DMI COLLEGE OF ENGINEERING

PALANCHUR CHENNAI – 600 123

DEPARTMENT OF INFORMATION TECHNOLOGY

LABORATORY MANUAL

SUB CODE	:	**CS3362**
SUBJECT TITLE	:	**DATA SCIENCE LABARATORY**
SEMESTER	:	III
YEAR	:	**II**
DEPARTMENT	:	INFORMATION TECHNOLOGY

Vision of the Department

To achieve academic excellence by producing competent IT professionalsby inculcating self discipline and ethical values.

Mission of the Department

DM1: To provide quality education by maintaining high standards

DM 2: To enhance their knowledge and skills in the field of Information Technology.

DM 3: To create Professionals with personal responsibility and ethical leadership.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO I: Technical Proficiency with Employability Skills:

To offer a strong foundation in mathematical, logical and engineering core concepts necessary to plan, analyze and solve engineering problem's and to prepare them for employability skills.

PEO II: Continuous Learning:

To promote students to peruse higher studies and to continue the learning process through research and development in effective technical aspects.

PEO III: Inculcating Entrepreneurial Skills:

To prepare the students to be an active team player, possessing strong interpersonal skills and leadership qualities with entrepreneurial ability.

PEO IV: Ethical Values:

To inculcate ethical values in students on both professional and personal practices.

PROGRAM OUTCOMES (POs)

The Program Outcomes (POs) are described as.

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the

engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs):

- **PSO1: Analytical Thinking:** Identify, formulate and solve engineering problems by applying mathematical foundations and algorithmic principles in IT environment to meet challenges.
- **PSOII: Principles Of Information Technology:** Analyze, design and develop Software, Multimedia, Web Applications and Networking Technologies for an efficient design of Information based systems with high professional skills.
- **PSOIII: Engineering Ethics:** Understand best practices, ethical standards and replicate the same for the industry, research and social needs.

INSTRUCTIONS TO STUDENTS FOR WRITING THE RECORD

In the record, the index page should be filled properly by writing the corresponding experiment number, experiment name , date on which it was done and the page number.

On the **right side** page of the record following has to be written:

1. **Title**: The title of the experiment should be written in the page in capital letters.
2. In the left top margin, experiment number and date should be written.
3. **Aim**: The purpose of the experiment should be written clearly.
4. **Procedure/Algorithm:** Steps for doing the experiment should be briefly described
5. **Results**: The results of the experiment must be summarized in writing and should be fulfilling the aim.

Observations:

i) Data should be clearly recorded using Tabular Columns.

Course outcomes

CO1	Develop code for classical Encryption Techniques to solve the problems.
CO2	Build cryptosystems by applying symmetric and public key encryption algorithms.
CO3	Construct code for authentication algorithms.
CO4	Develop a signature scheme using Digital signature standard.
CO5	Demonstrate the network security system using open source tools

CO, PO, PSO Mappings.

Course Code and Course name	CO	Program Outcomes												PSO		
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**1**	**2**	**3**
CS336 Data science Lab	**CO1**	3	2	1	1	-	-	-	-	1	3	3	3	1	3	2
	CO2	3	2	2	3	-	-	-	-	3	1	3	2	1	3	3
	CO3	3	2	1	3	-	-	-	-	2	1	1	1	3	2	3
	CO4	2	2	1	3	-	-	-	-	2	3	2	3	3	3	1
	CO5	1	2	3	1	-	-	-	-	2	1	3	1	1	3	3
Average		2	2	2	2	1	-	-	-	2	2	2	2	2	3	2

INDEX

SI.NO	DATE	EXPERIMENTS	PAGE NO	SIGN
1.		Download, install and explore the features of numpy, scipy, jupyter, statsmodels and pandas packages		
2.a)		perform basic data analytics communication process with numpy		
2.b)		Working with Numpy arrays		
3.		Working with Pandas data frames		
4.a)		Reading data from text files		
4.b)		Reading data from excel files does exploring various commands		
4.c)		Exploring various commands for doing descriptive analytics on the Iris data set.		
5.a)		Use the diabetes data set from UCI and Pima Indians Diabetes data set for performing the following: Univariate analysis: Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis.		
5.b)		Linear Regression and Logistic Regression with the Diabetes Dataset Using Python Machine Learning		
5.c)		Use the diabetes data set from UCI and Pima Indians Diabetes data set for performing the following: Multiple Regression		
5.d)		Also compare the results of the above analysis for the two data sets.		
6.a)		Apply and explore various plotting functions on UCI data sets. Density and contour plots		
6.b)		Apply and explore various plotting functions like Correlation and scatter plots on UCI data sets		
6.c)		Apply and explore histograms and three dimensional plotting functions on UCI		

		data sets		
7.		**Visualizing Geographic Data with Basemap**		

EX. NO – 1

DATE:

DOWNLOAD, INSTALL AND EXPLORE THE FEATURES OF NUMPY, SCIPY, JUPYTER, STATSMODELS AND PANDAS PACKAGES

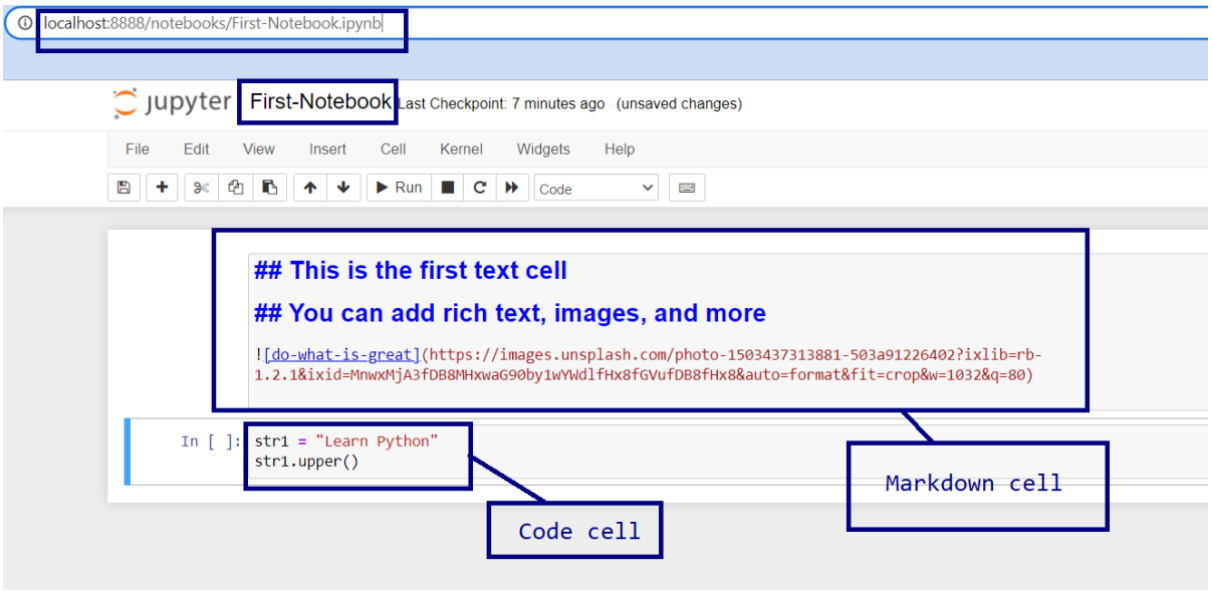
AIM:

To download, install and explore the features of NumPy, SciPy, Jupyter, Statsmodels and Pandas packages.

