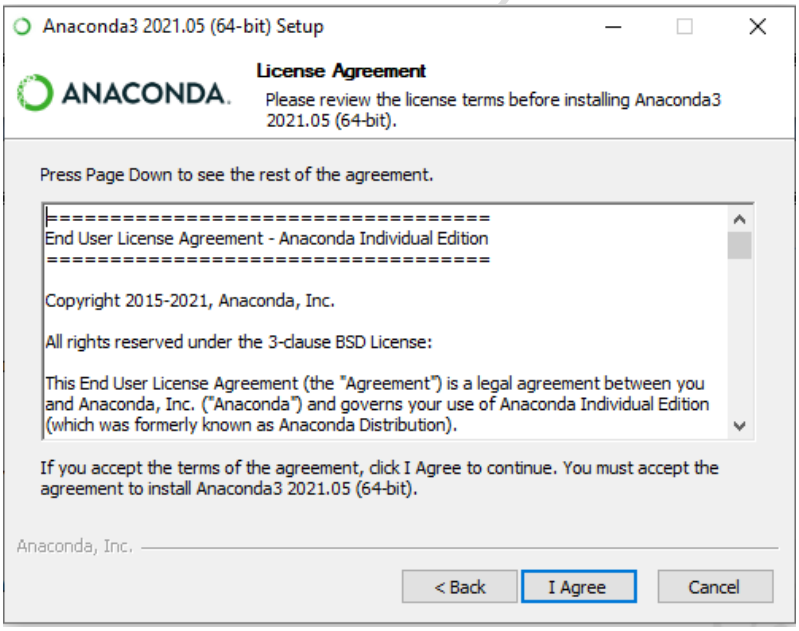
3. At the beginning of the install.

Once the download completes, open and run the .exe installer

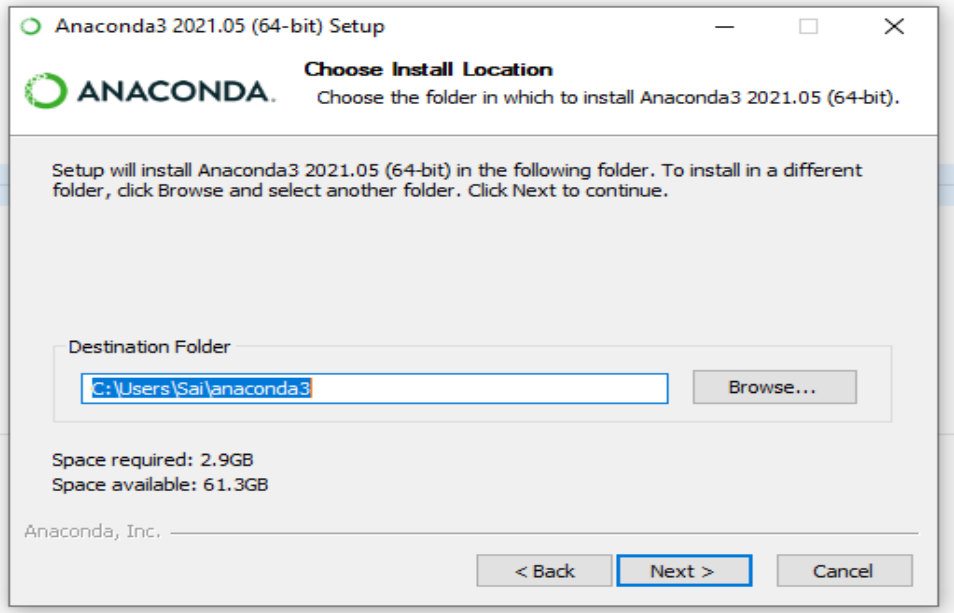
At the beginning of the install, you need to click Next to confirm the installation.



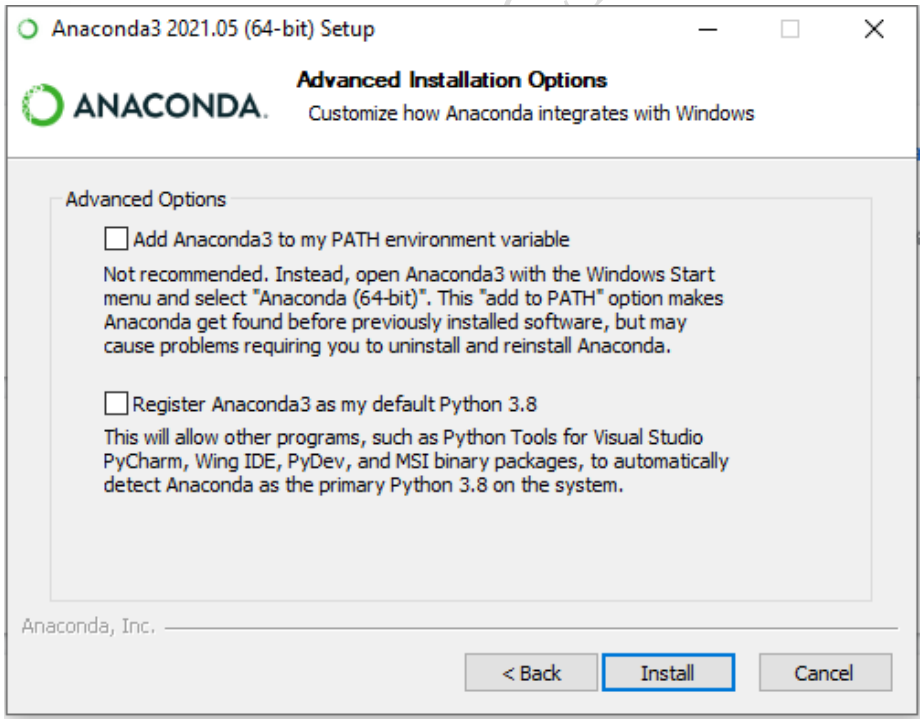
4. Agree To The License.



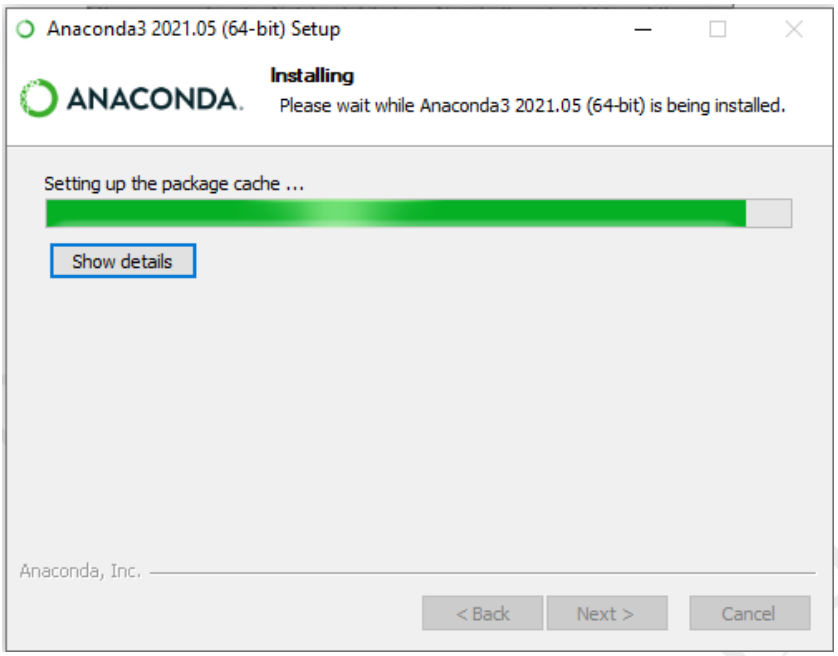
5. Write A Destination Folder And Than Click Next.



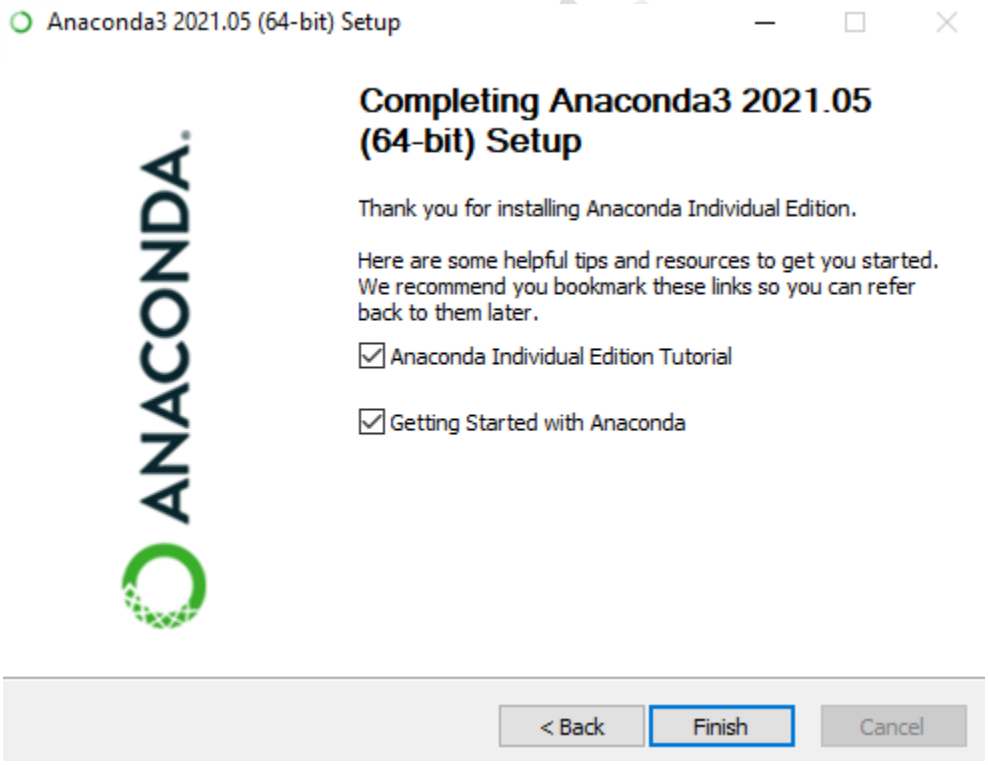
6. Select advanced Installation options.



7. Installing Anaconda.

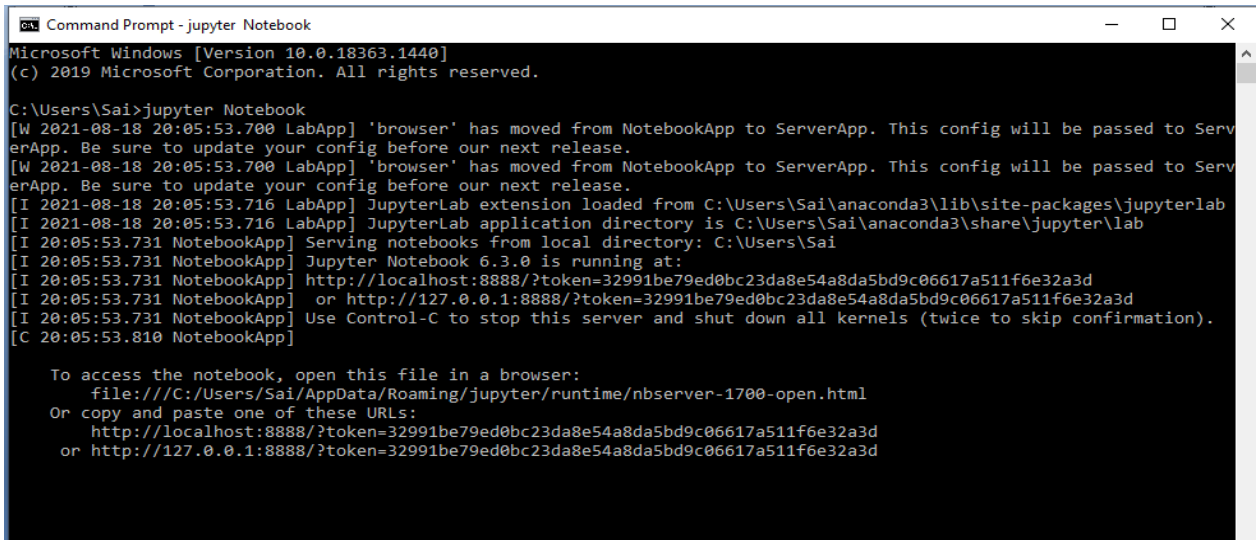


8. Finished The Installation.

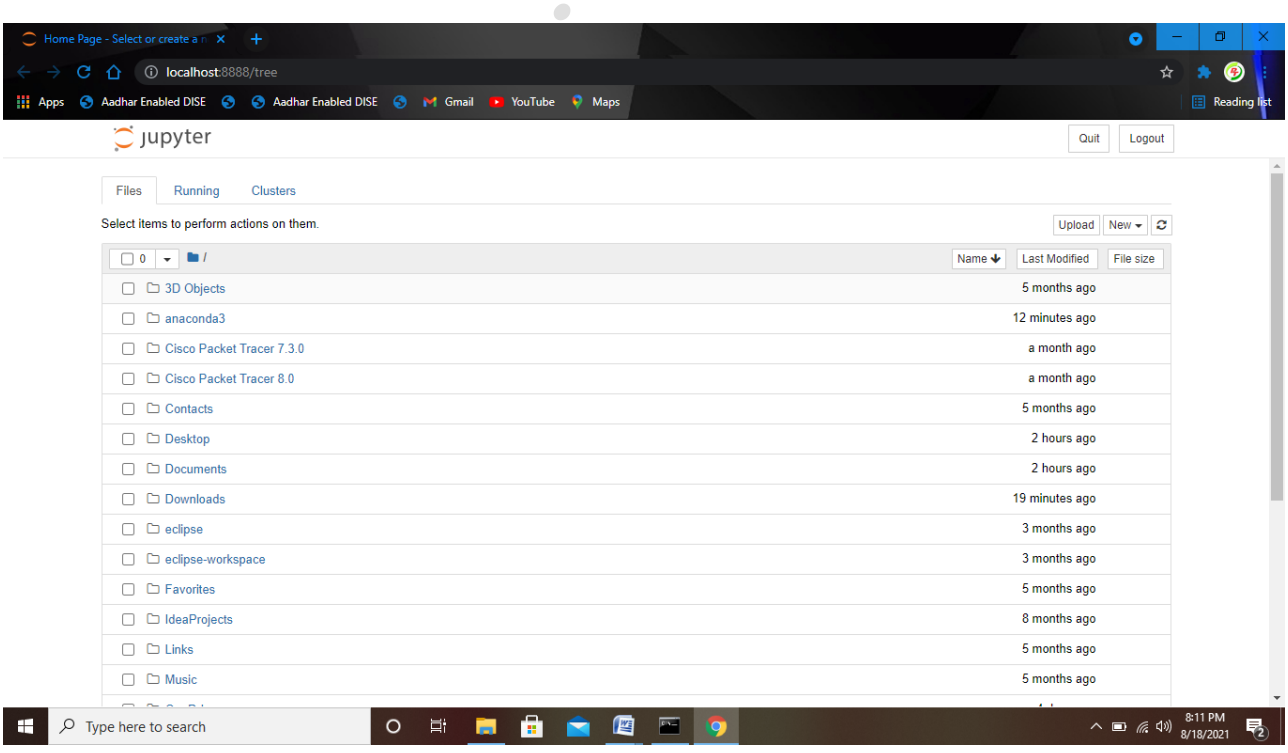


Practical 3.2 :- Working with Jupyter Notebook

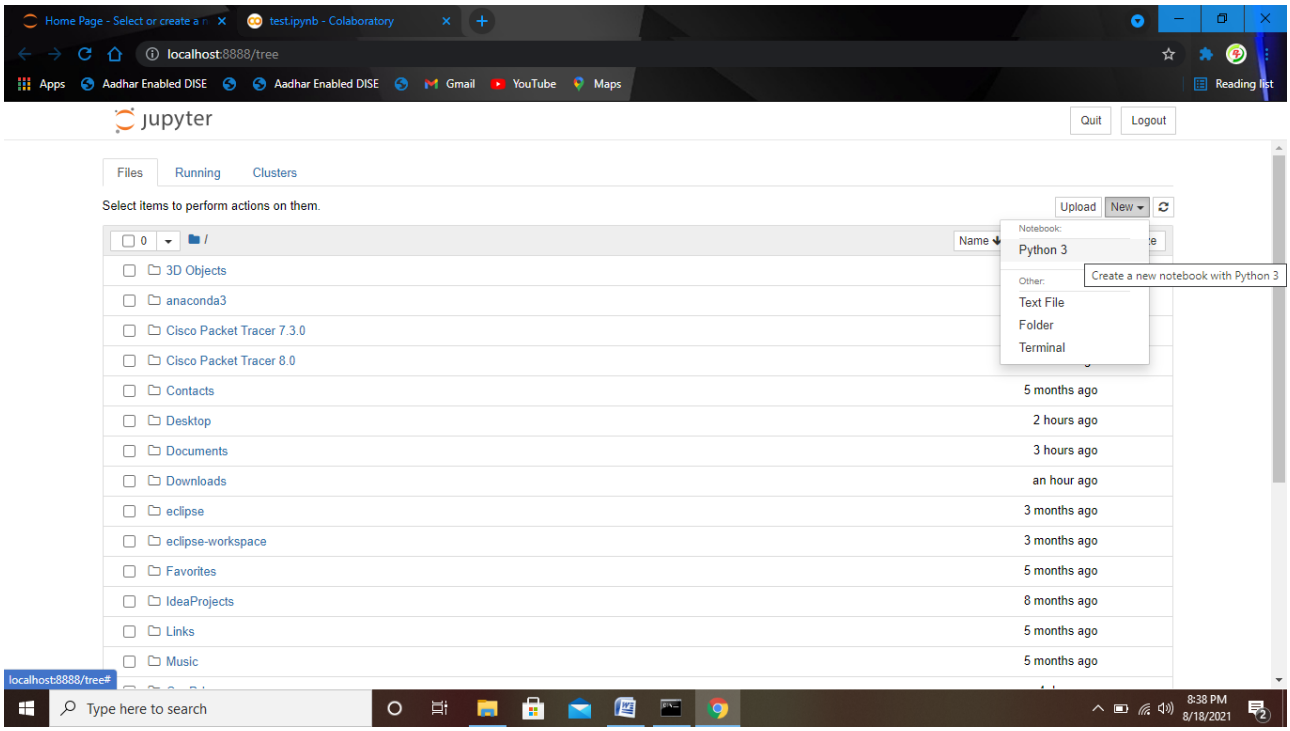
- **Open The Jupyter Notebook (Anaconda3) From The Anaconda Prompt.**