PROCEDURE:

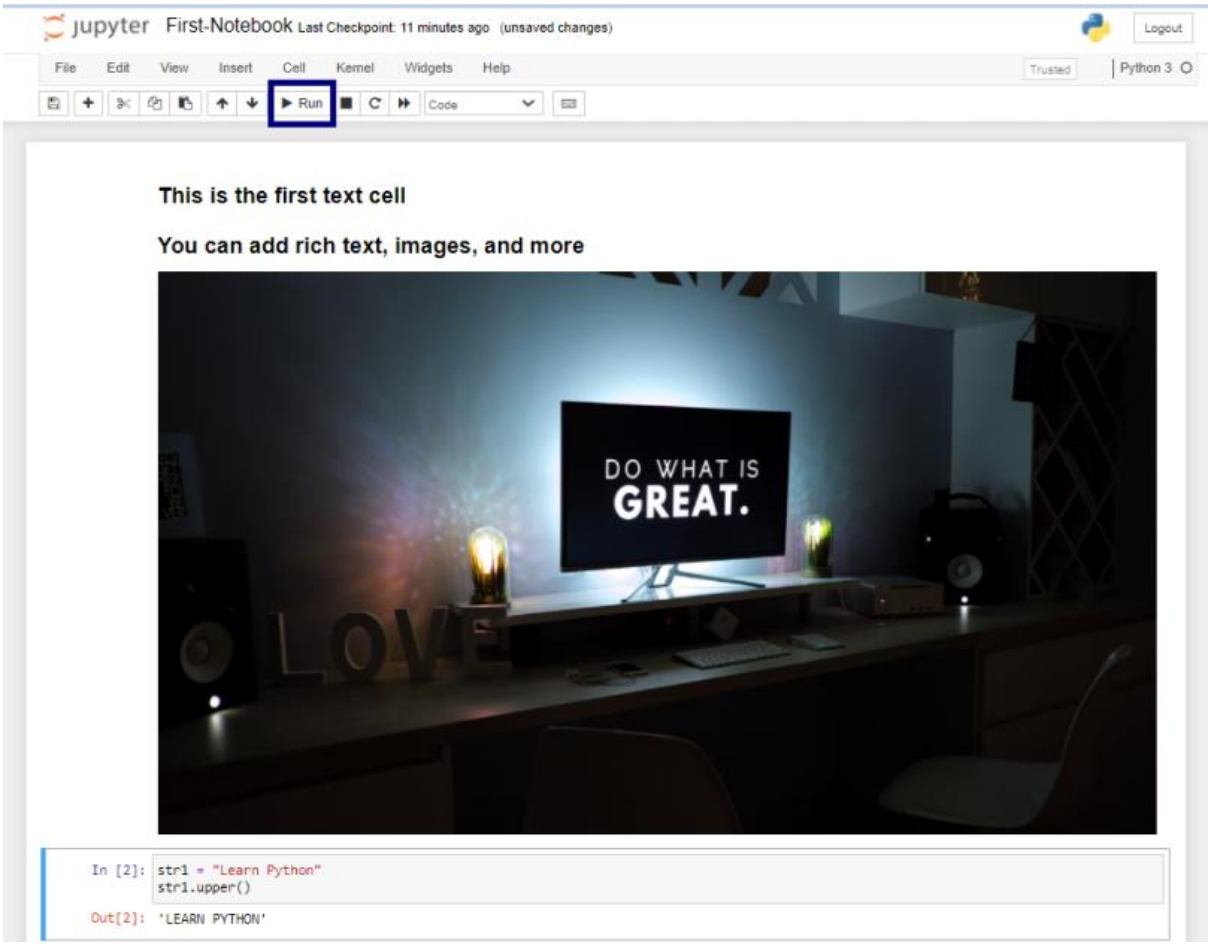
Jupyter Notebook is an interactive browser-based platform for scientific computing and is widely used in data science. In addition to providing an interactive coding platform, Jupyter Notebook supports both code and text cells. The text cells allow Markdown formatting. So Plain text, images, LaTex math equations can be used to explain project's workflow.

For example, the following image shows how can write both Markdown and code by specifying the cell type.



Markdown and Code Cells in Jupyter Notebook

To run a cell, the **Run** [▶] button can be pressed or press **Shift + Enter** to run a cell. The headings and images are rendered after running the cells.



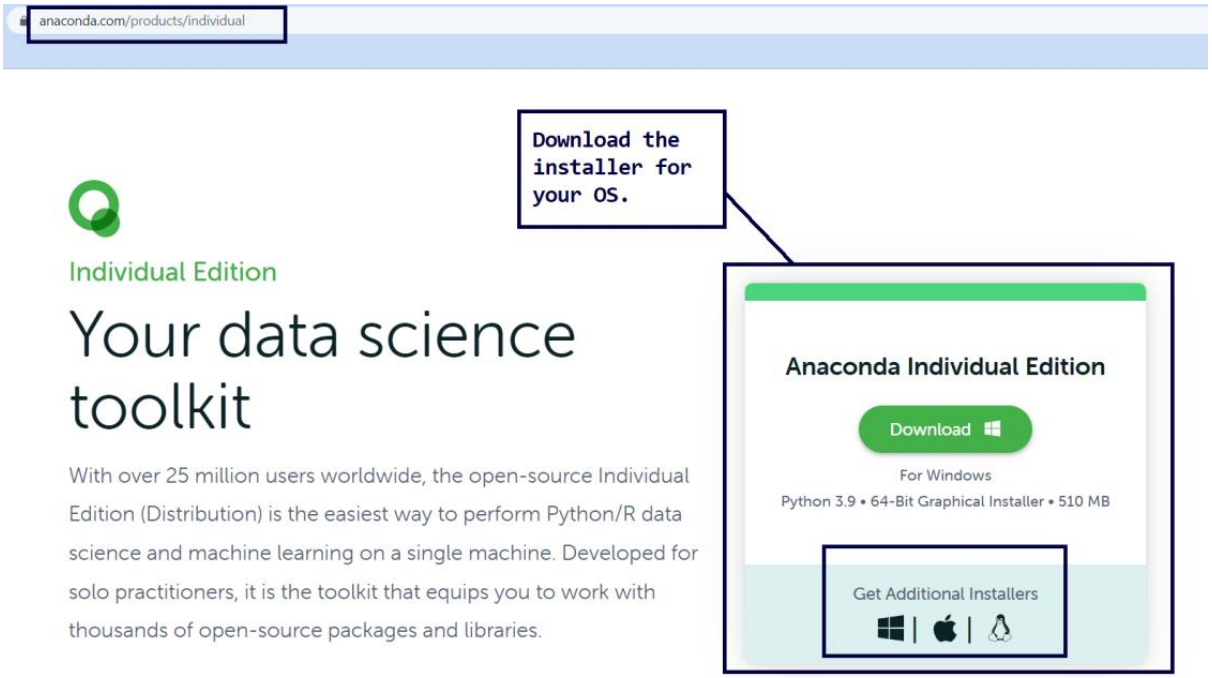
Jupyter Notebook Cells

Installation Using the Anaconda Distribution

It's recommended to use the Anaconda distribution of Python. In addition to Python, it comes with several useful data science packages pre-installed. The installation also includes Jupyter tools like Jupyter Notebook and JupyterLab.

Steps in this installation.

Step 1: Head over to the official website of Anaconda. Then, navigate to anaconda.com/products/individual. And download the installer corresponding to your operating system.

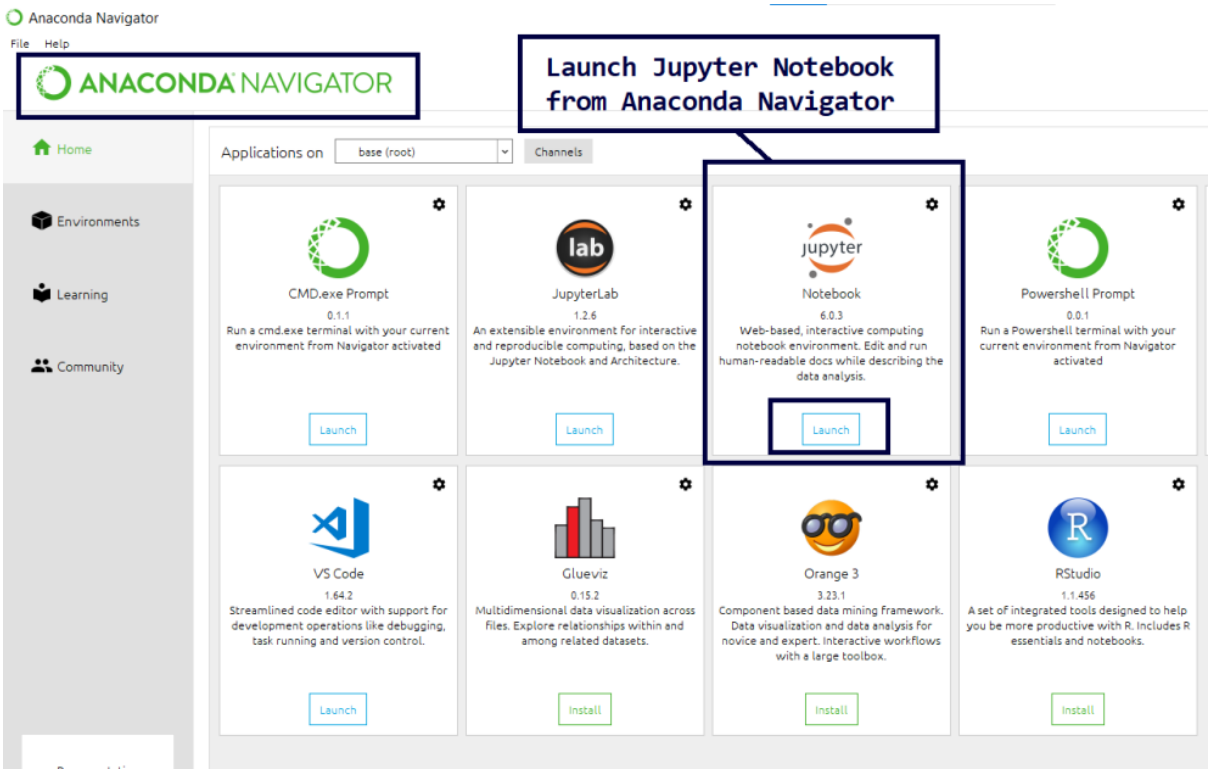


Installing the Anaconda Distribution

Step 2: Now, run the installer. Follow the prompts on your screen to complete the installation. The installation will typically take a few minutes. ⏳

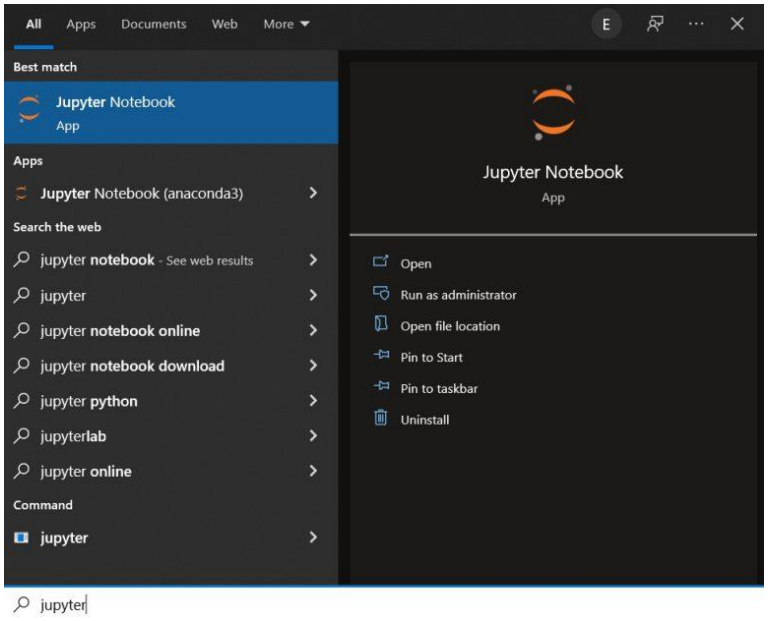
Launch Jupyter Notebook once the installation process is completed.

Step 3: Once installation is completed, launch **Anaconda Navigator**. From the navigator, click on the Launch option in the Jupyter Notebook tab, as shown below:



Launching Jupyter Notebook from Anaconda Navigator

Or the Jupyter Notebook shortcut is used to launch, as illustrated below:



Also launch jupyter notebook from the **Anaconda Command Prompt**.

NUMPY:

- Introduces objects for multidimensional arrays and matrices, as well as functions that allow to easily perform advanced mathematical and statistical operations on those objects

- provides vectorization of mathematical operations on arrays and matrices which significantly improves the performance
- many other python libraries are built on NumPy

SciPy:

- collection of algorithms for linear algebra, differential equations, numerical integration, optimization, statistics and more
- part of SciPy Stack
- built on NumPy

Pandas:

- adds data structures and tools designed to work with table-like data (similar to Series and Data Frames in R)
- provides tools for data manipulation: reshaping, merging, sorting, slicing, aggregation etc.
- allows handling missing data

Statsmodels:

statsmodels is a Python package that provides a complement to scipy for statistical computations including descriptive statistics and estimation and inference for statistical models.

Result: Above program was executed successfully.

EX. NO – 2A

DATE:

PERFORM BASIC DATA ANALYTICS COMMUNICATION PROCESS WITH NUMPY

AIM

To write a program for construct and perform the basic data analytics communication process with numpy and other packages

PROCEDURE

STEP 1: Import the numpy package file for our program. STEP 2: Perform the

Array operation using numpy package

STEP 3: Printing type of arr object by type(arr)

STEP 4: Print Array dimension by using ndim

STEP 5: Print the shape of the array by using shape.

STEP 6: Print the size of the array by using size

STEP 7: End

Source code:

A) ARRAY INNUMPY

```
import numpy as np
# Creating array object
arr = np.array( [[ 1, 2, 3], [ 4, 2, 5]] )
print(arr)
# Printing type of arr object
print("Array is of type: ", type(arr))
# Printing array dimensions (axes)
print("No. of dimensions: ", arr.ndim)
# Printing shape of array
print("Shape of array: ",arr.shape)
# Printing size (total number of elements) of array
print("Size of array: ", arr.size)
# Printing type of elements in array
print("Array stores elements of type: ", arr.dtype)
```

OUTPUT

```
[[1 2 3]
 [4 2 5]]
Array is of type:  <class 'numpy.ndarray'>
No. of dimensions:  2
Shape of array:  (2, 3)
Size of array:  6
Array stores elements of type:  int32
```

B) ARRAYCREATION

```
# array creation techniques

import numpy as np
# Creating array from list with type float
a = np.array([[1, 2, 4], [5, 8, 7]], dtype = 'float')
print ("Array created using passed list:\n", a)
# Creating array from tuple
b = np.array((1 , 3, 2))
```

```
print ("\nArray created using passed tuple:\n", b)
# Creating a 3X4 array with all zeros
c = np.zeros((3, 4))
print ("\nAn array initialized with all zeros:\n", c)
# Create a constant value array of complex type
d = np.full((3, 3), 6, dtype = 'complex')
print ("\nAn array initialized with all 6s.""Array type is complex:\n", d)
# Create an array with random values
e = np.random.random((2, 2))
print ("\nA random array:\n", e)
# Create a sequence of integers
# from 0 to 30 with steps of 5
f = np.arange(0, 30, 5)
print ("\nA sequential array with steps of 5:\n", f)
# Create a sequence of 10 values in range 0 to 5
g = np.linspace(0, 5, 10)
print ("\nA sequential array with 10 values between""0 and 5:\n", g)
# Reshaping 3X4 array to 2X2X3 array
arr = np.array([[1, 2, 3, 4], [5, 2, 4, 2], [1, 2, 0, 1]])
newarr = arr.reshape(2, 2, 3)
print ("\nOriginal array:\n", arr)
 print ("Reshaped array:\n", newarr)
# Flattenarray
arr = np.array([[1, 2, 3], [4, 5, 6]])
flarr = arr.flatten()
print ("\nOriginal array:\n", arr)
print ("Fattened array:\n", flarr)
```

OUTPUT

```
Array created using passed list:
 [[1. 2. 4.]
 [5. 8. 7.]]

Array created using passed tuple:
 [1 3 2]

An array initialized with all zeros:
```

```
[[0. 0. 0. 0.]
[0. 0. 0. 0.]
[0. 0. 0. 0.]]

An array initialized with all 6s.Array type is complex:
[[6.+0.j 6.+0.j 6.+0.j]
[6.+0.j 6.+0.j 6.+0.j]
[6.+0.j 6.+0.j 6.+0.j]]

A random array:
[[0.18383822 0.28510541]
[0.85234723 0.42182804]]

A sequential array with steps of 5:
[ 0  5 10 15 20 25]

A sequential array with 10 values between0 and 5:
[0.         0.55555556 1.11111111 1.66666667 2.22222222 2.77777778
3.33333333 3.88888889 4.44444444 5.        ]

Original array:
[[1 2 3 4]
[5 2 4 2]
[1 2 0 1]]
Reshaped array:
[[[1 2 3]
 [4 5 2]]

[[4 2 1]
 [2 0 1]]]

Original array:
[[1 2 3]
[4 5 6]]
Fattened array:
[1 2 3 4 5 6]
```