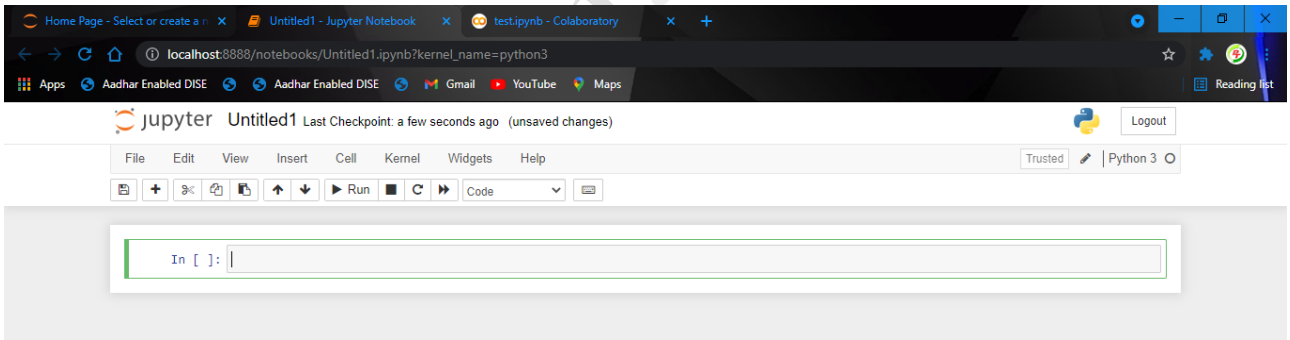
- **Open Jupyter Notebook**



- **After opening jupyter notebook click on new button at right top of the screen and select python 3.**

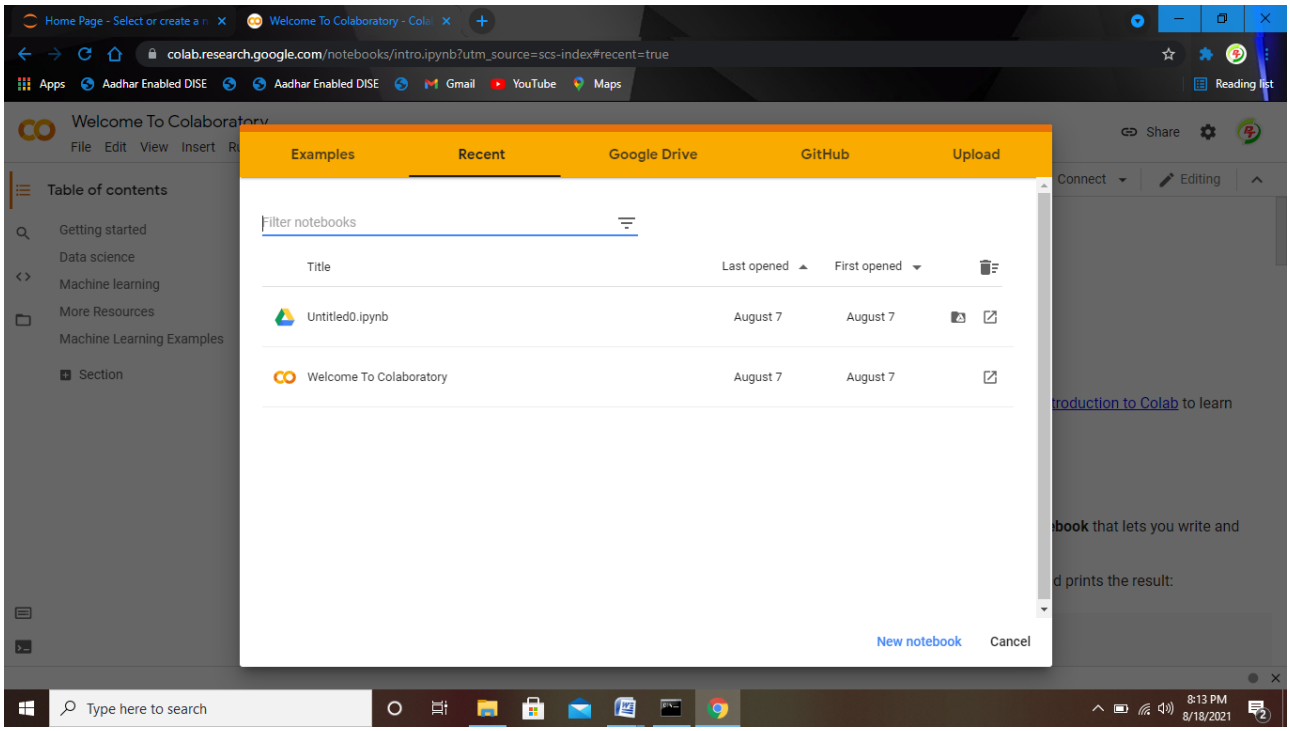


- **Example Run In Jupyter Notebook.**



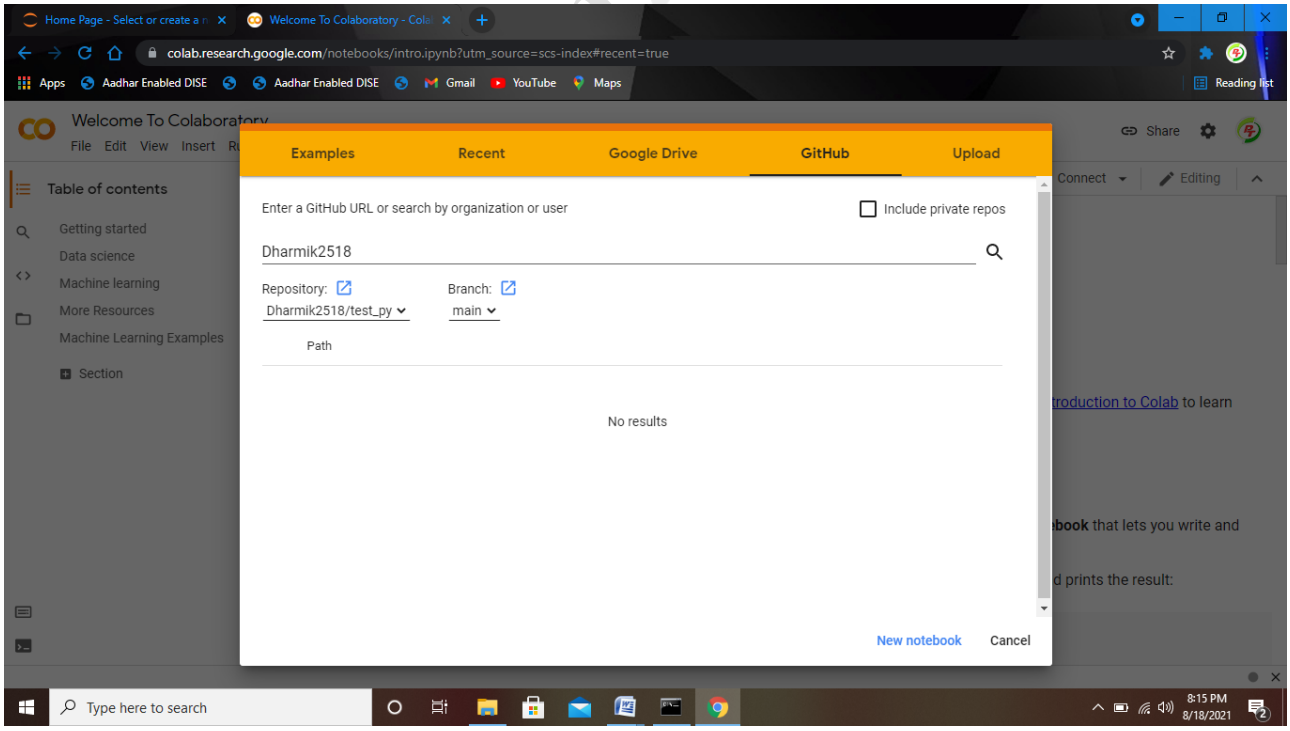
- **Using Google Drive for existing notebooks.**

Google Drive is the default location for many operations in colab, and you can always choose it as a destination. When working with Drive, you see a list of files similar to those shown in Figure. To open a particular file, you click its link in the dialog box. The file opens in the current tab of the browser.

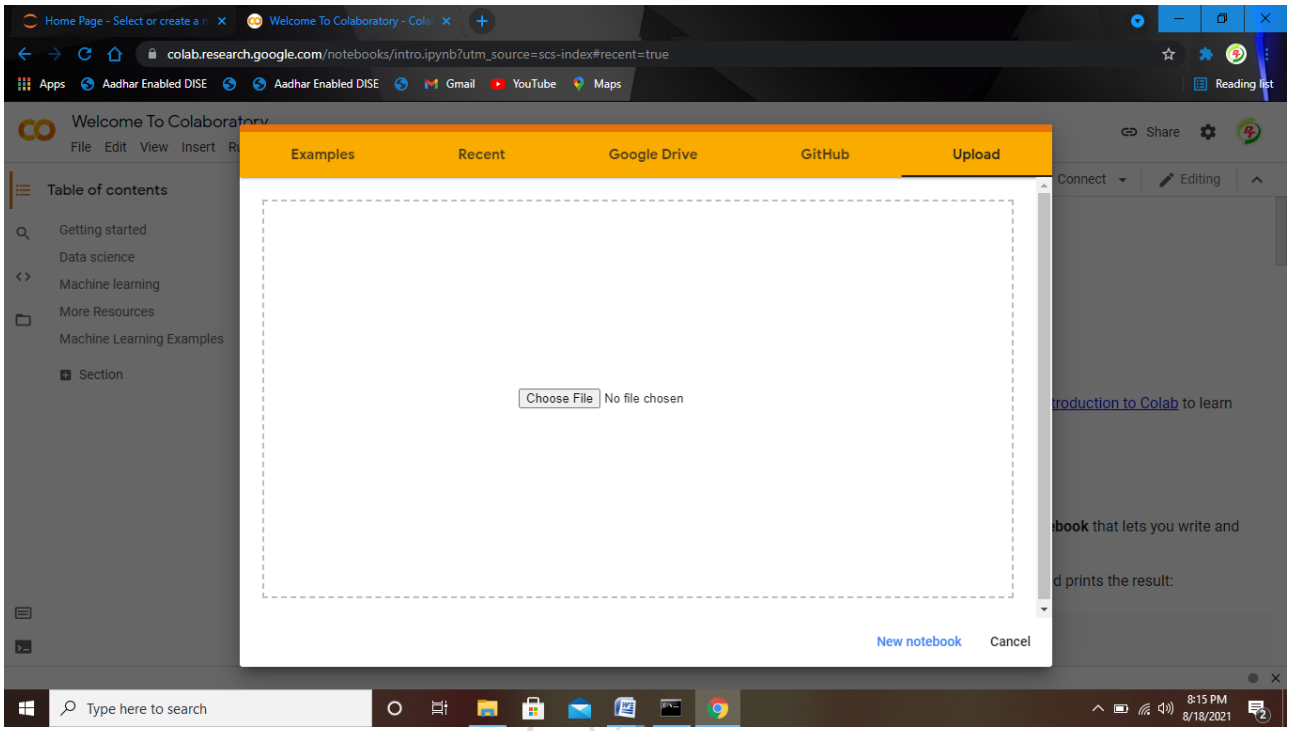


- **Using Github For Existing Notebooks.**

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



- **Using Local Storage For Existing Notebooks.**
  If you want to use the downloadable source for this book, or any local source for that matter, you select the Upload tab of the dialog box. In the center is a single button, Choose File. Clicking this button opens the File Open dialog box for your browser. You locate the file you want to upload, just as you normally would for opening any file.



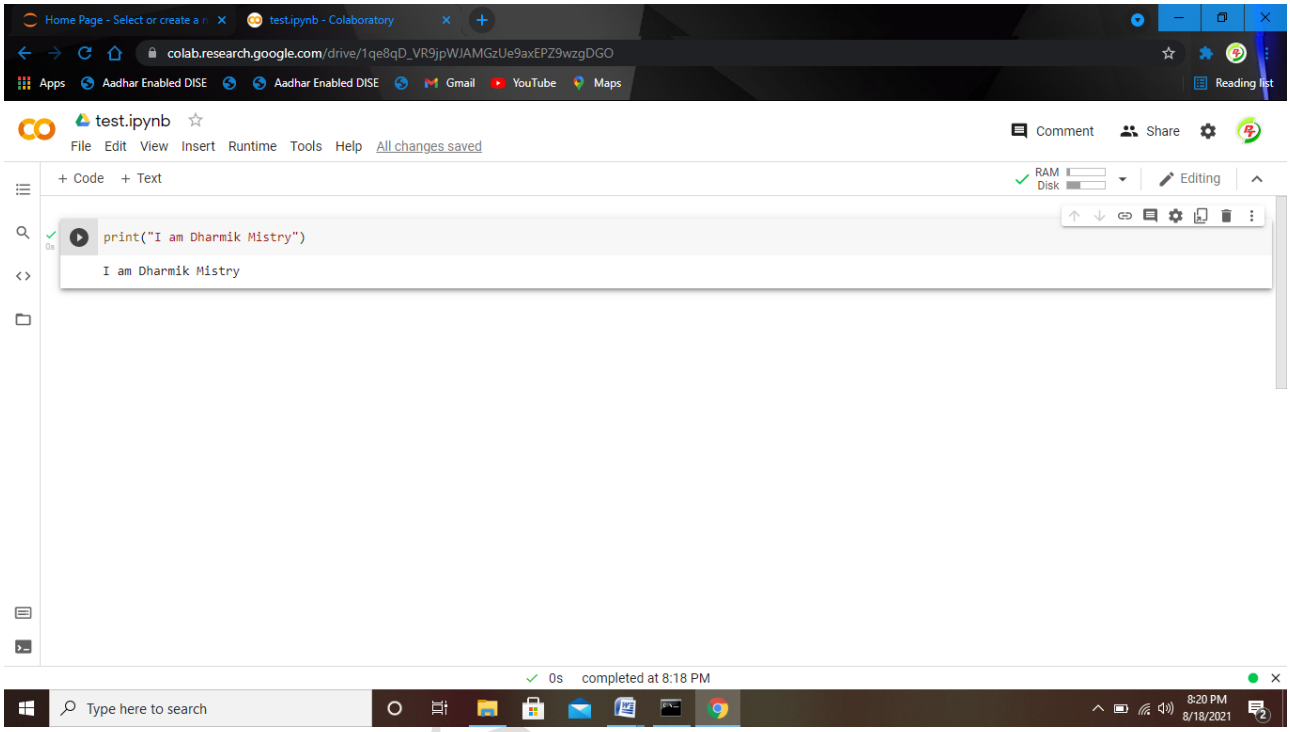
- **Using Drive to save notebooks**
  The default location for storing your data is Google Drive. When you choose File ⇒ Save, the content you create goes to the root directory of your Google Drive. If you want to save the content to a different folder, you need to select that folder in Google Drive (https://drive.google.com/).