C) ARRAYINDEXING

```
# indexing in
numpy import
numpy as np
```

```
# An
exemplararray
arr = np.array([[-1, 2, 0, 4], [4, -0.5, 6, 0], [2.6, 0, 7, 8], [3, -7, 4, 2.0]])
# Slicing array
temp = arr[:2,::2]
print ("Array with first 2 rows and alternatecolumns(0 and 2):\n", temp)
# Integer array indexing example
temp = arr[[0, 1, 2, 3], [3, 2, 1, 0]]
print ("\nElements at indices (0, 3), (1, 2), (2, 1),""(3, 0):\n", temp)
# boolean array indexing example
cond = arr > 0
# cond is a boolean array
temp = arr[cond]
print ("\nElements greater than 0:\n", temp)
```

OUTPUT

```
Array with first 2 rows and alternatecolumns(0 and 2):
 [[-1.  0.]
 [ 4.  6.]]
Elements at indices (0, 3), (1, 2), (2, 1),(3, 0):
 [4. 6. 0. 3.]

Elements greater than 0:
 [2.  4.  4.  6.  2.6 7.  8.  3.  4.  2. ]
```

D) OPERATIONS ON SINGLEARRAY

```
# basic operations on single array

import numpy as np
a = np.array([1, 2, 5, 3])
# add 1 to every element
print ("Adding 1 to every element:", a+1)
# subtract 3 from each element
print ("Subtracting 3 from each element:", a-3)
# multiply each element by 10
print ("Multiplying each element by 10:", a*10)
# square each element
print ("Squaring each element:", a**2)
```

```
# modify existing array
a *= 2
print ("Doubled each element of original array:", a)
# transpose of array
a = np.array([[1, 2, 3], [3, 4, 5], [9, 6, 0]])
print ("\nOriginal array:\n", a)
print ("Transpose of array:\n", a.T)
```

OUTPUT

```
Adding 1 to every element: [2 3 6 4]
Subtracting 3 from each element: [-2 -1  2  0]
Multiplying each element by 10: [10 20 50 30]
Squaring each element: [ 1  4 25  9]
Doubled each element of original array: [ 2  4 10  6]

Original array:
 [[1 2 3]
 [3 4 5]
 [9 6 0]]
Transpose of array:
 [[1 3 9]
 [2 4 6]
 [3 5 0]]
```

E) UNARYOPERATORS

```
# unary operators in
numpy import numpy as
np
arr = np.array([[1, 5, 6], [4, 7, 2], [3, 1, 9]])
# maximum element of array
print ("Largest element is:", arr.max())
print ("Row-wise maximum elements:", arr.max(axis = 1))
# minimum element of array
print ("Column-wise minimum elements:", arr.min(axis = 0))
# sum of array elements
print ("Sum of all array elements:", arr.sum())
# cumulative sum along each row
print ("Cumulative sum along each row:\n", arr.cumsum(axis = 1))
```

OUTPUT

```
Largest element is: 9
Row-wise maximum elements: [6 7 9]
Column-wise minimum elements: [1 1 2]
Sum of all array elements: 38
Cumulative sum along each row:
 [[ 1  6 12]
 [ 4 11 13]
 [ 3  4 13]]
```

F) BINARYOPERATORS

```
# binary operators in
Numpy import numpy as
np
a = np.array([[1, 2], [3, 4]])
b = np.array([[4, 3], [2, 1]])
# add arrays
print ("Array sum:\n", a + b)
# multiply arrays (elementwise multiplication)
print ("Array multiplication:\n", a*b)
# matrix multiplication
print ("Matrix multiplication:\n", a.dot(b))
```

OUTPUT

```
Array sum:
 [[5 5]
 [5 5]]
Array multiplication:
 [[4 6]
 [6 4]]
Matrix multiplication:
 [[ 8  5]
 [20 13]]
```

G) UNIVERSAL FUNCTIONS(ufunc)

```
# universal functions in
```

```
numpy import numpy as np
# create an array of sine values
a = np.array([0, np.pi/2,np.pi])
print ("Sine values of array elements:", np.sin(a))
# exponential values
a = np.array([0, 1, 2, 3])
print ("Exponent of array elements:", np.exp(a))
# square root of array values
print ("Square root of array elements:", np.sqrt(a))
```

OUTPUT

```
Sine values of array elements: [0.0000000e+00 1.0000000e+00 1.2246468e-16]
Exponent of array elements: [ 1.          2.71828183  7.3890561  20.08553692]
Square root of array elements: [0.         1.         1.41421356 1.73205081]
```

H) SORTINGARRAY

```
# Python program to demonstrate sorting in
numpy

import numpy as np
a = np.array([[1, 4, 2], [3, 4, 6], [0, -1, 5]])
# sorted array
print ("Array elements in sorted order:\n", np.sort(a, axis = None))
# sort array row-wise
print ("Row-wise sorted array:\n", np.sort(a, axis = 1))
# specify sort algorithm
print ("Column wise sort by applying merge- sort:\n", np.sort(a, axis = 0,
kind = 'mergesort'))
# Example to show sorting of structured array
# set alias names for dtypes
dtypes = [('name', 'S10'), ('grad_year', int), ('cgpa', float)]
# Values to be put in array
values = [('Hrithik', 2009, 8.5), ('Ajay', 2008, 8.7), ('Pankaj', 2008,
  7.9), ('Aakash', 2009, 9.0)]
# Creating array
arr = np.array(values, dtype = dtypes)
```

```
print ("\nArray sorted by names:\n", np.sort(arr, order = 'name'))
print ("Array sorted by graduation year and then cgpa:\n", np.sort(arr, order
= ['grad_year', 'cgpa']))
```

OUTPUT

```
Array elements in sorted order:
 [-1  0  1  2  3  4  4  5  6]
Row-wise sorted array:
 [[ 1  2  4]
 [ 3  4  6]
 [-1  0  5]]
Column wise sort by applying merge- sort:
 [[ 0 -1  2]
 [ 1  4  5]
 [ 3  4  6]]

Array sorted by names:
 [(b'Aakash', 2009, 9. ) (b'Ajay', 2008, 8.7) (b'Hrithik', 2009, 8.5)
 (b'Pankaj', 2008, 7.9)]
Array sorted by graduation year and then cgpa:
 [(b'Pankaj', 2008, 7.9) (b'Ajay', 2008, 8.7) (b'Hrithik', 2009, 8.5)
 (b'Aakash', 2009, 9. )]
```

Result: Above program was executed successfully.

Ex2B Working with Numpy arrays

Aim:

Working with different instructions used in Numpy array

Procedure

STEP 1: Import the numpy package file for our program.
STEP 2: Perform the Array operation using numpy package
STEP 3: Printing type of arr object by type(arr)
STEP 4: Print Array dimension by using ndim
STEP 5: Print the shape of the array by using shape.
STEP 6: Print the size of the array by using size
STEP 7**:** End

Program

```
import numpy as np
a = np.arange(15).reshape(3, 5)
To print the array
print(a)
To print the shape of the array
print(a.shape)
To the function return the number of dimensions of an array.
print(a.ndim)
This data type object (dtype) informs us about the layout of the array
print(a.dtype.name)
This returns the size (in bytes) of each element of a NumPy array
print(a.itemsize)
This is the total number of elements in the ndarray.
print(a.size)


arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
x = np.where(arr%2 == 1)
print(x)
Slice elements from index 1 to index 5 from the following array
print(arr[1:5])
Get third and fourth elements from the following array and add them.
print(arr[2] + arr[3])
```

Write a NumPy program to convert the values of Centigrade degrees into Fahrenheit degrees and vice versa. Values are stored into a NumPy array.

```
import numpy as np
fvalues =[0,12,45.21,34,99.91,32]
F = np.array(fvalues)
```

```
print("Values in Fahrenheit degrees:")
print(F)
print("Values in  Centigrade degrees:")
print(np.round((5*F/9-5*32/9),2))
```

Output

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]
(3, 5)
2
int32
4
15
(array([0, 2, 4, 6], dtype=int64),)
[2 3 4 5]
7
Values in Fahrenheit degrees:
[ 0.   12.   45.21 34.   99.91 32.  ]
Values in  Centigrade degrees:
[-17.78 -11.11   7.34   1.11  37.73   0.  ]
```

Result: Above program was executed successfully.

Ex.NO :3 Working with Pandas data frames

Aim

To work with Pandas data frames

Procedure

1. Create a dataset with name, city, age and pyscore.
2. Use .head() to show the first few items and .tail() to show the last few items.
3. Get the DataFrame's row labels with .index and its column labels with .columns
4. Get the data types for each column of a Pandas DataFrame with .dtypes
5. Check the amount of memory used by each column.
6. Accessor .loc[], which you can use to get rows or columns by their labels, Pandas offers the accessor .iloc[], which retrieves a row or column by its integer index.
7. Creating a new Series object that represents this new candidate

Program

```
import pandas as pd
data = {
    'name': ['Muthu', 'Anand', 'Ramkumar', 'Roja', 'Robin', 'Rajan', 'Joel'],
    'city': ['Chennai', 'Madurai', 'Tirunelveli', 'Saidapet','Tambaram', 'Irukattukottai', 'Central Station'],
     'age': [41, 28, 33, 34, 38, 31, 37],
  'py-score': [88.0, 79.0, 81.0, 80.0, 68.0, 61.0, 84.0]
}
row_labels = [101, 102, 103, 104, 105, 106, 107]
```

```
df is a variable that holds the reference to your Pandas DataFrame
df=pd.DataFrame(data=data,index=row_labels)
df
```

We can use .head() to show the first few items and .tail() to show the last few items.