- **Using GitHub to save notebooks**
  GitHub provides an alternative to Google Drive for saving content. It offers an organized method of sharing code for the purpose of discussion, review, and distribution. You can find GitHub at https://github.com/.

Practical 3.3 :- Performing Common Tasks

- **Creating code cells**

  The first cell that Colab creates for you is a code cell. The code you create in Colab uses all the same features that you find in Notebook. However, off to the side of the cell, you see a menu of extras that you can use with Colab that aren't present in Notebook, as shown in Figure.

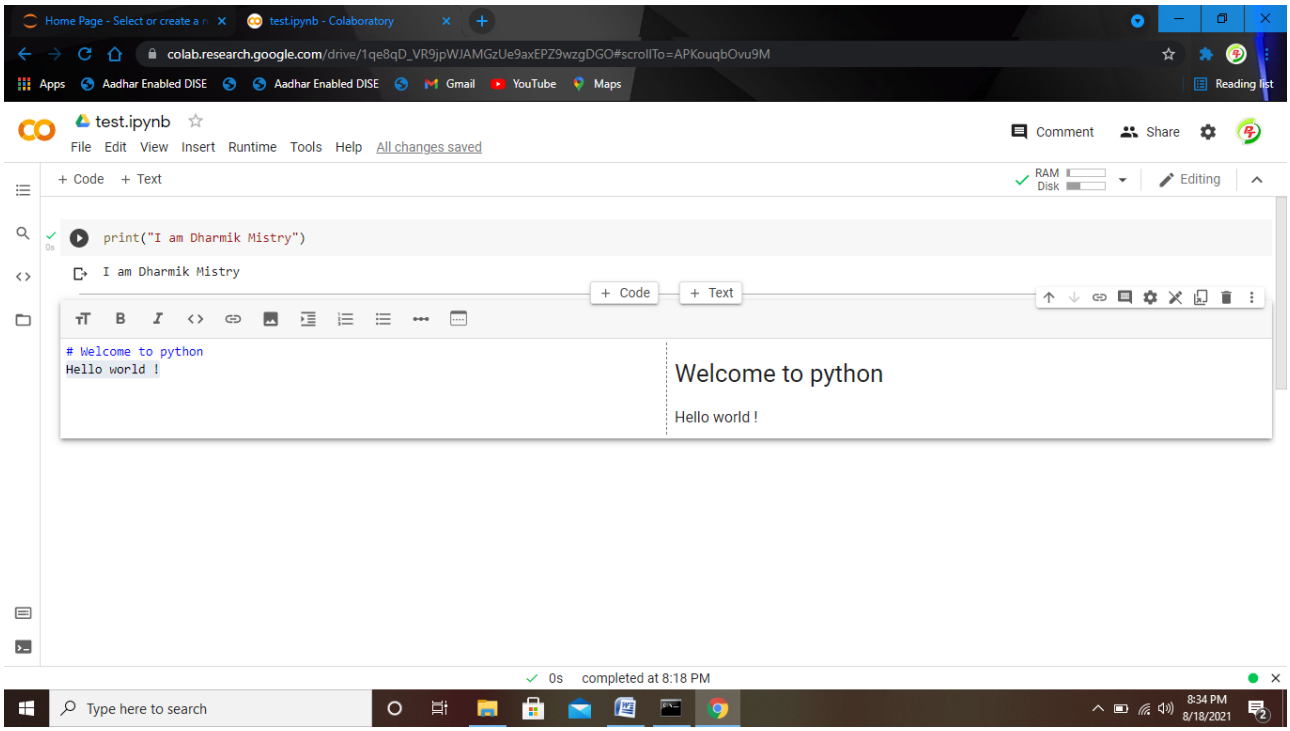


  - ➢ Right click on the cell and select "link to cell" option to link cell.
  - ➢ Right click on the cell and select "delete cell" option to remove cell.
  - ➢ Right click on the cell and select "clear output" option to remove output from cell.
  - ➢ Right click on the cell and select "add a comment" option to add comment to the right of the cell.
  - ➢ Right click on the cell and select "add a form" option to add a form into a cell to right of the code.

- **Creating text cells**

  Text cells work much like Markup cells in Notebook. However, Figure shows that you receive additional help in formatting the text using a graphical interface. The markup is the same, but you have the option of allowing the GUI to help you create the markup. Notice the menu to the right of the text cell. This menu contains many of the same options that a code cell does. For example, you can create a list of links to help people access specific parts of your notebook through an index. Unlike Notebook, you can't execute text cells to resolve the markup they contain.

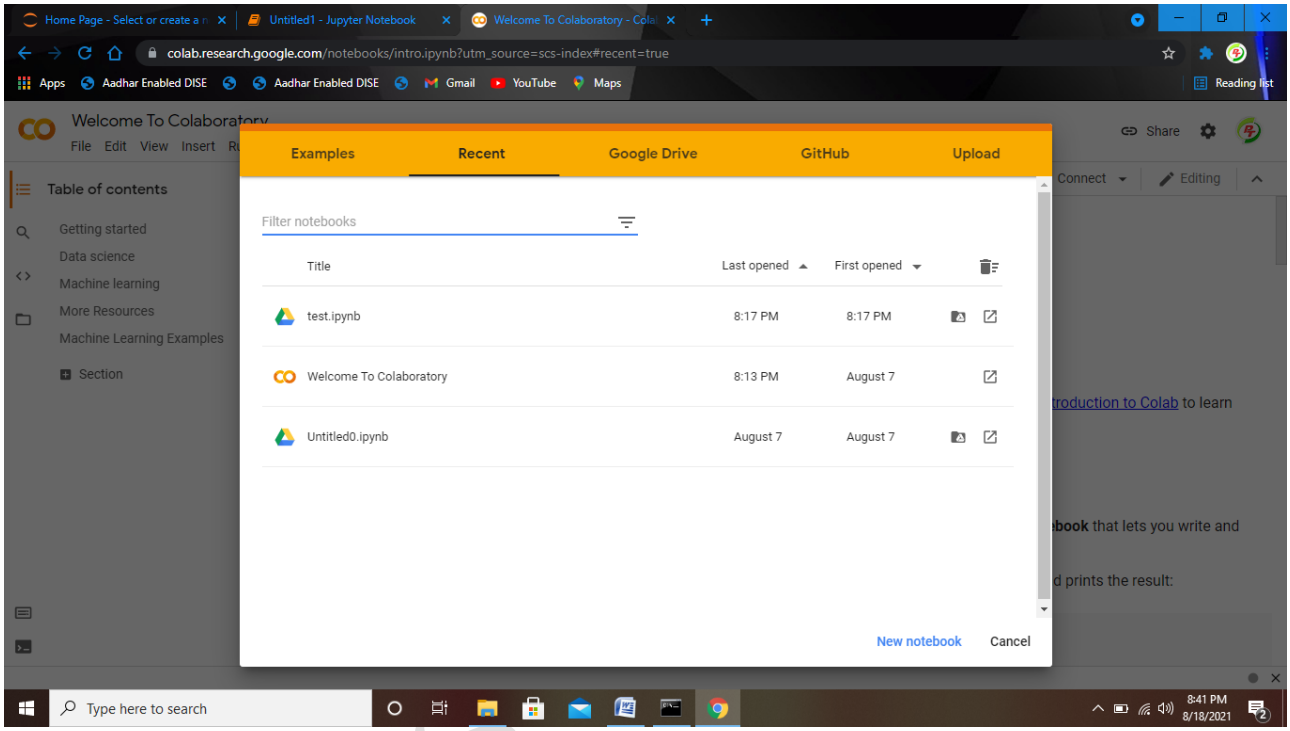
- **Creating special cells Executing the Code**
  The special cells that Colab provides are variations of the text cell. These special cells, which you access using the Insert menu option, make creating the required cells faster. The following sections describe each of these special cell types.
  - **Working with headings:**
    When you choose **Insert ⇒ Section Header Cell**, you see a new cell created below the currently selected cell that has the appropriate header level 1 entry in it. You can increase the heading level by clicking the double T icon. The GUI looks the same as the one in Fig, so you have all the standard formatting features for your text.
  - **Working with table of contents:**
    An interesting addition to Colab is the automatic generation of a table of contents for your notebook. To use this feature, choose **Insert ⇒ Table** of Contents Cell. Fig shows the output for this particular example.



- **Editing cells**
  Both Colab and Notebook have Edit menus that contain the options you except, such as the ability to cut, copy, and paste cells. The two products have some interesting differences. For example, Notebook allows you to split and merge cells. Colab contains an option to show or hide the code as a toggle. These differences give each product a slightly different flavor but don't really change your ability to use it to create and modify Python code.

- **Moving cells**
  The same technique you use for moving cells in Notebook also works with Colab. The only difference is that Colab relies exclusively on toolbar buttons, while Notebook also has cell movement options on the Edit menu.

- **Viewing Your Notebook**
  Save the notebook at Google Drive by following particular path i.e., **File → save / save a copy** in Drive and then view the notebook by following particular path i.e., **File → Open** Notebook and then select the notebook which you want to open.



Practical 3.4 :- Defining the code repository

- **Defining a new folder**

  You use folders to hold your code files for a particular project. The project for this book is Dhruv_19_38.

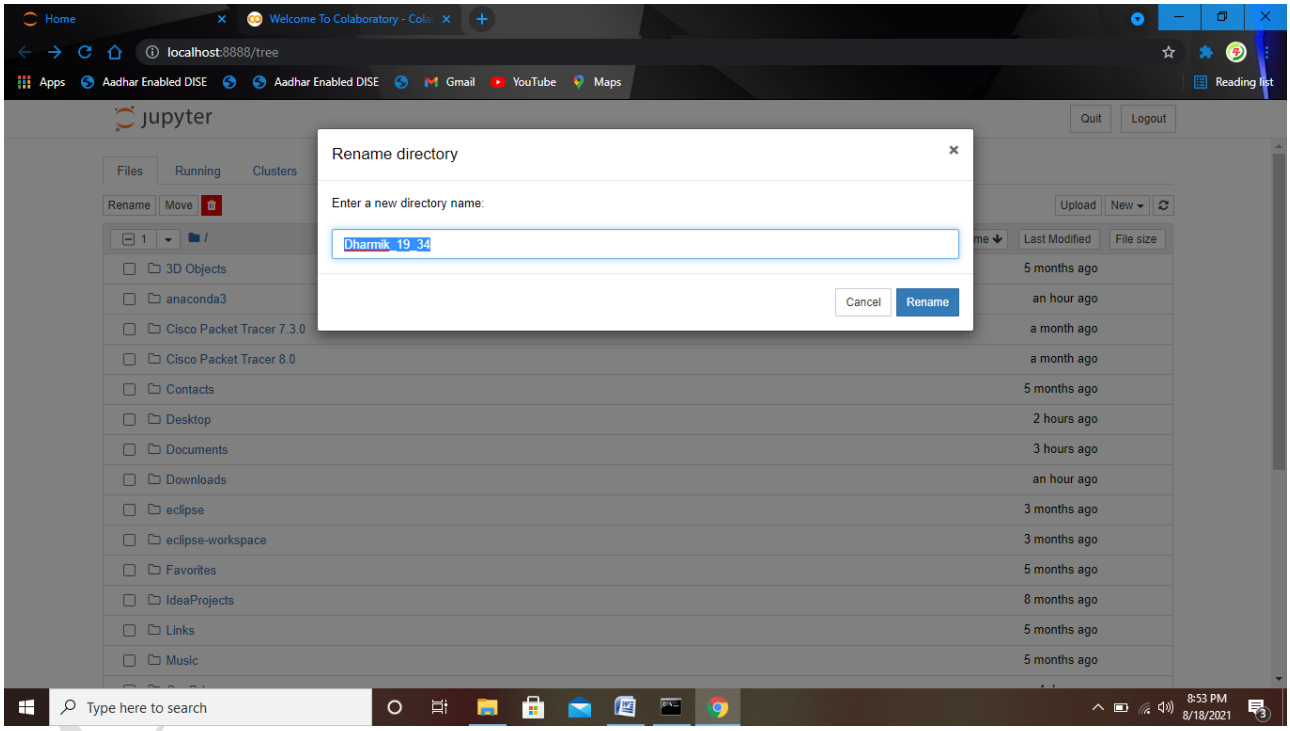
  The following steps help you create a new folder for this book.

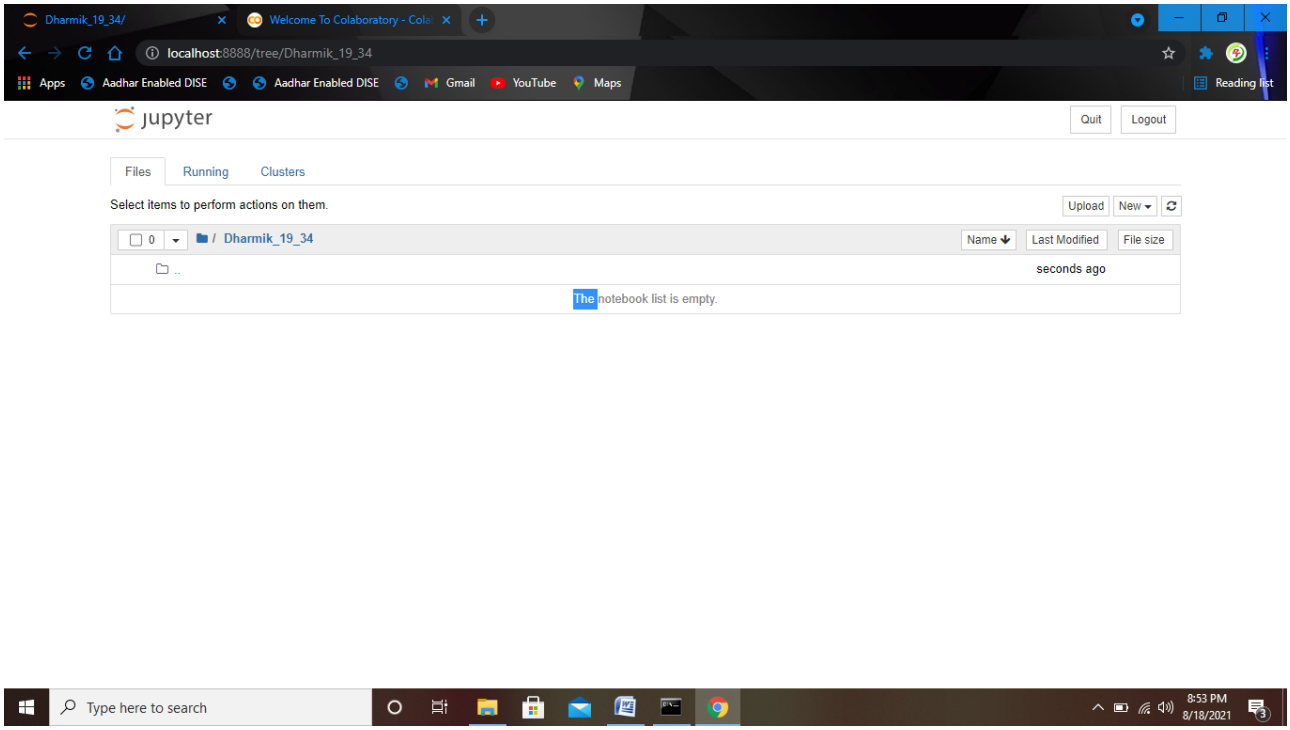
1. **Choose New ⇒ Folder.**

   Notebook creates a new folder for you. The name of the folder can vary, but for Windows users, it's simply listed as Untitled Folder. You may have to scroll down the list of available folders to find the folder in question.
2. **Place a check in the box next to Untitled Folder.**
3. **Click Rename at the top of the page.**

   You see the Rename Directory dialog box.
4. **Type Folder name and press Enter.**

   Notebook renames the folder for you.

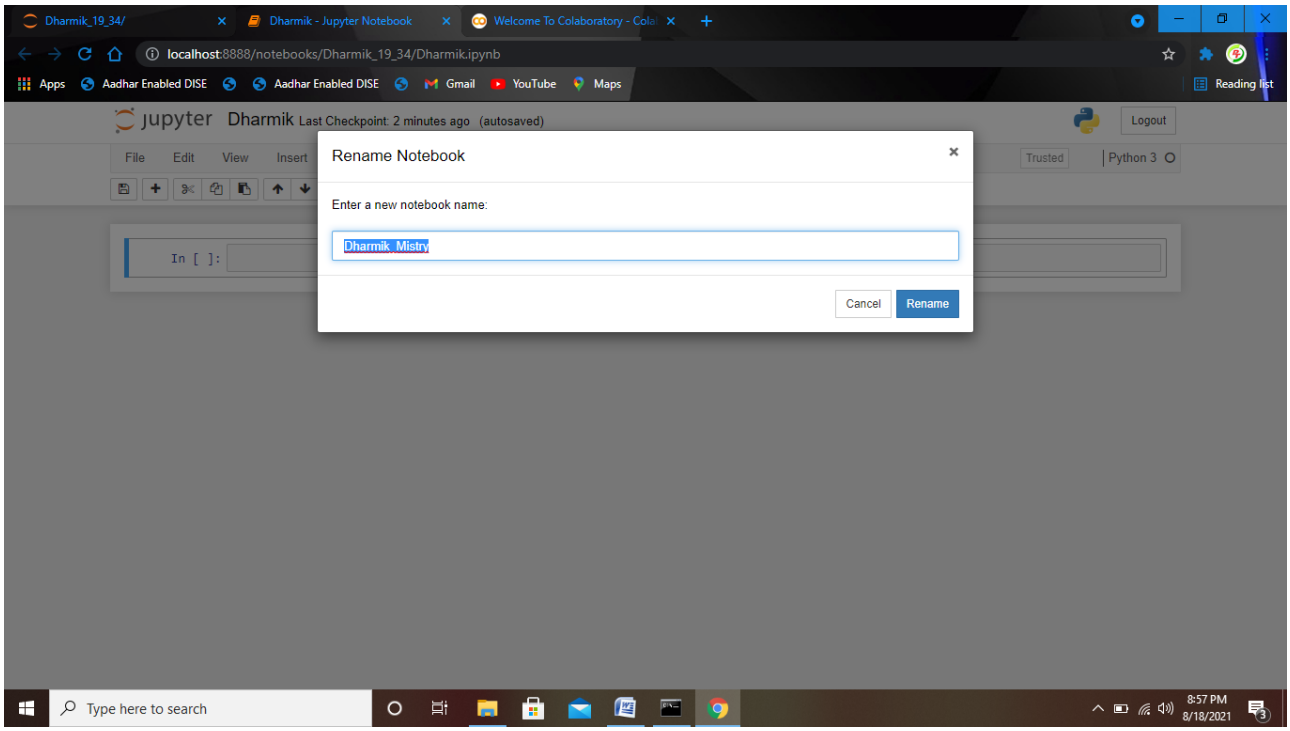




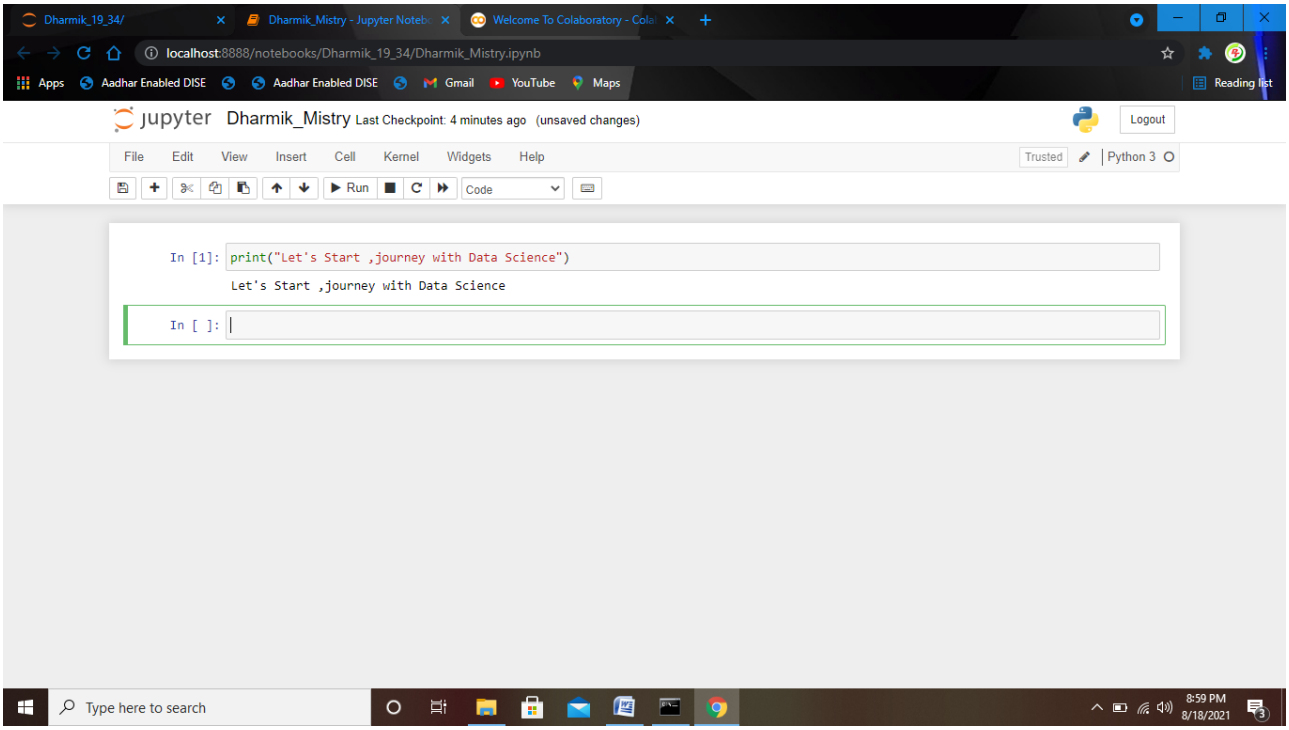
- **Creating a new notebook**

  Every new notebook is like a file folder. You can place individual examples within the file folder, just as you would sheets of paper into a physical file folder. Each example appears in a cell. You can put other sorts of things in the file folder, too, but you see how these things work as the book progresses. Use these steps to create a new notebook.

1. **Click the Folder name entry on the Home page.**
   You see the contents of the project folder for this book, which will be blank if you're performing this exercise from scratch.
2. **Choose New ⇒ Python 3.**
   You see a new tab open in the browser with the new notebook, as shown in Figure. Notice that the notebook contains a cell and that Notebook has highlighted the cell so that you can begin typing code in it. The title of the notebook is Untitled right now. That's not particularly helpful title,so you need to change it.
3. **Click Untitled on the page.**
   Notebook asks whether you want to use a new name, as shown in Figure
4. **Type Filename and press Enter.**



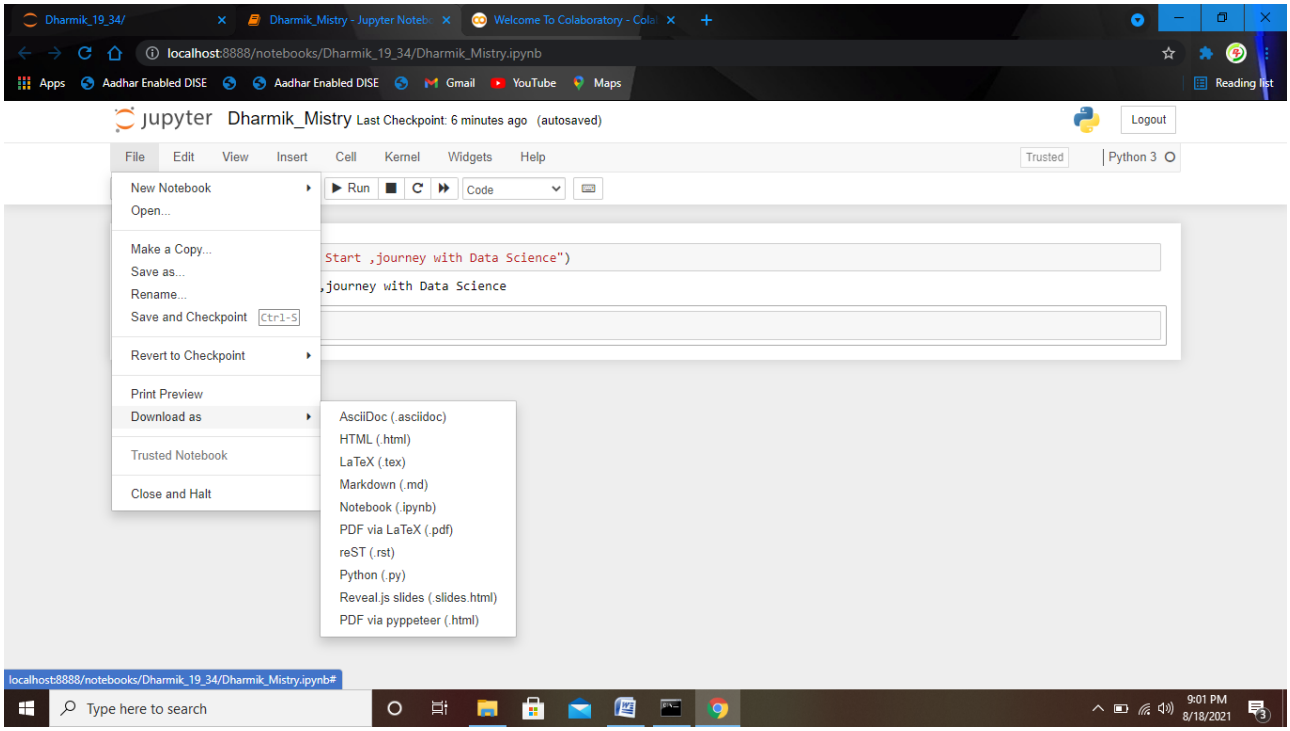
- **Adding notebook content**



Exporting a notebook

It isn't much fun to create notebooks and keep them all to yourself. At some point, you want to share them with other people. To perform this task, you must export your notebook from the repository to a file. You can then send the file to someone else who will import it into his or her repository. The previous section shows how to create a notebook named Sample. You can open this notebook by clicking its entry in the repository list. The file reopens so that youcan see your code again.

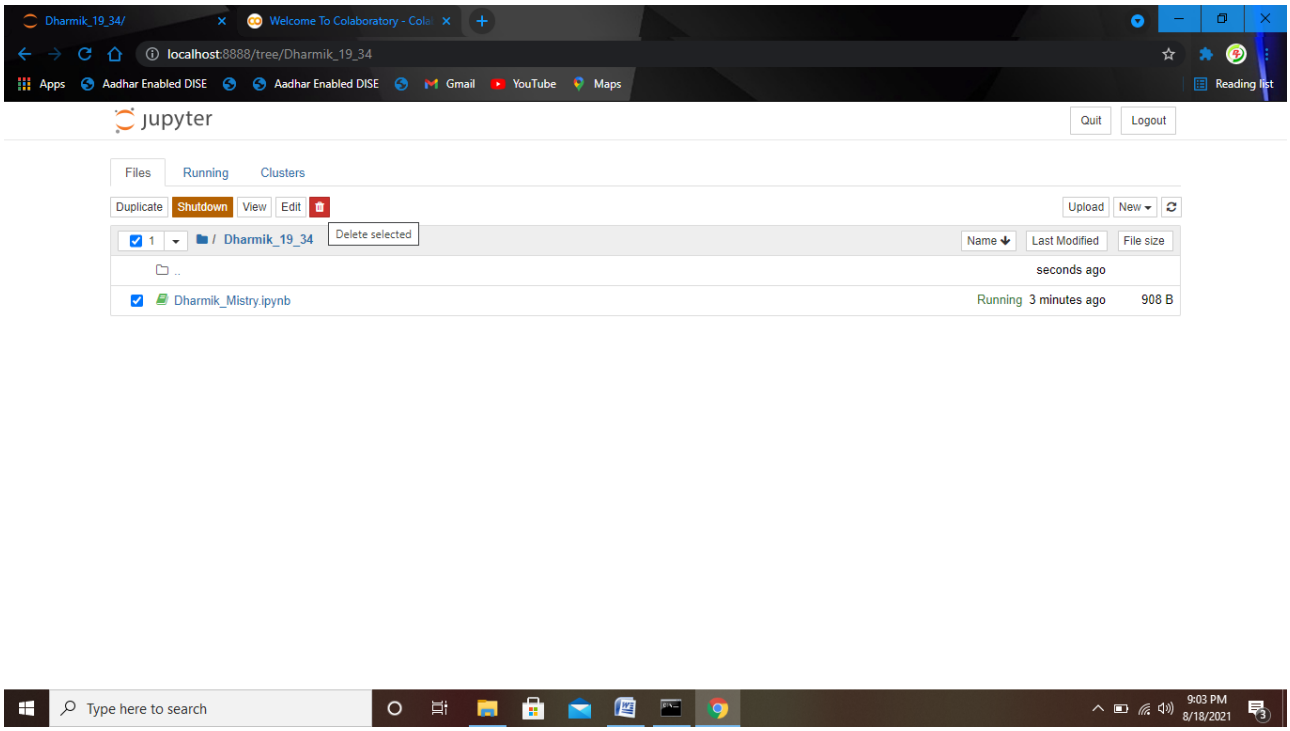
To export this code, choose **File ⇒ Download As ⇒ Notebook (.ipynb).** What you see next depends on your browser, but you generally see some sort of dialog box for saving the notebook as a file. Use the same method for saving the Notebook file as you use for any other file you save using your browser.



Removing a notebook

Sometimes notebooks get outdated or you simply don't need to work with them any longer. Rather than allow your repository to get clogged with files you don't need, you can remove these unwanted notebooks from the list. Use these steps to remove the file:

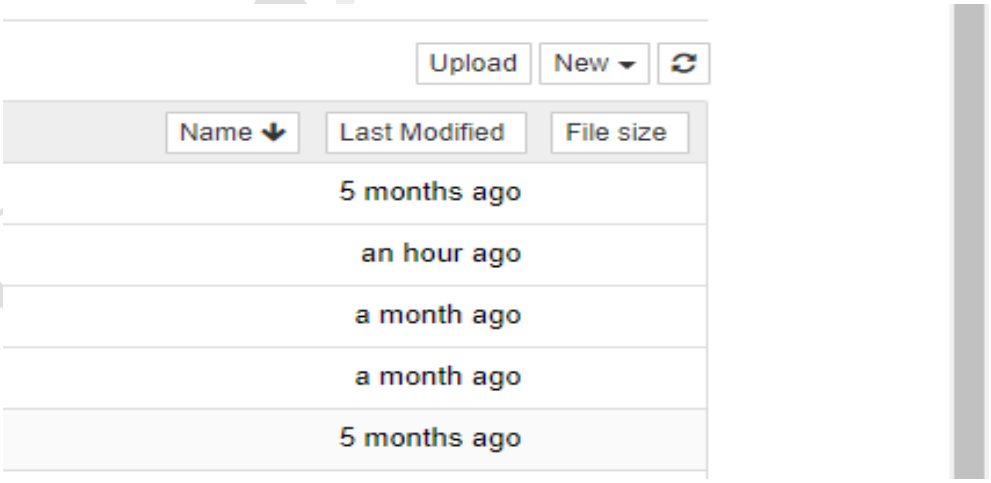
1. **Select the check box next of the notebook you want to remove.**
2. **Click the Delete (trashcan) icon.**
   You see a Delete notebook warning message like the one shown in Figure.
3. **Click Delete.**
   Notebook removes the notebook file from the list.