```
df.head(n=2)
df.tail(n=2)
```

You can get the DataFrame's row labels with .index and its column labels with .columns

```
df.index
df.columns
```

We can get the data types for each column of a Pandas DataFrame with .dtypes

```
df.dtypes
```

You can even check the amount of memory used by each column with .memory_usage()

```
df.memory_usage()
```

In addition to the accessor .loc[], which you can use to get rows or columns by their labels, Pandas offers the accessor .iloc[], which retrieves a row or column by its integer index.

```
df.loc[101]
df.iloc[0]
```

We can start by creating a new Series object that represents this new candidate

```
john = pd.Series(data=['Jovan', 'Medavakkam', 34, 79],index=df.columns, name =17)
john
df = df.append(john)
df
```

Output

name	city	age	py-score	
101	Muthu	Chennai	41	88.0
102	Anand	Madurai	28	79.0
103	Ramkumar	Tirunelveli	33	81.0

name	city	age	py-score	
104	Roja	Saidapet	34	80.0
105	Robin	Tambaram	38	68.0
106	Rajan	Irukattukottai	31	61.0
107	Joel	Central Station	37	84.0
17	Jovan	Medavakkam	34	79.0

Result: Above program was executed successfully.

EX.NO:4A Reading data from text files

Aim

Write a python program to Reading data from text files

Procedure

1. Type the text file with some data and save as filename.txt.
2. Load the text file using the open() function with write command.
3. Do the functions like read(), write() commands as needed.

Program

```
# Program to show various ways to read and

# write data in a file.

file1 = open("myfile.txt","w")

L = ["This is Delhi \n","This is Paris \n","This is London \n"]


# \n is placed to indicate EOL (End of Line)

file1.write("Hello \n")

file1.writelines(L)

file1.close() #to change file access modes


file1 = open("myfile.txt","r+")


print("Output of Read function is ")

print(file1.read())

print()


# seek(n) takes the file handle to the nth

# bite from the beginning.

file1.seek(0)
```

```
print( "Output of Readline function is ")
print(file1.readline())
print()

file1.seek(0)

# To show difference between read and readline
print("Output of Read(9) function is ")
print(file1.read(9))
print()

file1.seek(0)

print("Output of Readline(9) function is ")
print(file1.readline(9))

file1.seek(0)
# readlines function
print("Output of Readlines function is ")
print(file1.readlines())
print()
file1.close()
```

Output

```
Output of Read function is
Hello
This is Delhi
This is Paris
This is London
```

Output of Readline function is
Hello

Output of Read(9) function is
Hello
Th

Output of Readline(9) function is
Hello

Output of Readlines function is
['Hello \n', 'This is Delhi \n', 'This is Paris \n', 'This is London \n']

Result: Above program was executed successfully.

Ex 4B. Reading data from excel files does exploring various commands

Aim

Write a python program to Reading data from excel files

Procedure

1. First of all create an excel file with some 10 records(10 rows) and 5 columns with numerical data and save them as filename.xlsx format.
2. Reading data from excel files into pandas using Python.
3. Exploring the data from excel files in Pandas.
4. Using functions to manipulate and reshape the data in Pandas.

To view 5 columns from the top and from the bottom of the data.

The shape() method can be used to view the number of rows and columns.

If any column contains numerical data, we can sort that column using the sort_values() method in pandas.

Suppose our data is mostly numerical. We can get the statistical information like mean, max, min, etc. about the data frame using the describe() method.

Program

```
import pandas as pd

data = pd.read_excel (r'd:\mark.xlsx')

data

df = pd.DataFrame(data, columns= ['Name','CGPA'])

print (df)

data.head()

data.tail()

data.shape

sorted_column = data.sort_values(['Name'], ascending = False)

sorted_column['Name'].head(5)

data.describe()
```

Output

	Name	CGPA
1	AARTHI N	8.0
2	ABISHEK R	7.0
3	ABISHEK S	7.3
4	AGUSTHIAN D	6.5
5	AISHWARYA J	7.1
6	AJAY JUDU S	7.5
7	AKASH R	8.0
8	ALAGU SIVA A	6.0
9	ANNIE RUSHMA	6.5
10	AROCKIA PREM KUMAR A	7.5
11	DEVADHARSHINI R	7.6
12	DHARSHINI S	7.7
13	DHESAVAN R	6.6
14	DINESHKUMAR V	5.0
15	FARVESH MUSHRAF S A	7.8
16	GOKUL S	9.2
17	GOPINATH G	8.0
18	HARSHA VARTHINI J U	6.5
19	HEMARAJ T	4.0
20	IMMACULATE A	8.0
21	JAYASRI R	9.1
22	JAYA VARSHINI K	7.8
23	JESICA MARY P	8.5
24	KARTHIKEYAN K	9.0
25	KATHIRVEL A	7.2
26	KEERTHANA L	8.5
27	KEERTHICK RAJA P	7.6
28	KENITH RAJ ANTONY L	7.4
29	KUMARAGURU G	6.3
30	MOHANRAJ R	7.0
31	MURUGADASS A	7.0
32	NARENDRAN K	7.0
33	NATHAN C	1.0
34	POOJA S	7.0
35	PRAKASH K	6.4
36	RAJESWARI V	7.6
37	RAMALINGAM K	1.0
38	RAMYA P	7.0

39	SABARISH S 8.9
40	SAKTHIVEL P 8.0
41	SANJAY M 7.4
42	SATHISH KUMAR P 7.8
43	SELVIN PAULRAJ K 8.0
44	SIVASUKI S 7.4
45	SNEHA S 8.0
46	SRIRAM K 7.0
47	SUNDAR P 7.0
48	SURRYA GANESH V L 6.8
49	VENKATKUMAR K R 8.0
50	VIDHUN K 8.0
51	VIGNESH WARAN K 7.8
52	VIJAY ESHWERRAJ R V 1.0
53	VIMALRAJ J 7.9
54	YOKESHWARAN S 7.8
55	BHARATH 1.0
56	SIMON RAJ 1.0

	CGPA
count	56.000000
mean	6.839286
std	2.051224
min	1.000000
25%	6.750000
50%	7.400000
75%	8.000000
max	9.200000

Result: Above program was executed successfully.

Ex 4c. Exploring various commands for doing descriptive analytics on the Iris data set.

Aim

To explore various commands for doing descriptive analytics on the Iris data set.

Procedure

1. To understand idea behind Descriptive Statistics.
2. Load the packages we will need and also the `iris` dataset.
3. *load_iris()* loads in an object containing the iris dataset, which I stored in `*iris_obj*`.
4. Basic statistics.
5. This number is the number of rows in the dataset, and can be obtained via *`count()`*.
6. Mean for every numeric column
7. Median for every numeric column
8. **variance** is a measure of dispersion, roughly the "average" squared distance of a data point from the mean.
9. The **standard deviation** is the square root of the variance and interpreted as the "average" distance a data point is from the mean.
10. The maximum and minimum values.

Program Code

```
import pandas as pd
from pandas import DataFrame
from sklearn.datasets import load_iris
# sklearn.datasetsincludes common example datasets
# A function to load in the iris dataset
iris_obj = load_iris()
# Dataset preview
iris_obj.data
iris = DataFrame(iris_obj.data, columns=iris_obj.feature_names,index=pd.Inde
x([i for i in range(iris_obj.data.shape[0])])).join(DataFrame(iris_obj.target, colu
mns=pd.Index(["species"]), index=pd.Index([i for i in range(iris_obj.target.shap
e[0])])))
iris # prints iris data
```

Commands

```
iris_obj.feature_names
iris.count()
iris.mean()
iris.median()
iris.var()
iris.std()
iris.max()
iris.min()
iris.describe()
```

Output

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2

149	5.9	3.0	5.1	1.8	2

Result: Above program was executed successfully.

5A. Use the diabetes data set from UCI and Pima Indians Diabetes data set for performing the following:
Univariate analysis: Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis.

Aim:

Analysis the various univariate functions like Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis on dataset like Pima Indian diabetes dataset.

Procedure

1. Download dataset like Pima Indian diabetes dataset. Save them in any drive and call them for process.
2. The mean() function can be used to calculate mean/average of a given list of numbers.
3. The median() method calculates the median (middle value) of the given data set.
4. The mode of a set of data values is the value that appears most often.
5. The var() method calculates the variance for each column.
6. Standard deviation std() is a number that describes how spread out the values are.
7. The skew() method calculates the skew for each column. Skewness refers to a distortion or asymmetry that deviates from the symmetrical bell curve, or normal distribution, in a set of data.
8. **Kurtosis:**
9. It is also a statistical term and an important characteristic of frequency distribution. It determines whether a distribution is heavy-tailed in respect of the normal distribution. It provides information about the shape of a frequency distribution.

Program:

```
import pandas as pd
from scipy.stats import kurtosis
import pylab as p
df = pd.read_csv (r'd:\\diabetes.csv')
print (df)
df1 = pd.DataFrame(df, columns= ['Age','Glucose'])
print (df1)
df1.mean()
df1.median()
df1.mode()
print(df1.var())
df1.std()
print(df1.skew())
print(kurtosis(df, axis=0, bias=True))
```

Dataset download link

https://github.com/npradaschnor/Pima-Indians-Diabetes-Dataset/blob/master/Pima%20Indians%20Diabetes%20Dataset.ipynb

Output

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4
```

```
     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
    Age  Glucose
0    50      148
1    31       85
2    32      183
3    21       89
4    33      137
..  ...      ...
763  63      101
764  27      122
765  30      121
766  47      126
767  23       93

[768 rows x 2 columns]
Age        138.303046
Glucose   1022.248314
dtype: float64
Age        1.129597
Glucose    0.173754
dtype: float64
[ 0.15038274  0.62881333  5.13869066 -0.52449449  7.15957492  3.26125742
  5.55079205  0.63117694 -1.59832836]
```

Result: Above program was executed successfully.

5 B. Linear Regression and Logistic Regression with the Diabetes Dataset Using Python Machine Learning

Aim

In this experiment we use the diabetes dataset from sklearn and then we need to implement the Linear Regression over this:

Procedure

1. Load sklearn Libraries.
2. Load Data
3. Load the diabetes dataset
4. Split Dataset
5. Creating Model Linear Regression and Logistic Regression
6. Make predictions using the testing set
7. Finding Coefficient And Mean Square Error

Program

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

#To calculate accuracy measures and confusion matrix
from sklearn import metrics

diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)

diabetes_X = diabetes_X[:, np.newaxis, 2]

# Split the data into training/testing sets
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]

# Split the targets into training/testing sets
diabetes_y_train = diabetes_y[:-20]
diabetes_y_test = diabetes_y[-20:]

# Create linear regression object
regr = linear_model.LinearRegression()
```

```
# Train the model using the training sets
regr.fit(diabetes_X_train, diabetes_y_train)


# Make predictions using the testing set
diabetes_y_pred = regr.predict(diabetes_X_test)

# Create Logistic regression object
Logistic_model = LogisticRegression()
Logistic_model.fit(diabetes_X_train, diabetes_y_train)

# The coefficients
print('Coefficients: \n', regr.coef_)
# The mean squared error
print('Mean squared error: %.2f'
      % mean_squared_error(diabetes_y_test, diabetes_y_pred))
# The coefficient of determination: 1 is perfect prediction
print('Coefficient of determination: %.2f'
      % r2_score(diabetes_y_test, diabetes_y_pred))

y_predict = Logistic_model.predict(diabetes_X_train)
#print("Y predict/hat ", y_predict)
y_predict
```

Output

```
Coefficients:
 [938.23786125]
Mean squared error: 2548.07
Coefficient of determination: 0.47
```

5 C. Use the diabetes data set from UCI and Pima Indians Diabetes data set for performing the following: Multiple Regression

Aim

Multiple regression is like linear regression, but with more than one independent value, meaning that we try to predict a value based on **two or more** variables.

Procedure

The Pandas module allows us to read csv files and return a DataFrame object.

Then make a list of the independent values and call this variable X.

Put the dependent values in a variable called y.

From the sklearn module we will use the LinearRegression() method to create a linear regression object.

This object has a method called fit() that takes the independent and dependent values as parameters and fills the regression object with data that describes the relationship.

We have a regression object that are ready to predict age values based on a person Glucose and BloodPressure

Program

```
import pandas as pd

from sklearn import linear_model

df = pd.read_csv (r'd:\\diabetes.csv')

print (df)

X = df[['Glucose', 'BloodPressure']]

y = df['Age']

regr = linear_model.LinearRegression()

regr.fit(X, y)

predictedage = regr.predict([[150, 13]])

print(predictedage)
```

Output

[28.77214401]

Result: Above program was executed successfully.

5 D. Also compare the results of the above analysis for the two data sets.

Aim

In this program, we can compare the results of the two different data sets.

Procedure

Step 1: Prepare the datasets to be compared

Step 2: Create the two DataFrames

Based on the above data, you can then create the following two DataFrames
Step 3: Compare the values between the two Pandas DataFrames

In this step, you'll need to import the NumPy package.
Let's say that you have the following data stored in a CSV file called **car1.csv**

While you have the data below stored in a second CSV file called **car2.csv**

Program

```
import pandas as pd

import numpy as np

data_1 = pd.read_csv(r'd:\car1.csv')

df1 = pd.DataFrame(data_1)

data_2 = pd.read_csv(r'd:\car2.csv')

df2 = pd.DataFrame(data_2)

df1['amount1'] = df2['amount1']

df1['prices_match'] = np.where(df1['amount'] == df2['amount1'], 'True', 'False')

df1['price_diff'] = np.where(df1['amount'] == df2['amount1'], 0, df1['amount'] -
df2['amount1'])