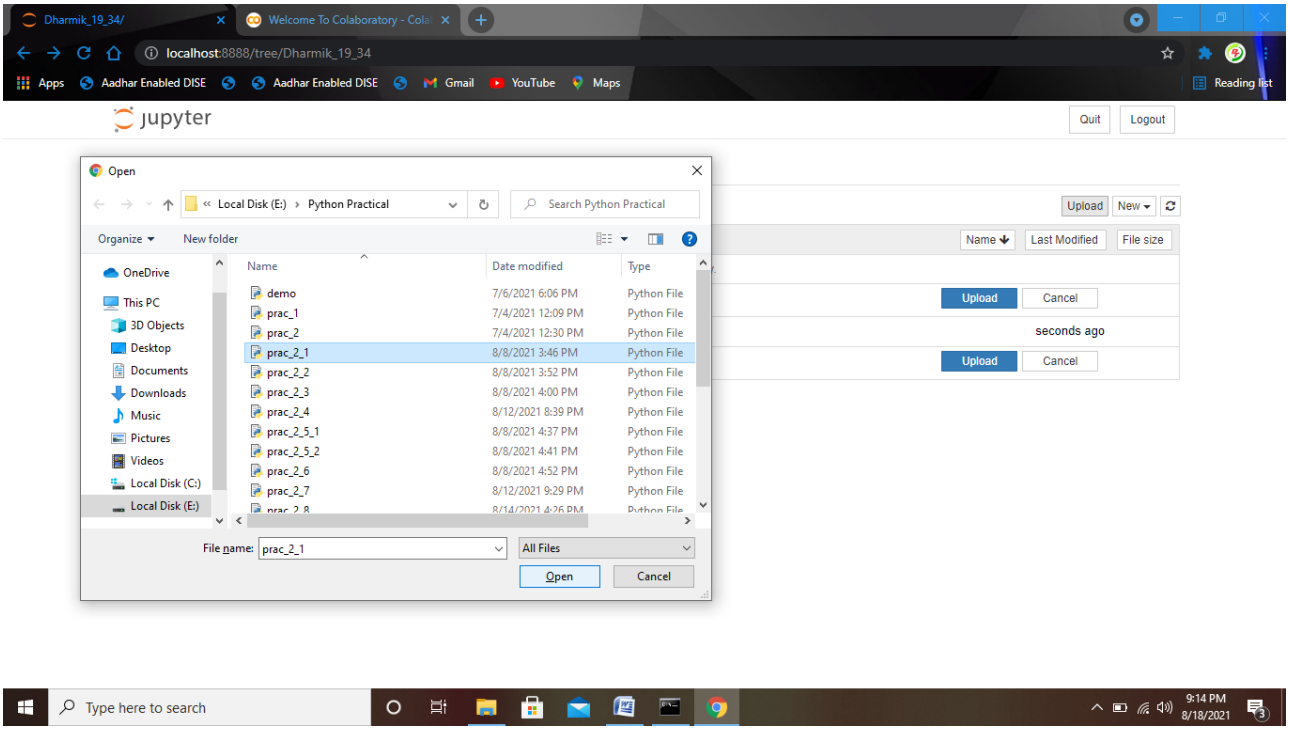


Importing a notebook

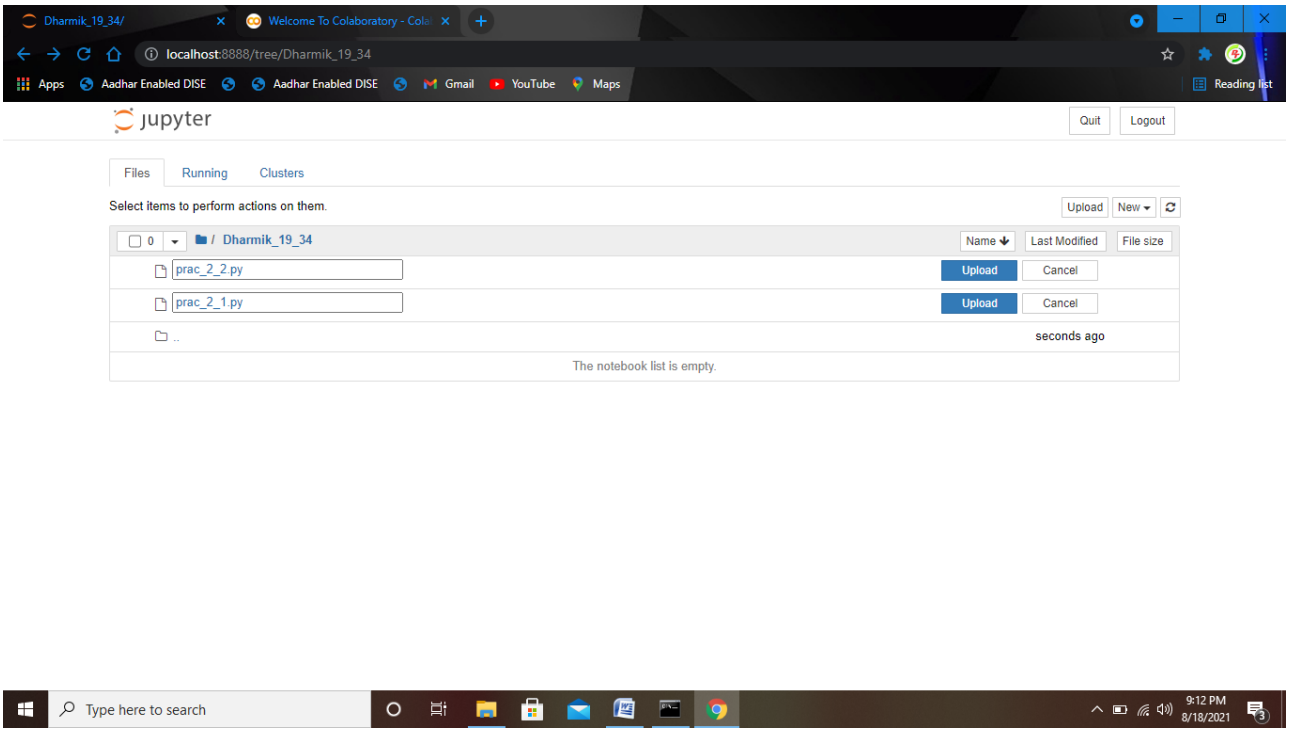
To use the source code from this book, you must import the downloaded files into your repository. The source code comes in an archive file that you extract to a location on your hard drive. The archive contains a list of .ipynb (IPython Notebook) files containing the source code. The following steps tell how to import these files into your repository:

1. **First, navigate to the Jupyter Notebook interface home page.**
2. **Click the "Upload" button to open the file chooser window.**
3. **Choose the file you wish to upload. You may select multiple files if you wish.**





4. Click "Upload" for each file that you wish to upload.



Practical 3.4 : Understanding the following datasets with its code.

This book uses a number of datasets, all of which appear in the Scikit-learn library. These dataset demonstrate various ways in which you can interact with data, and you use them in the examples to perform a variety of tasks. The following list provides a quick overview of the function used to import each of the datasets into your Python code:

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

  Load and return the boston house-prices dataset (regression).

```
In [1]: from sklearn.datasets import load_boston
        Boston = load_boston()
        print(Boston.data.shape)

        (506, 13)
```

- **load_iris(): Classification with the Iris dataset**

  Load and return the iris dataset (classification).The iris dataset is a classic and very easy multi-class classification dataset.

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

        (150, 4)
```

- **load_diabetes(): Regression with the diabetes dataset**

  Load and return the diabetes dataset (regression).

```
In [3]: from sklearn.datasets import load_diabetes
        diabetes = load_diabetes()
        print(diabetes.data.shape)

        (442, 10)
```

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

  Load and return the digits dataset (classification).
  Each datapoint is a 8x8 image of a digit.

```
In [4]: from sklearn.datasets import load_digits
        digits = load_digits()
        print(digits.data.shape)

        (1797, 64)
```

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

  Load the filenames and data from the 20 newsgroups dataset (classification).
  Download it if necessary.

```
In [1]: from sklearn.datasets import fetch_20newsgroups
        twenty_train = fetch_20newsgroups(subset = 'train')
        print(twenty_train.data[0:1])

        ["From: lerxst@wam.umd.edu (where's my thing)\nSubject: WHAT car is this!?\nNntp-Posting-Host: rac3.wam.umd.edu\nOrganization:
        University of Maryland, College Park\nLines: 15\n\n I was wondering if anyone out there could enlighten me on this car I saw\nt
        he other day. It was a 2-door sports car, looked to be from the late 60s/\nearly 70s. It was called a Bricklin. The doors were
        really small. In addition,\nthe front bumper was separate from the rest of the body. This is \nall I know. If anyone can tellme
        a model name, engine specs, years\nof production, where this car is made, history, or whatever info you\nhave on this funky loo
        king car, please e-mail.\n\nThanks,\n- IL\n   ---- brought to you by your neighborhood Lerxst ----\n\n\n\n\n"]
```

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

  Load the Olivetti faces data-set from AT&T (classification).
  Download it if necessary.

```
In [3]: from sklearn.datasets import fetch_olivetti_faces
        faces = fetch_olivetti_faces()
        print(faces.data.shape)

        (400, 4096)
```

PRACTICAL-04

Aim: Uploading, Streaming, and Sampling Data Using Pandas

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

Program :

```
with open("/content/sample_data/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
```

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

Program :

```
with open("/content/sample_data/colors.txt", 'r') as open_file:
 for observation in open_file:
  print('Reading Data: ' + observation)
```

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
```

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

Program :

```
d = 2
with open("/content/sample_data/colors.txt", 'r') as open_file:
 for r, observation in enumerate(open_file):
  if r % d!= 0:
   print('Reading Line: ' + str(r) +
   ' Content: ' + observation)
```

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
```

Practical-4.4: Select random samples from the file Colors.txt

Program :

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

OUTPUT :

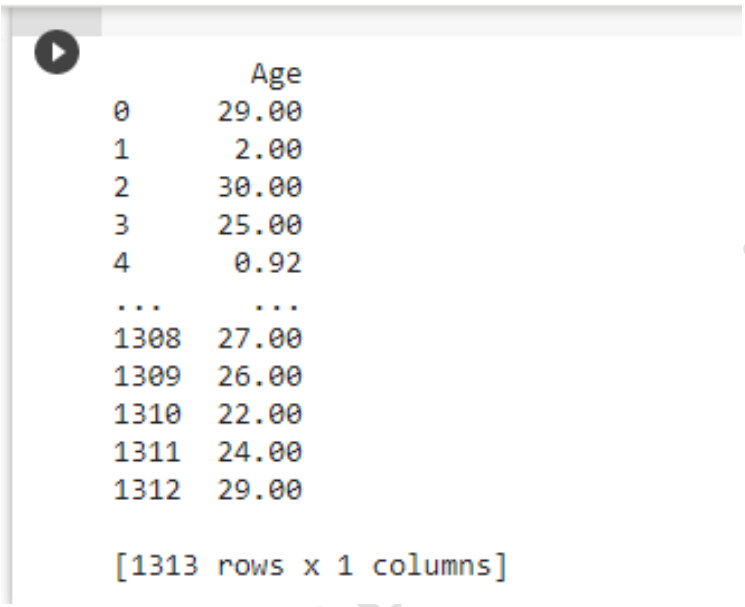
```
Reading Line: 3 Content: Yellow 3

Reading Line: 7 Content: Black  7
```

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

Program :

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

OUTPUT :



```
          Age
0       29.00
1        2.00
2       30.00
3       25.00
4        0.92
...       ...
1308    27.00
1309    26.00
1310    22.00
1311    24.00
1312    29.00

[1313 rows x 1 columns]
```

Practical-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("/content/sample_data/Values.xlsx")

data=excel.parse(nrows=10)

print(data)
```

OUTPUT :

	Ship Mode	Profit	Unit Price	Shipping Cost	Customer Name
0	Regular Air	-213.2500	38.94	35.00	Muhammed MacIntyre
1	Delivery Truck	457.8100	208.16	68.02	Barry French
2	Regular Air	46.7075	8.69	2.99	Barry French
3	Regular Air	1198.9710	195.99	3.99	Clay Rozendal
4	Regular Air	-4.7150	5.28	2.99	Claudia Miner
5	Regular Air	782.9100	39.89	3.04	Neola Schneider
6	Regular Air	93.8000	15.74	1.39	Allen Rosenblatt
7	Delivery Truck	440.7200	100.98	26.22	Sylvia Foulston
8	Regular Air	-481.0410	100.98	69.00	Sylvia Foulston
9	Regular Air	-11.6820	65.99	5.26	Jim Radford

<u>Practical-4.7</u> : 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

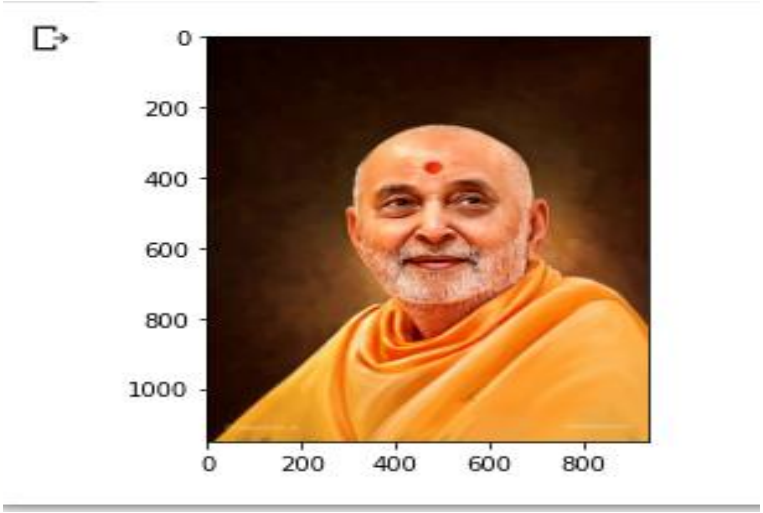
d) Flatening the image

I. <u>Reading image from local storage</u> :

Program :

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

ex_file= ("/content/sample_data/guru.jpg")
image=imread(ex_file,as_gray=False)
plt.imshow(image,cmap=cm.gray)
plt.show()
```

<u>OUTPUT</u> :



(a). Displaying the image information

Program :

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

OUTPUT :

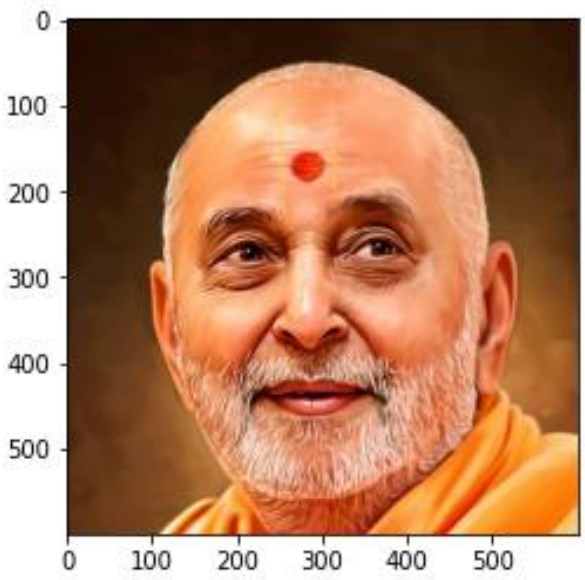
```
data type:<class 'numpy.ndarray'>,
shape: (1149, 940, 3)
size: 3240180
```

(b). Cropping the image

Program :

```
image2=image[200:800,200:800]
plt.imshow(image2,cmap=cm.gray)
plt.show()
```

OUTPUT :



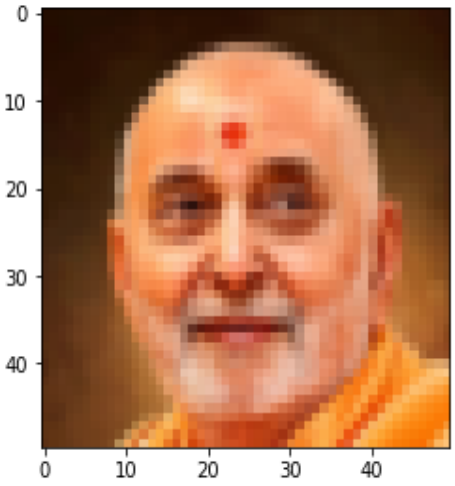
(c).Resizing the image

Program :

```
image3 = resize(image2,(50,50))
plt.imshow(image3,cmap=cm.gray)
print("data type:%s,shape:%s"%(type(image3),image3.shape))
plt.show()
```

OUTPUT :

```
data type:<class 'numpy.ndarray'>,shape:(50, 50, 3)
```



(d).Flatening the image

Program :

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

OUTPUT :

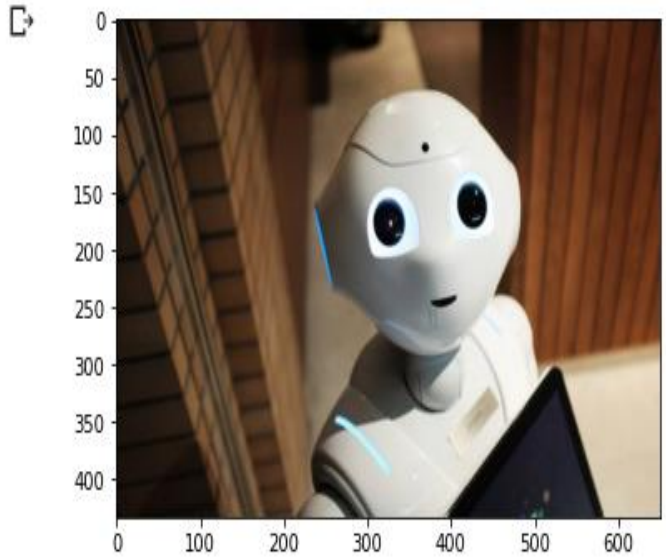
```
data type:<class 'numpy.ndarray'>,shape:(7500,)
```

II. <u>Reading image from specific URL</u> :

Program :

```
from skimage.io import imread

from skimage.transform import resize

from matplotlib import pyplot as plt

import matplotlib.cm as cm


example_file = ("https://www.wgu.edu/content/dam/web-sites/blog-
newsroom/blog/images/national/2020/march/artificial-intelligence.jpg")

im = imread(example_file,as_gray=False)

plt.imshow(im,cmap=cm.gray)

plt.show()
```

<u>OUTPUT</u> :



<u>(a).Display the image information</u>

Program :

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

print("shape:",im.shape)

print("size:",im.size)
```