print(df1)
```

Output

	Model	City	Year	amount	amount1	prices_match	price_diff
0	Maruti	Chennai	2022	600000	600000	True	0
1	Hyndai	Chennai	2022	700000	700000	True	0
2	Ford	Chennai	2022	800000	850000	False	-50000
3	Kia	Chennai	2022	900000	900000	True	0
4	XL6	Chennai	2022	1000000	1000000	True	0
5	Tata	Chennai	2022	1100000	1150000	False	-50000
6	Audi	Chennai	2022	1200000	1200000	True	0
7	Ertiga	Chennai	2022	1300000	1300000	True	0

Please click here to download the Dataset

Dataset 1: car1.csv

Dataset 2: car2.csv

6 A). Apply and explore various plotting functions on UCI data sets. Density and contour plots

Aim

To apply and explore various plotting functions like Density and contour plots on datasets.

Procedure

There are three Matplotlib functions that can be helpful for this task: plt.contour for contour plots, plt.contourf for filled contour plots, and plt.imshow for showing images

A contour plot can be created with the plt.contour function. It takes three arguments: a grid of *x* values, a grid of *y* values, and a grid of *z* values.

The *x* and *y* values represent positions on the plot, and the *z* values will be represented by the contour levels.

Perhaps the most straightforward way to prepare such data is to use the np.meshgrid function, which builds two-dimensional grids from one-dimensional arrays.

Next standard line-only contour plot and for color the lines can be color-coded by specifying a colormap with the cmap argument.

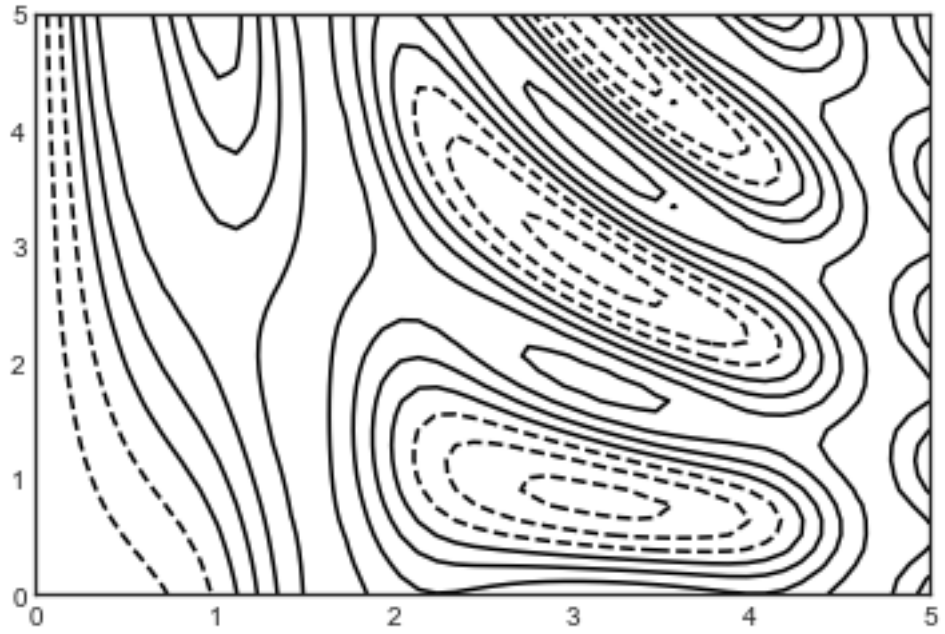
Additionally, we'll add a plt.colorbar() command, which automatically creates an additional axis with labeled color information for the plot.

Program

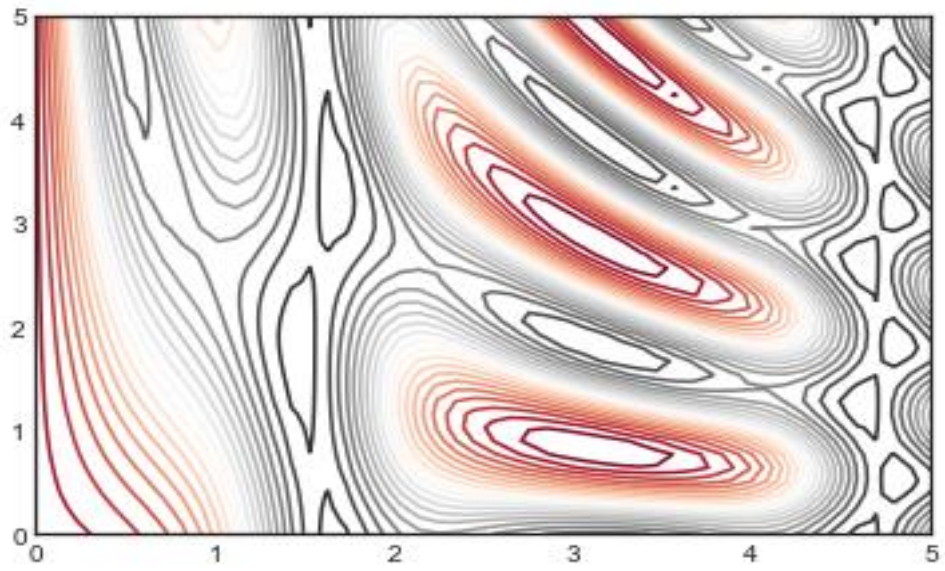
```
%matplotlib inline

import matplotlib.pyplot as plt

plt.style.use('seaborn-white')

import numpy as np

def f(x, y):

    return np.sin(x) ** 10 + np.cos(10 + y * x) * np.cos(x)

x = np.linspace(0, 5, 50)
```

```
y = np.linspace(0, 5, 40)

X, Y = np.meshgrid(x, y)

Z = f(X, Y)

plt.contour(X, Y, Z, colors='black');
```

Output

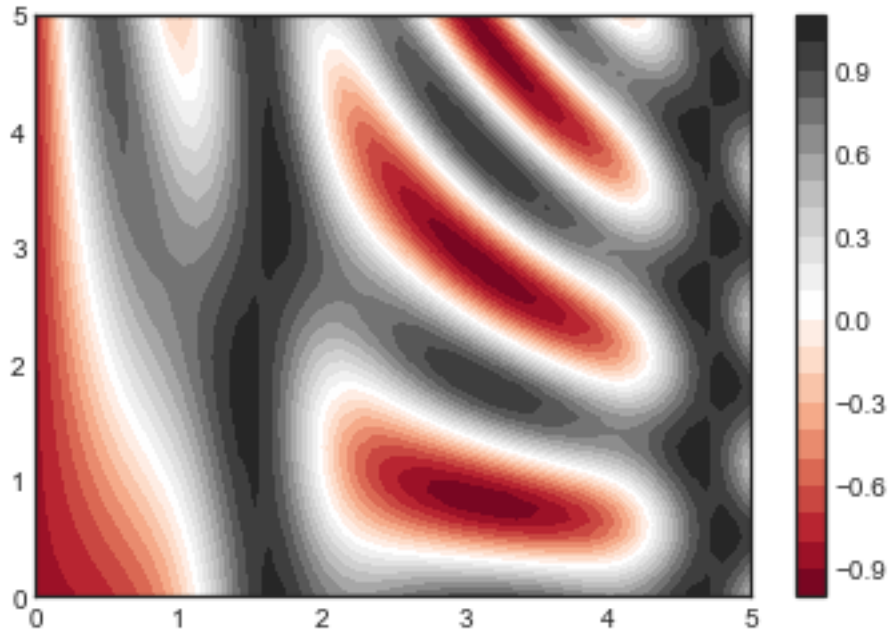



```
plt.contour(X, Y, Z, 20, cmap='RdGy');
```



```
plt.contourf(X, Y, Z, 20, cmap='RdGy')
plt.colorbar();
```

Output



6 B). Apply and explore various plotting functions like Correlation and scatter plots on UCI data sets

Aim

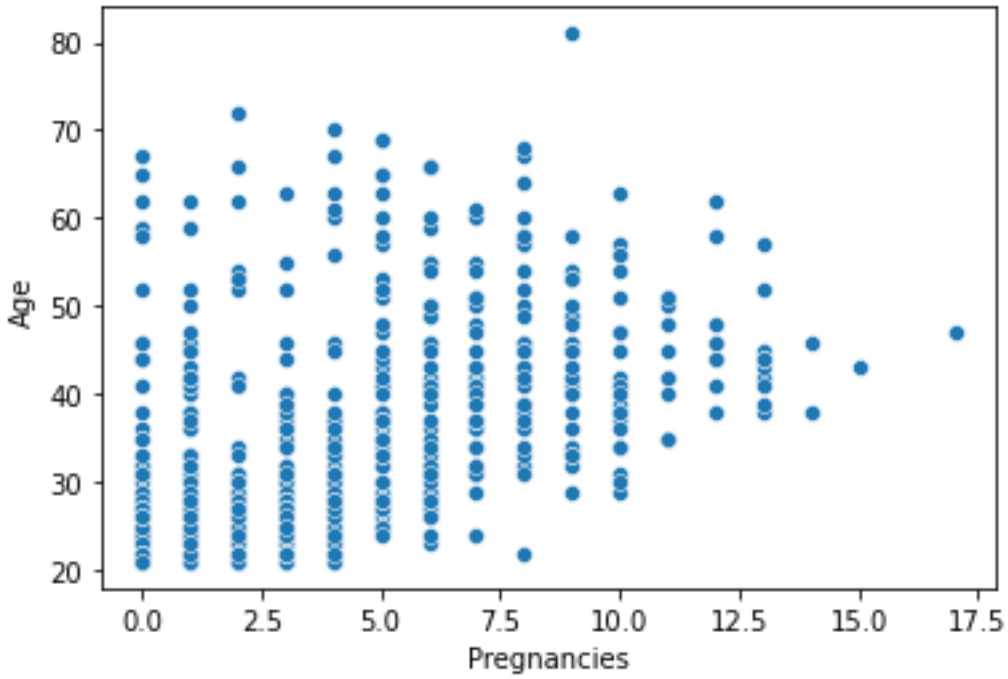
To apply and explore various plotting functions like Correlation and scatter plots on datasets.

Procedure

Program