OUTPUT :

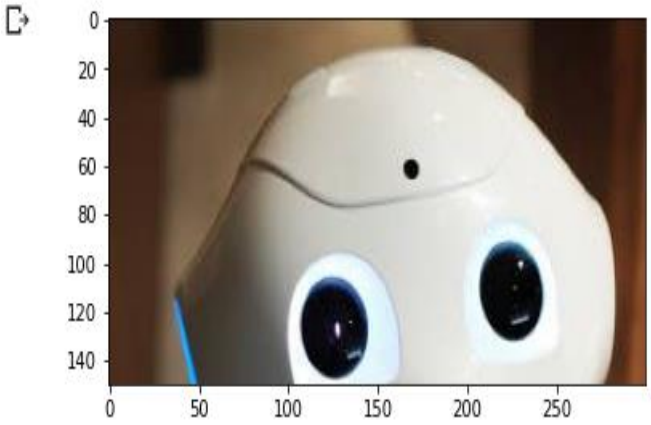
```
data type:<class 'numpy.ndarray'>,
shape: (433, 650, 3)
size: 844350
```

(b).Cropping the image

Program :

```
image2=im[50:200,200:500]
plt.imshow(image2,cmap=cm.gray)
plt.show()
```

OUTPUT :



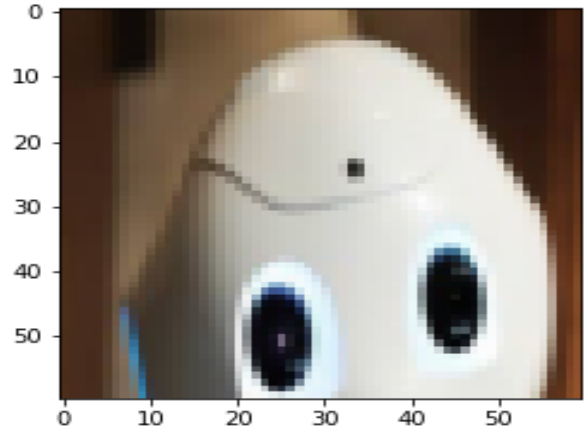
(c).Resizing the image

Program :

```
image3 = resize(image2,(60,60))
plt.imshow(image3,cmap=cm.gray)
print("data type:%s,shape:%s"%(type(image3),image3.shape))
plt.show()
```

OUTPUT :

```
data type:<class 'numpy.ndarray'>,shape:(60, 60, 3)
```



(d).Flatening the image

Program :

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

OUTPUT :

```
data type:<class 'numpy.ndarray'>,shape:(10800,)
```

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

Program :

```
from lxml import objectify
import pandas as pd
xml=objectify.parse(open('/content/sample_data/XMLData.xml'))
root=xml.getroot()
df=pd.DataFrame(columns=('Number','String','Boolean'))
for i in range(0,6):
  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   False
3       4  Fourth   False
4       5   Fifth    True
5       6   sixth    True
```

PRACTICAL-06

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

Program :

```
import pandas as pd

df=pd.read_csv("/content/sample_data/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

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

Using mean() method :

Program :

```
import pandas as pd
from sklearn.impute import SimpleImputer
df=pd.read_csv("/content/sample_data/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

Using median() method :

Program :

```
import pandas as pd
from sklearn.impute import SimpleImputer
df=pd.read_csv("/content/sample_data/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

Using most_frequent() method :

Program :

```
import pandas as pd
from sklearn.impute import SimpleImputer
df=pd.read_csv("/content/sample_data/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

Practical-6. 3 : Delete all the missing data entry in cars.csv

Program :

```
import pandas as pd

df=pd.read_csv("/content/sample_data/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-08

Aim: Working with Data Visualization.

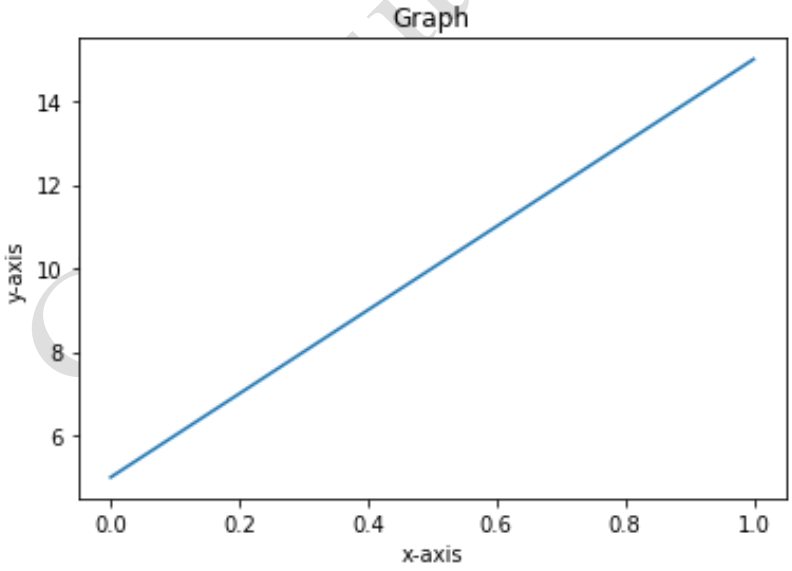
Practical 8.1:

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 :



Practical 8.2:

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

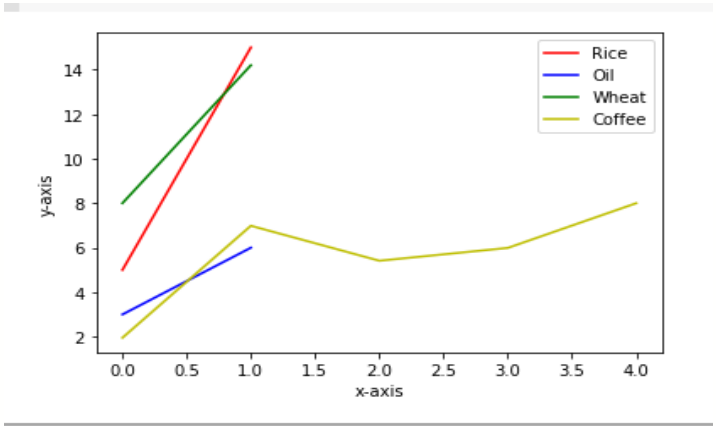
Rice	[5, 15]
Oil	[3, 6]
Wheat	[8.0010, 14.2]
Coffee	[1.95412, 6.98547, 5.41411, 5.99, 7.9999]

Also Mark the Line graph with Marker

Program:

```
import matplotlib.pyplot as plt

%matplotlib inline

plt.plot([5,15],'r',label='Rice')

plt.plot([3,6],'b',label='Oil')

plt.plot([8.0010,14.2],'g',label='Wheat')

plt.plot([1.95412,6.98547,5.41411,5.99,7.9999],'y',label='Coffee')

plt.legend()

plt.xlabel('x-axis')

plt.ylabel('y-axis')

plt.show()
```

OUTPUT :



Practical 8.3:

Plot scatter with marker size of 20 and 10 for x1 and x2 for the following data:

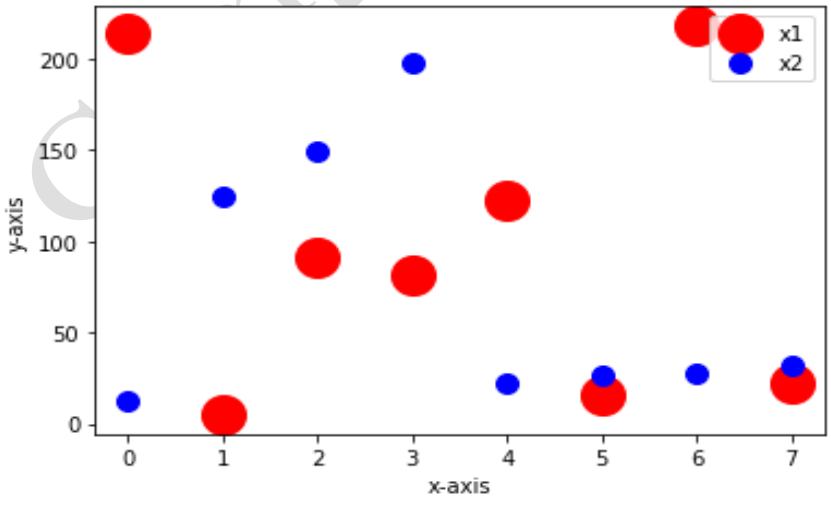
x1 = [214, 5, 91, 81, 122, 16, 218, 22]

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 :



Practical 8.4:

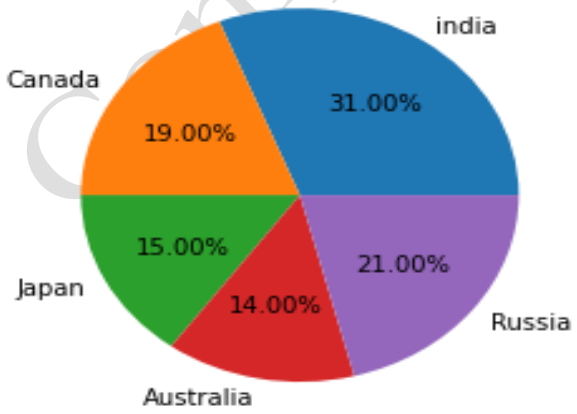
Plot pie graph for the following values and also turn on the axis of graph.

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

Program:

```
import matplotlib.pyplot as plt
%matplotlib inline
label=["india","Canada","Japan","Australia","Russia"]
size=[31,19,15,14,21]
plt.pie(size,labels=label,autopct='%.2f%%')
plt.show()
```

OUTPUT :



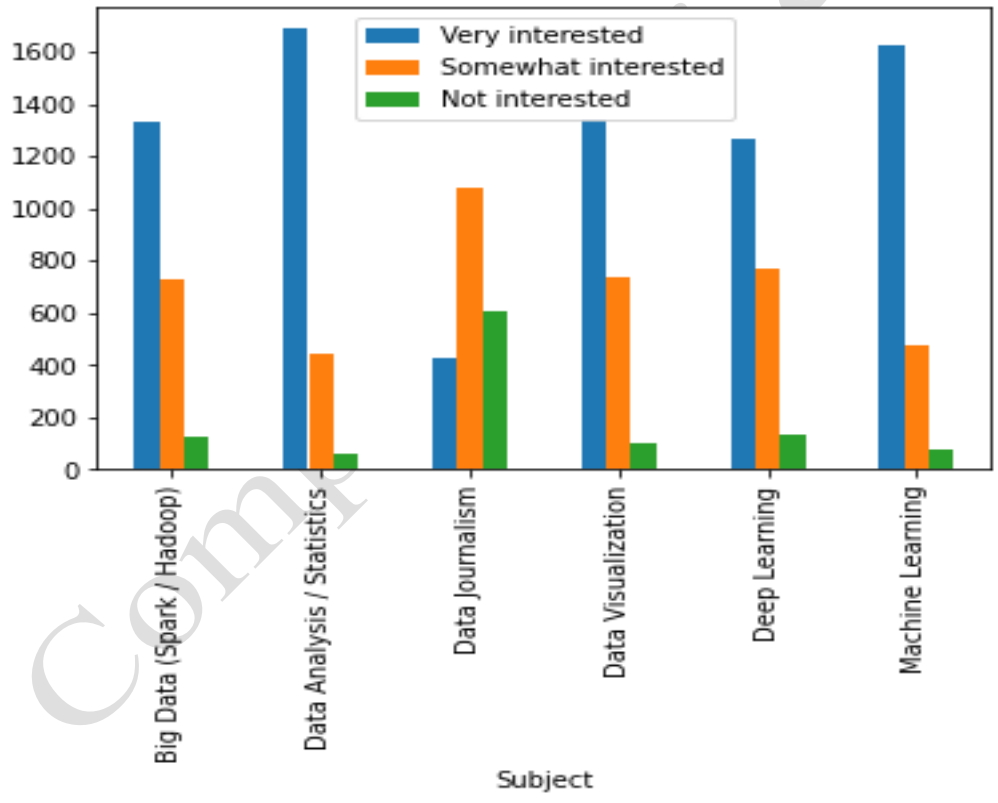
Practical 8.5:

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 matplotlib.pyplot as plt

import pandas as pd

df=pd.read_csv('/content/sample_data/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 :



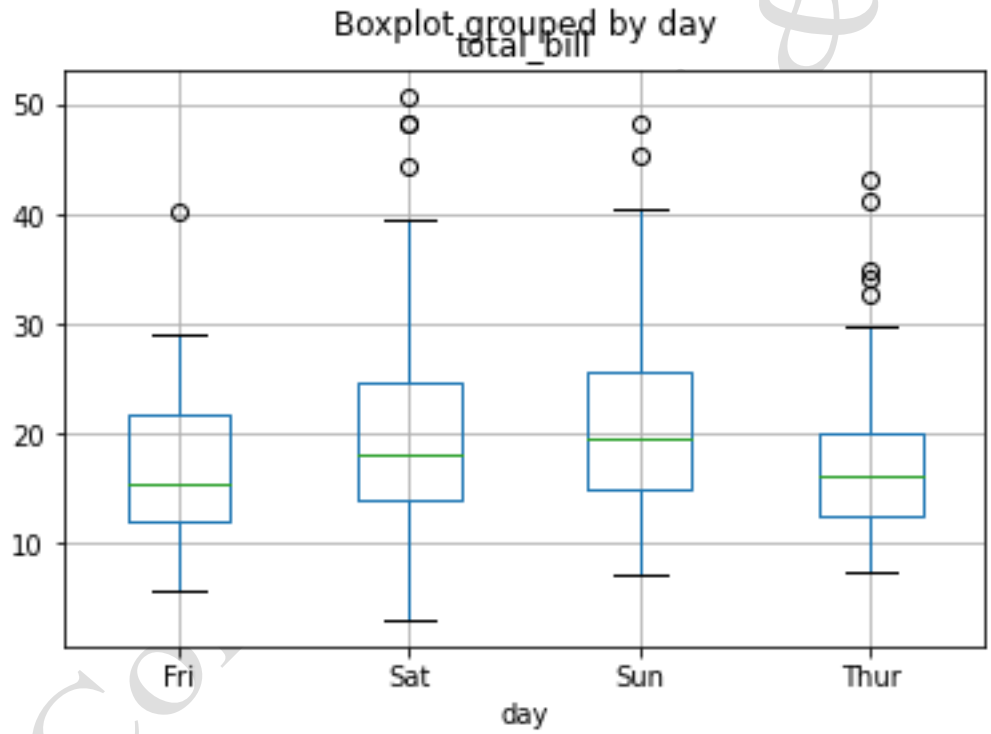
Practical 8.6:

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 pandas as pd

import matplotlib.pyplot as plt

df=pd.read_csv("/content/sample_data/tips.csv")

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

OUTPUT :



Practical 8.7:

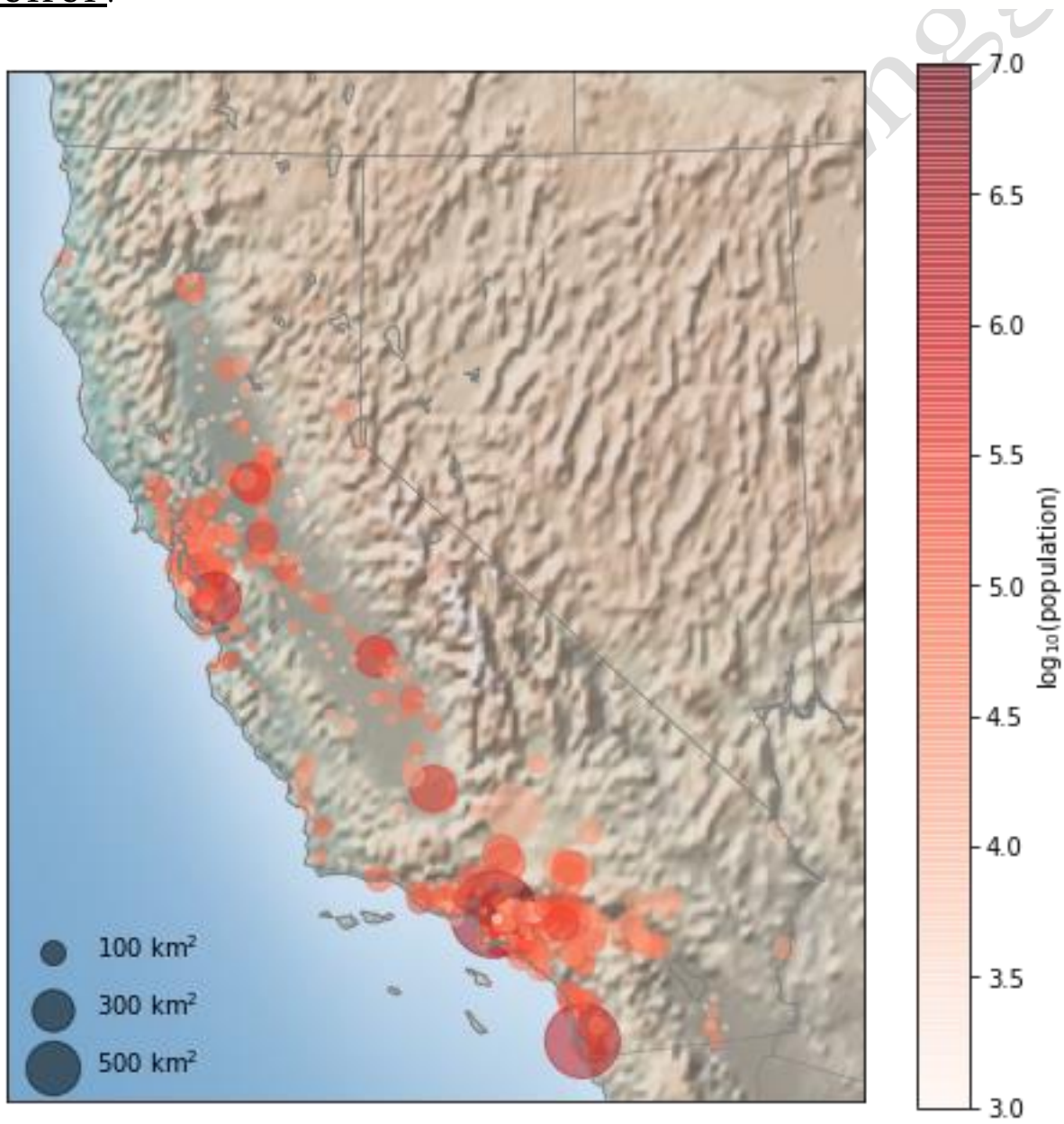
Plot the Geographic Data from california_cities.csv file using basemap.

Program:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
cities = pd.read_csv('/content/sample_data/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 :



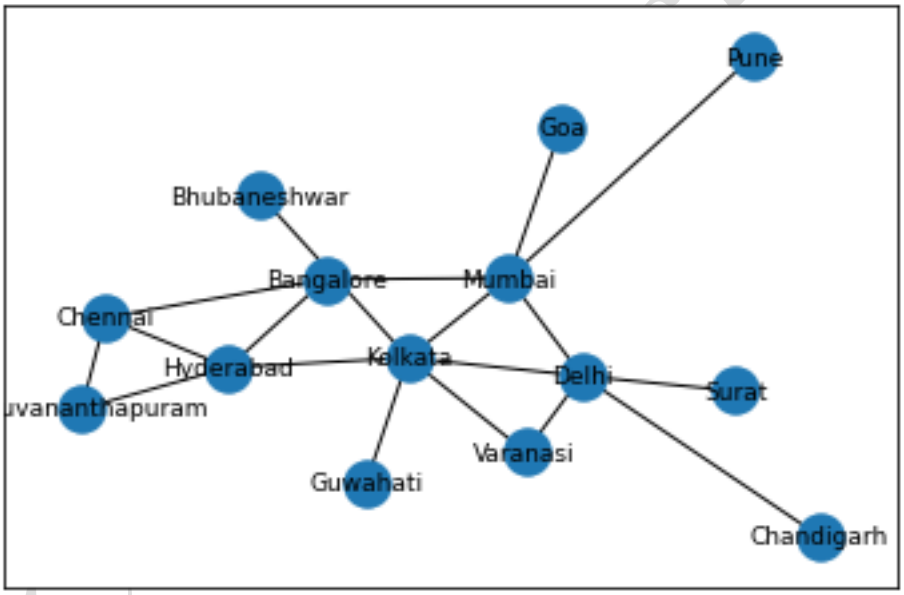
Practical 8.8:

Here is a Dataset of various Indian cities and the distances between them in edge_list.txt. Draw the Graphfor the data.

Program:

```
import pandas as pd

import networkx as nx

df = pd.read_csv('/content/sample_data/edge_list.txt', delim_whitespace = True,

header = None, names =['n1', 'n2', 'weight'])

graph = nx.read_weighted_edgelist('/content/sample_data/edge_list.txt', delimiter =" ")

nx.draw_networkx(graph,with_labels=True, font_size=9)
```

OUTPUT :



PRACTICAL-09

Aim : Working with Data Wrangling

Load the Boston dataset from sklearn library concerns the housing prices in housing city of Boston. Thedataset 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.

Program :

```
#import required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_boston
boston = load_boston()

#shape of the dataset
print("Shape:",boston.data.shape)

#check features in the dataset
print("Features:",boston.feature_names)
```

```
Shape: (506, 13)
Features: ['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

#Converting data from nd-
array to DataFrame amd aading features names and 'price' columns to the dataset

```
data=pd.DataFrame(boston.data)

data.columns=boston.feature_names

data['Price']=boston.target

data.head(10)
```

	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 bodton dataset

```
data.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	P
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	506.000000	506.00
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	12.653063	22.53
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	7.141062	9.19
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	1.730000	5.00
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	6.950000	17.02
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	11.360000	21.20
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	16.955000	25.00
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	37.970000	50.00

#data Preprocessing

data.isnull().sum()

```
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
```

#Spliting data to training and testing dataset

#input data

x=boston.data

#output data

y=boston.target

#spliting data to training and tetsing datasets

from sklearn.model_selection import train_test_split

xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=4)

print("xtrain shape:",xtrain.shape)

print("xtest shape:",xtest.shape)

print("ytrain shape:",ytrain.shape)

print("ytest shape:",ytest.shape)

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

#Training the model

#fitting multi linear regresion model to training model

```
from sklearn.linear_model import LinearRegression
regressor=LinearRegression()
regressor.fit(xtrain,ytrain)
```

#predict the test set results

```
y_pred=regressor.predict(xtest)
```

#check model accuracy

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

#Error Calculation by RMSE(Root Mean Square Error)

```
from math import sqrt
print('RMSE : ',sqrt(mse))
```

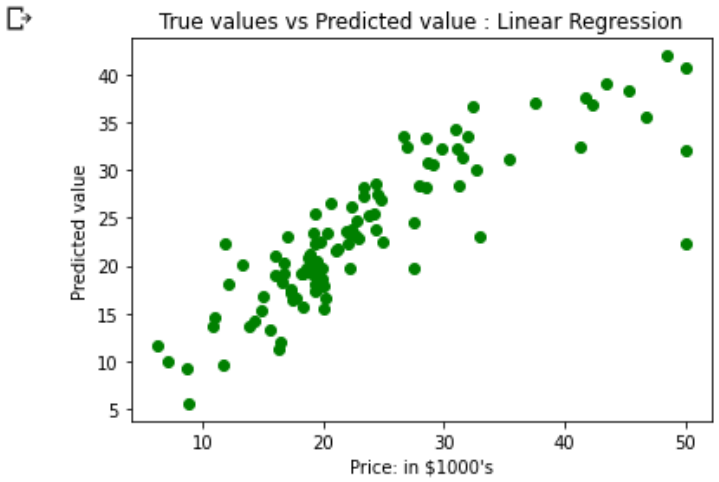
```
Mean Square Error: 25.419587126821682
RMSE :  5.041784121402034
```

#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 values vs Predicted value : Linear Regression")
plt.show()
```



PRACTICAL: 05

Aim: Connecting Pandas to a PostgreSQL Database with SQLAlchemy.

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

Program:

```
from sqlalchemy import create_engine
import pandas as pd

connection=create_engine("postgresql://postgres:admin@localhost:5432/Computer19")
print("Connection established successfully!!")

df=pd.read_sql_query('SELECT * FROM "student"',connection)
print(df)
```

Output:

```
Connection established successfully!!
Empty DataFrame
Columns: [name, pen]
Index: []
```

Practical 5.2:

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

Program:

```
import pandas as pd
import psycopg2

connection=psycopg2.connect(database='Computer19',user='postgres',password='admin',host
='localhost',port='5432')
print("Database opened successfully!!")

cursor=connection.cursor()
cursor.execute("CREATE TABLE bookIssued(bookId INT PRIMARY KEY NOT
NULL,bookName TEXT NOT NULL,Quantity INT NOT NULL);")
cursor.execute("INSERT INTO bookIssued (bookId,bookName,Quantity)
VALUES(1001,'Python for Dummies',15)")
cursor.execute("INSERT INTO bookIssued (bookId,bookName,Quantity)
VALUES(1002,'Python for Data Science',40)")
cursor.execute("INSERT INTO bookIssued (bookId,bookName,Quantity)
VALUES(1003,'Computer networks',30)")
print("Table is created successfully!!")

df=pd.read_sql_query('select bookId,bookName from bookIssued',connection)
print(df)

connection.commit()
connection.close()
```

Output:

From Notebook:

```
Database opened successfully!!
Table is created successfully!!
   bookid                 bookname
0    1001       Python for Dummies
1    1002  Python for Data Science
2    1003        Computer networks
```

From postgreSQL

	bookid [PK] integer	bookname text	quantity integer
1	1001	Python for Dummies	15
2	1002	Python for Data Science	40
3	1003	Computer networks	30

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

Program:

```
from sqlalchemy import create_engine
from sqlalchemy.types import Integer,Text,String,DateTime
import pandas as pd

connection=create_engine("postgresql://postgres:admin@localhost:5432/Computer19")
print("Connection established successfully!!")

job_df=pd.read_csv("E:/RNGPIT/STUDIES/SEM5/PDS/PRACTICALS/nyc-jobs.csv")
table_name='nyc_jobs'
job_df.to_sql(table_name,connection,if_exists='replace',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})

df_table=pd.read_sql_table(table_name,connection)
print("Table created successfully")
print(df_table)
```

Output:

	index bigint	Job ID bigint	Agency text	Posting Type text	# Of Positions bigint	Business Title text
1	0	87990	DEPARTMENT OF BUSINESS SERV.	Internal	1	Account Manager
2	1	97899	DEPARTMENT OF BUSINESS SERV.	Internal	1	EXECUTIVE DIRECTOR, BUSINESS DEVELOPMENT
3	2	132292	NYC HOUSING AUTHORITY	External	52	Maintenance Worker - Technical Services-Heating Unit
4	3	132292	NYC HOUSING AUTHORITY	Internal	52	Maintenance Worker - Technical Services-Heating Unit
5	640	370128	TAXI & LIMOUSINE COMMISSION	Internal	1	Software Architect
6	4	133921	NYC HOUSING AUTHORITY	Internal	50	Temporary Painter
7	5	133921	NYC HOUSING AUTHORITY	External	50	Temporary Painter
8	6	137433	DEPT OF HEALTH/MENTAL HYGIENE	Internal	1	Contract Analyst
9	7	138531	DEPT OF ENVIRONMENT PROTECTION	Internal	1	Associate Chemist

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

Program:

```
from sqlalchemy import create_engine
import pandas as pd

connection=create_engine("postgresql://postgres:admin@localhost:5432/Computer19")
print("Connection established successfully!!")

df=pd.read_sql('SELECT * FROM "bookissued"',connection)
print(df)
```

Output:

```
Connection established successfully!!
   bookid                 bookname  quantity
0    1001       Python for Dummies        15
1    1002  Python for Data Science        40
2    1003        Computer networks        30
```

PRACTICAL: 07

AIM: Working with Data Shaping

Practical 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**

Bag of Words (BoW) model is the representation of text in numbers.

There are 11 Unique Words -> **'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:

Representation of Text in numbers:

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of review
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	0	1	0	0	1	1	6

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

Conclusions:

1. The Bag of Words model just creates a set of vectors containing the count of word occurrences in the document (reviews).
2. However, Bag of Words vectors are easy to interpret.

Program:

```
import pandas as pd
from sklearn.feature_extraction.text import *
sentence1="The movie is very scary and long"
sentence2="The movie is not scary and is long"
sentence3="The movie is spooky and good"
CountVec=CountVectorizer(ngram_range=(1,1),stop_words='english')
Count_data=CountVec.fit_transform([sentence1,sentence2,sentence3])
cv_dataframe=pd.DataFrame(Count_data.toarray(),columns=CountVec.get_feature_names())
print(cv_dataframe)
```

Output:

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

Practical 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**

What is Inverse Document Frequency (IDF) model?

IDF is a measure of how important a term is. We need the IDF value because computing just the TF alone is not sufficient to understand the importance of words:

$$tf_{t,d} = \frac{n_{t,d}}{Number\ of\ terms\ in\ the\ document}$$

We can calculate the IDF values for the all the words in Review 2:

IDF('this') = log(number of documents/number of documents containing the word 'this') = log(3/3) = log(1) = 0

Similarly,

IDF('movie',) = log(3/3) = 0
IDF('is') = log(3/3) = 0
IDF('not') = log(3/1) = log(3) = 1.09
0.48 IDF('scary') = log(3/2) = 0.18
IDF('and') = log(3/3) = 0
IDF('slow') = log(3/1) = 0.48

We can calculate the IDF values for each word like this. Thus, the IDF values for the entire vocabulary would be:

Term	Review 1	Review 2	Review 3	IDF
This	1	1	1	0.00
movie	1	1	1	0.00
is	1	2	1	0.00
very	1	0	0	0.48
scary	1	1	0	0.18
and	1	1	1	0.00
long	1	0	0	0.48
not	0	1	0	0.48
slow	0	1	0	0.48
spooky	0	0	1	0.48
good	0	0	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.

$$(tf\_idf)_{t,d} = tf_{t,d} * idf_t$$

We can now calculate the TF-IDF score for every word in Review 2:

TF-IDF('this', Review 2) = TF('this', Review 2) * IDF('this') = 1/8 * 0 = 0

Similarly,

TF-IDF('movie', Review 2) = 1/8 * 0 = 0
0 TF-IDF('is', Review 2) = 1/4 * 0 = 0
TF-IDF('not', Review 2) = 1/8 * 0.48 = 0.06
TF-IDF('scary', Review 2) = 1/8 * 0.18 = 0.0225
0.023 TF-IDF('and', Review 2) = 1/8 * 0 = 0
TF-IDF('slow', Review 2) = 1/8 * 0.48 = 0.06

Similarly, we can calculate the TF-IDF scores for all the words with respect to all the reviews:

Term	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	0.022	0.000
and	1	1	1	0.00	0.000	0.000	0.000
long	1	0	0	0.48	0.068	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

We have now obtained the TF-IDF scores for our vocabulary. 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.

Conclusions:

1. The TF-IDF model contains information on the more important words and the less important ones as well.
2. Also, it usually performs better in machine learning models than the BoW model.

Program:

```
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('Review 1',analyze(sentence1))
print('Review 2',analyze(sentence2))
print('Review 3',analyze(sentence3))
print('The words are:',vectorizer.get_feature_names())
print('tfid of words:',X.toarray())
```

Output:

```
Review 1 ['the', 'movie', 'is', 'very', 'scary', 'and', 'long']
Review 2 ['the', 'movie', 'is', 'not', 'scary', 'and', 'is', 'long']
Review 3 ['the', 'movie', 'is', 'spooky', 'and', 'good']
The words are: ['and', 'good', 'is', 'long', 'movie', 'not', 'scary', 'spooky', 'the', 'very']
tfid of words: [[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.        ]]
```

PEN NO : 190840131034

PRACTICAL-10

Aim : Mini Project related to Data Science Applications.

Project Title: Football Players Analysis

Program :

#import required libraries for analysis

```
import pandas as pd

from matplotlib import pyplot as plt
```

#read the csv file

```
fifa=pd.read_csv("players_20.csv")
```

#print the dataset

```
fifa
```

	sofifa_id	player_url	short_name	long_name	age	dob	height_cm	weight_kg	nationality	club	...	lwb	ldm	cdm
0	158023	https://sofifa.com/player/158023/lionel-messi/...	L. Messi	Lionel Andrés Messi Cuccittini	32	1987-06-24	170	72	Argentina	FC Barcelona	...	68+2	66+2	66+2
1	20801	https://sofifa.com/player/20801/c-ronaldo-dos-...	Cristiano Ronaldo	Cristiano Ronaldo dos Santos Aveiro	34	1985-02-05	187	83	Portugal	Juventus	...	65+3	61+3	61+3
2	190871	https://sofifa.com/player/190871/neymar-da-sil...	Neymar Jr	Neymar da Silva Santos Junior	27	1992-02-05	175	68	Brazil	Paris Saint-Germain	...	66+3	61+3	61+3
3	200389	https://sofifa.com/player/200389/jan-oblak/20/...	J. Oblak	Jan Oblak	26	1993-01-07	188	87	Slovenia	Atlético Madrid	...	NaN	NaN	NaN
4	183277	https://sofifa.com/player/183277/eden-hazard/2...	E. Hazard	Eden Hazard	28	1991-01-07	175	74	Belgium	Real Madrid	...	66+3	63+3	63+3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18273	245006	https://sofifa.com/player/245006/shuai-shao/20...	Shao Shuai	邵帅	22	1997-03-10	186	79	China PR	Beijing Renhe FC	...	43+2	42+2	42+2
18274	250995	https://sofifa.com/player/250995/mingjie-xiao/...	Xiao Mingjie	Mingjie Xiao	22	1997-01-01	177	66	China PR	Shanghai SIPG FC	...	44+2	43+2	43+2
18275	252332	https://sofifa.com/player/252332/wei-zhang/20/...	Zhang Wei	张威	19	2000-05-16	186	75	China PR	Hebei China Fortune FC	...	47+2	49+2	49+2

18275	252332	https://sofifa.com/player/252332/wei-zhang/20/...	Zhang Wei	张威	19	2000-05-16	186	75	China PR	Hebei China Fortune FC	...	47+2	49+2	49+2
18276	251110	https://sofifa.com/player/251110/haijian-wang/...	Wang Haijian	汪海健	18	2000-08-02	185	74	China PR	Shanghai Greenland Shenhua FC	...	48+2	48+2	48+2
18277	233449	https://sofifa.com/player/233449/ximing-pan/20...	Pan Ximing	潘喜明	26	1993-01-11	182	78	China PR	Hebei China Fortune FC	...	48+2	49+2	49+2

18278 rows × 104 columns

#print only first five rows from the dataset

fifa.head()

Out[4]:

	sofifa_id	player_url	short_name	long_name	age	dob	height_cm	weight_kg	nationality	club	...	lwb	ldm	cdm	
0	158023	https://sofifa.com/player/158023/lionel-messi/...	L. Messi	Lionel Andrés Messi Cuccittini	32	1987-06-24	170	72	Argentina	FC Barcelona	...	68+2	66+2	66+2	6
1	20801	https://sofifa.com/player/20801/c-ronaldo-dos-...	Cristiano Ronaldo	Cristiano Ronaldo dos Santos Aveiro	34	1985-02-05	187	83	Portugal	Juventus	...	65+3	61+3	61+3	6
2	190871	https://sofifa.com/player/190871/neymar-da-sil...	Neymar Jr	Neymar da Silva Santos Junior	27	1992-02-05	175	68	Brazil	Paris Saint-Germain	...	66+3	61+3	61+3	6
3	200389	https://sofifa.com/player/200389/jan-oblak/20/...	J. Oblak	Jan Oblak	26	1993-01-07	188	87	Slovenia	Atlético Madrid	...	NaN	NaN	NaN	N
4	183277	https://sofifa.com/player/183277/eden-hazard/2...	E. Hazard	Eden Hazard	28	1991-01-07	175	74	Belgium	Real Madrid	...	66+3	63+3	63+3	6

4	183277	https://sofifa.com/player/183277/eden-hazard/2...	E. Hazard	Eden Hazard	28	1991-01-07	175	74	Belgium	Real Madrid	...	66+3	63+3	63+3	6

5 rows × 104 columns

#check what are the columns

```
for col in fifa.columns:
  print(col)
```

```
sofifa_id
player_url
short_name
long_name
age
dob
height_cm
weight_kg
nationality
club
overall
potential
value_eur
wage_eur
player_positions
preferred_foot
international_reputation
weak_foot
skill_moves
work_rate
body_type
real_face
release_clause_eur
player_tags
team_position
team_jersey_number
loaned_from
joined
contract_valid_until
nation_position
nation_jersey_number
pace
shooting
passing
dribbling
defending
physic
gk_diving
gk_handling
gk_kicking
gk_reflexes
gk_speed
gk_positioning
player_traits
attacking_crossing
attacking_finishing
attacking_heading_accuracy
attacking_short_passing
attacking_volleys
skill_dribbling
skill_curve
skill_fk_accuracy
skill_long_passing
skill_ball_control
movement_acceleration
movement_sprint_speed
movement_agility
```

```
movement_reactions
movement_balance
power_shot_power
power_jumping
power_stamina
power_strength
power_long_shots
mentality_aggression
mentality_interceptions
mentality_positioning
mentality_vision
mentality_penalties
mentality_composure
defending_marking
defending_standing_tackle
defending_sliding_tackle
goalkeeping_diving
goalkeeping_handling
goalkeeping_kicking
goalkeeping_positioning
goalkeeping_reflexes
ls
st
rs
lw
lf
cf
rf

rw
lam
cam
ram
lm
lcm
cm
rcm
rm
lwb
ldm
cdm
rdm
rwb
lb
lcb
cb
rcb
rb
```

#shape of the dataset

fifa.shape

```
Out[6]: (18278, 104)
```

fifa['nationality'].value_counts()

```
Out[7]: England        1667
        Germany        1216
        Spain          1035
        France          984
        Argentina       886
                       ...
        Gibraltar         1
        Guam              1
        Hong Kong         1
        South Sudan       1
        Jordan            1
        Name: nationality, Length: 162, dtype: int64
```

fifa['nationality'].value_counts()[0:10]

```
Out[8]: England        1667
        Germany        1216
        Spain          1035
        France          984
        Argentina       886
        Brazil          824
        Italy           732
        Colombia        591
        Japan           453
        Netherlands     416
        Name: nationality, dtype: int64
```

fifa['nationality'].value_counts()[0:5]

```
Out[9]: England      1667
        Germany      1216
        Spain        1035
        France        984
        Argentina     886
        Name: nationality, dtype: int64
```

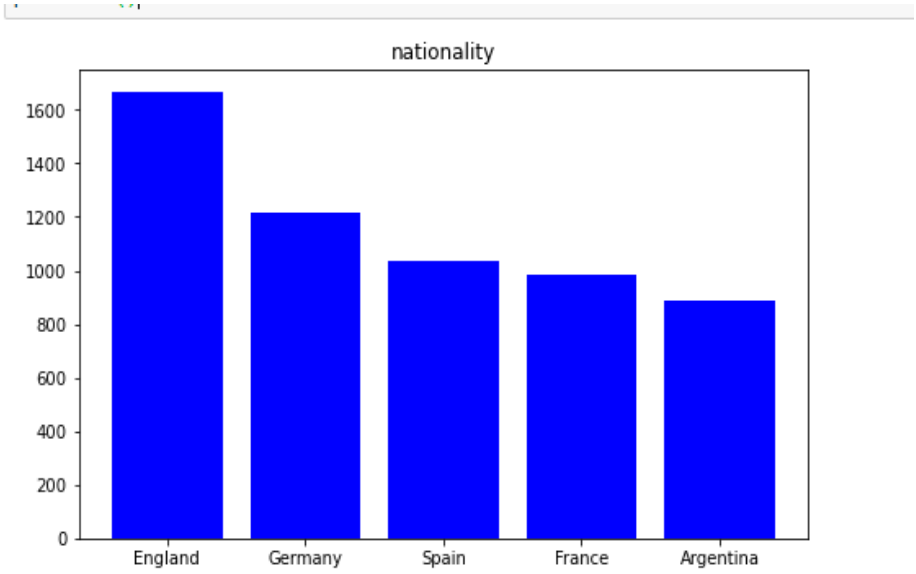
#extract only key or index and print

```
fifa['nationality'].value_counts()[0:5].keys()
```

```
Out[10]: Index(['England', 'Germany', 'Spain', 'France', 'Argentina'], dtype='object')
```

#visualize the graphical representation using 'bar' plot

```
plt.figure(figsize=(8,5))

plt.bar(list(fifa['nationality'].value_counts()[0:5].keys()),list(fifa['nationality'].value_counts()[0:5]),color='b')

plt.title('nationality')

plt.show()
```



#to extract some specified columns from the dataset for the analysis

```
player_salary = fifa[['short_name','wage_eur']]
player_salary.head()
```

Out[13]:

	short_name	wage_eur
0	L. Messi	565000
1	Cristiano Ronaldo	405000
2	Neymar Jr	290000
3	J. Oblak	125000
4	E. Hazard	470000

#to sort the salary using columns 'wage_eur'

```
player_salary=player_salary.sort_values(by=['wage_eur'],ascending=False)
player_salary.head()
```

Out[15]:

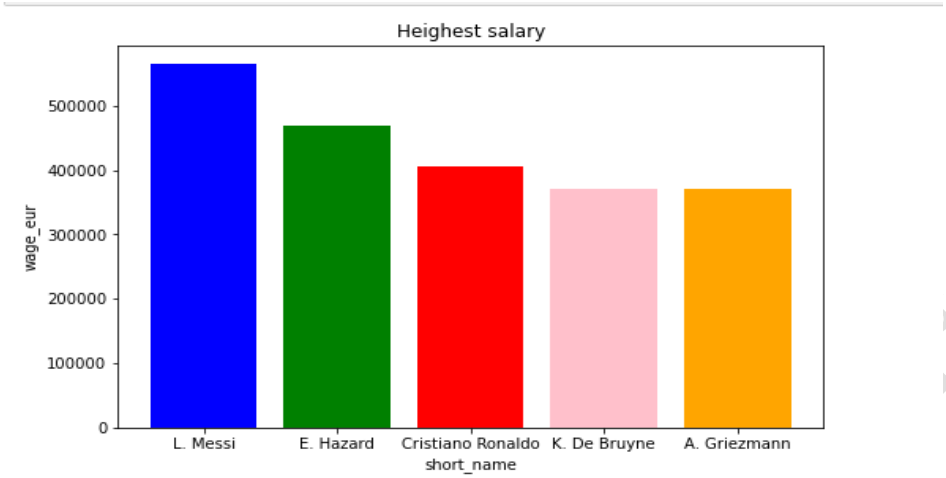
	short_name	wage_eur
0	L. Messi	565000
4	E. Hazard	470000
1	Cristiano Ronaldo	405000
5	K. De Bruyne	370000
22	A. Griezmann	370000

#visualize in-terms of graph for whose player salary is high

```
plt.figure(figsize=(8,5))

plt.bar(list(player_salary['short_name'])[0:5],list(player_salary['wage_eur'])[0:5],color=["blue","
green","red","pink","orange"])

plt.xlabel('short_name')

plt.ylabel('wage_eur')
```

plt.title('Heighest salary')

plt.show()



#simple analysis: players for the Germany

fifa['nationality']=='Germany'

```
Out[17]: 0        False
         1        False
         2        False
         3        False
         4        False
                  ...
         18273    False
         18274    False
         18275    False
         18276    False
         18277    False
         Name: nationality, Length: 18278, dtype: bool
```

#Germany

Germany=fifa[fifa['nationality']=='Germany']

Germany.head(10)

	sofifa_id	player_url	short_name	long_name	age	dob	height_cm	weight_kg	nationality	club	...	lwb	ldm	cdm
6	192448	https://sofifa.com/player/192448/marc-andre-te...	M. ter Stegen	Marc-André ter Stegen	27	1992-04-30	187	85	Germany	FC Barcelona	...	NaN	NaN	NaN
31	167495	https://sofifa.com/player/167495/manuel-neuer/...	M. Neuer	Manuel Neuer	33	1986-03-27	193	92	Germany	FC Bayern München	...	NaN	NaN	NaN
36	182521	https://sofifa.com/player/182521/toni-kroos/20...	T. Kroos	Toni Kroos	29	1990-01-04	183	76	Germany	Real Madrid	...	75+3	79+3	79+3
37	188350	https://sofifa.com/player/188350/marco-reus/20...	M. Reus	Marco Reus	30	1989-05-31	180	71	Germany	Borussia Dortmund	...	69+3	65+3	65+3
49	178603	https://sofifa.com/player/178603/mats-hummels/...	M. Hummels	Mats Hummels	30	1988-12-16	191	94	Germany	Borussia Dortmund	...	77+3	83+3	83+3
55	222492	https://sofifa.com/player/222492/leroy-sane/20...	L. Sané	Leroy Sané	23	1996-01-11	183	75	Germany	Manchester City	...	65+2	59+2	59+2
61	212622	https://sofifa.com/player/212622/joshua-kimmic...	J. Kimmich	Joshua Kimmich	24	1995-02-08	176	73	Germany	FC Bayern München	...	83+3	82+3	82+3
70	189596	https://sofifa.com/player/189596/thomas-muller...	T. Müller	Thomas Müller	29	1989-09-13	186	75	Germany	FC Bayern München	...	69+3	68+3	68+3
77	212190	https://sofifa.com/player/212190/niklas-sule/2...	N. Süle	Niklas Süle	23	1995-09-03	195	97	Germany	FC Bayern München	...	71+2	77+2	77+2
109	235790	https://sofifa.com/player/235790/kai-havertz/2...	K. Havertz	Kai Havertz	20	1999-06-11	188	83	Germany	Bayer 04 Leverkusen	...	65+2	63+2	63+2

10 rows × 104 columns

#whose the tallest german player

Germany.sort_values(by=['height_cm'],ascending=False).head()

	sofifa_id	player_url	short_name	long_name	age	dob	height_cm	weight_kg	nationality	club	...	lwb	ldm	cdm
8016	236831	https://sofifa.com/player/236831/aaron-seydel/...	A. Seydel	Aaron Seydel	23	1996-02-07	199	90	Germany	1. FSV Mainz 05	...	46+2	43+2	43+
1217	200212	https://sofifa.com/player/200212/michael-esser...	M. Esser	Michael Esser	31	1987-11-22	198	97	Germany	Hannover 96	...	NaN	NaN	Na
1389	199833	https://sofifa.com/player/199833/lars-unnersta...	L. Unnerstall	Lars Unnerstall	28	1990-07-20	198	103	Germany	PSV	...	NaN	NaN	Na
11859	167437	https://sofifa.com/player/167437/dominik-stroh...	D. Stroh-Engel	Dominik Stroh-Engel	33	1985-11-27	197	94	Germany	SpVgg Unterhaching	...	39+2	45+2	45+
13576	239746	https://sofifa.com/player/239746/lukas-watkowi...	L. Watkowiak	Lukas Watkowiak	23	1996-03-06	197	103	Germany	SV Wehen Wiesbaden	...	NaN	NaN	Na

5 rows × 104 columns

```
Germany[['short_name','height_cm']].sort_values(by=['height_cm'],ascending=False).head()
```

Out[21]:

	short_name	height_cm
8016	A. Seydel	199
1217	M. Esser	198
1389	L. Unnerstall	198
11859	D. Stroh-Engel	197
13576	L. Watkowiak	197

#whose player is heighest weight

```
Germany.sort_values(by=['weight_kg'],ascending=False).head()
```

	sofifa_id	player_url	short_name	long_name	age	dob	height_cm	weight_kg	nationality	club	...	lwb	ldm	cdm
13576	239746	https://sofifa.com/player/239746/lukas-watkowi...	L. Watkowiak	Lukas Watkowiak	23	1996-03-06	197	103	Germany	SV Wehen Wiesbaden	...	NaN	NaN	NaN
1389	199833	https://sofifa.com/player/199833/lars-unnersta...	L. Unnerstall	Lars Unnerstall	28	1990-07-20	198	103	Germany	PSV	...	NaN	NaN	NaN
518	179783	https://sofifa.com/player/179783/ralf-fahrmann...	R. Fährmann	Ralf Fährmann	30	1988-09-27	197	98	Germany	Norwich City	...	NaN	NaN	NaN
1217	200212	https://sofifa.com/player/200212/michael-esser...	M. Esser	Michael Esser	31	1987-11-22	198	97	Germany	Hannover 96	...	NaN	NaN	NaN
77	212190	https://sofifa.com/player/212190/niklas-sule/2...	N. Süle	Niklas Süle	23	1995-09-03	195	97	Germany	FC Bayern München	...	71+2	77+2	77+2

5 rows × 104 columns

```
Germany[['short_name','weight_kg']].sort_values(by=['weight_kg'],ascending=False).head()
```

Out[22]:

	short_name	weight_kg
13576	L. Watkowiak	103
1389	L. Unnerstall	103
518	R. Fährmann	98
1217	M. Esser	97
77	N. Süle	97

#high paid player with his name

Germany[['short_name','wage_eur']].sort_values(by=['wage_eur'],ascending=False).head()

Out[23]:

	short_name	wage_eur
36	T. Kroos	330000
6	M. ter Stegen	250000
55	L. Sané	195000
146	I. Gündoğan	180000
70	T. Müller	170000

#visualize the graphical representation

plt.figure(figsize=(8,5))

plt.bar(list(Germany['short_name'])[0:5],list(Germany['wage_eur'])[0:5],color=["blue","green","red","pink","orange"])

plt.xlabel('short_name')

plt.ylabel('wage_eur')

plt.title('Heighest Paid German Player')

plt.show()