```
import pandas as pd

con = pd.read_csv('D:/diabetes.csv')

con

list(con.columns)

import seaborn as sns

sns.scatterplot(x="Pregnancies", y="Age", data=con);
```

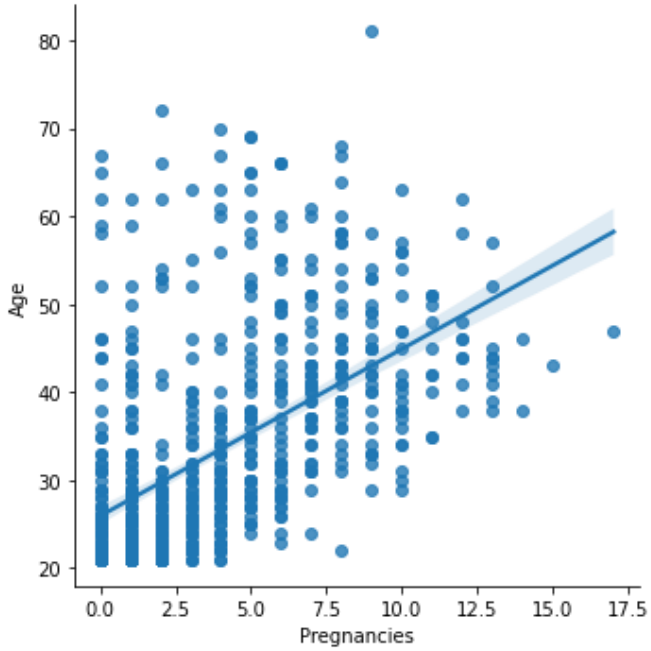
Output



Draw a scatter plot onto a FacetGrid

sns.lmplot(x="Pregnancies", y="Age", data=con);

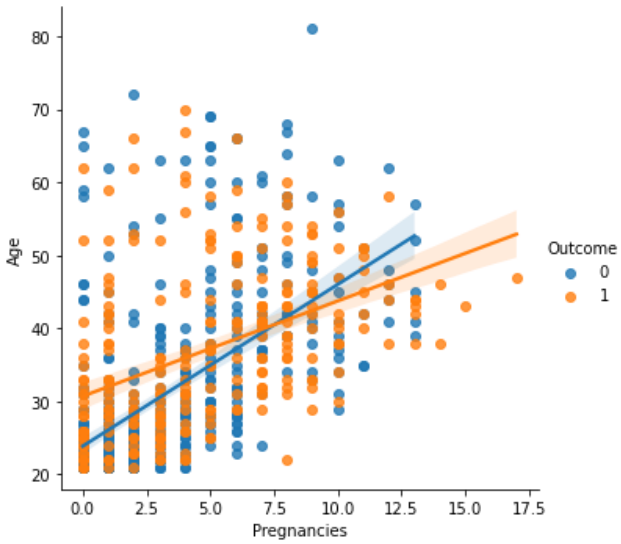
Output



draw a scatter plot onto a FacetGrid with mapping

sns.lmplot(x="Pregnancies", y="Age", hue="Outcome", data=con);

Output



To find the correlation co-efficient

```
from scipy import stats
stats.pearsonr(con['Age'], con['Outcome'])
```

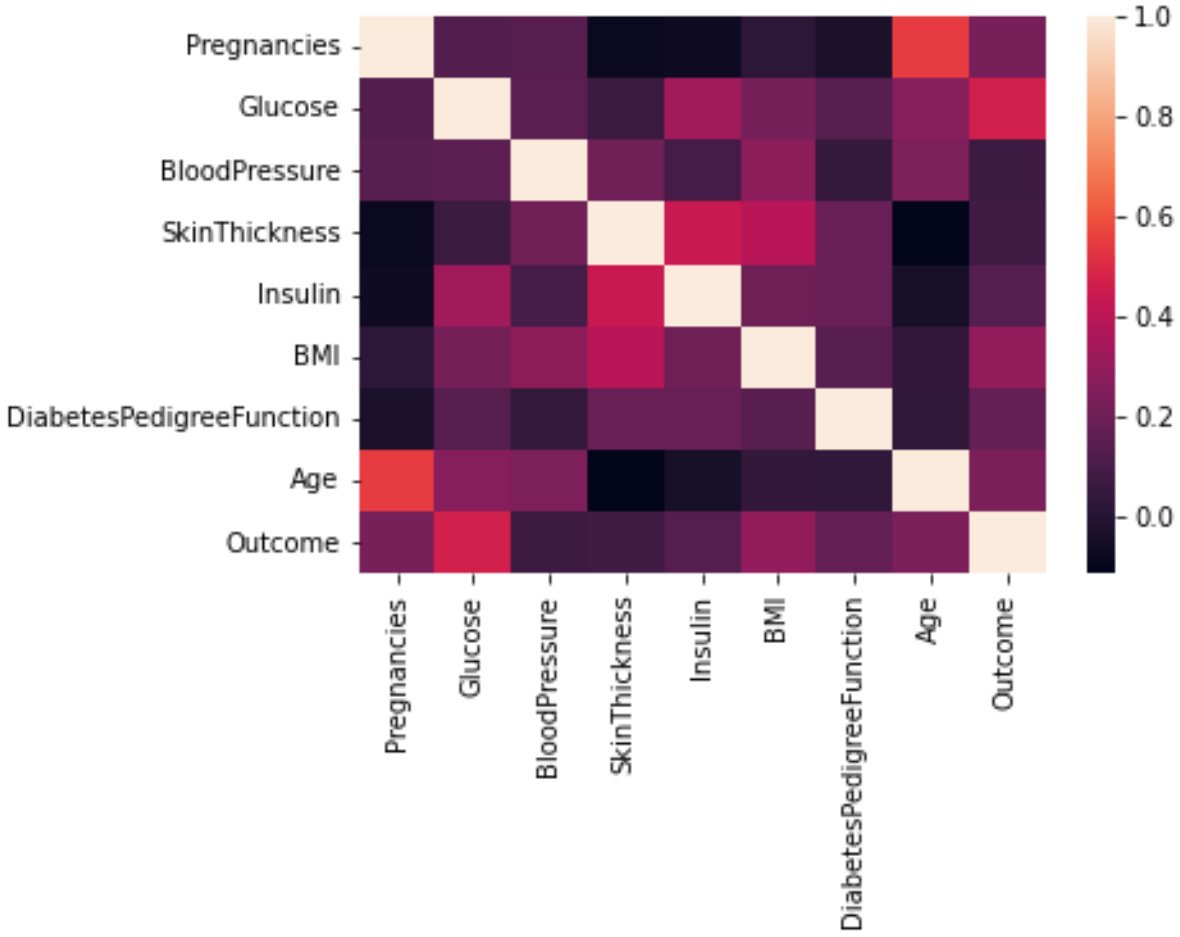
Output of Pearson coefficent

(0.23835598302719774, 2.209975460664566e-11)

To find the correlation

```
cormat = con.corr()
round(cormat,2)
sns.heatmap(cormat);
```

Output



Result: Above program was executed successfully.

6 C. Apply and explore histograms and three dimensional plotting functions on UCI data sets

Aim

To apply and explore histograms and three dimensional plotting functions on UCI data sets

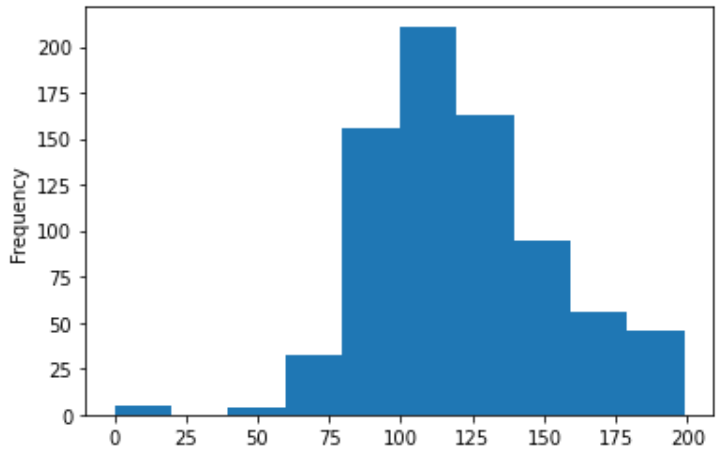
Procedure

1. Download CSV file and upload to explore.
2. A histogram is basically used to represent data provided in a form of some groups.
3. To create a histogram the first step is to create bin of the ranges, then distribute the whole range of the values into a series of intervals, and count the values which fall into each of the intervals.
4. Bins are clearly identified as consecutive, non-overlapping intervals of variables.The matplotlib.pyplot.hist() function is used to compute and create histogram of x.
5. The first one is a standard import statement for plotting using matplotlib, which you would see for 2D plotting as well.
6. The second import of the Axes3D class is required for enabling 3D projections. It is, otherwise, not used anywhere else.

Program

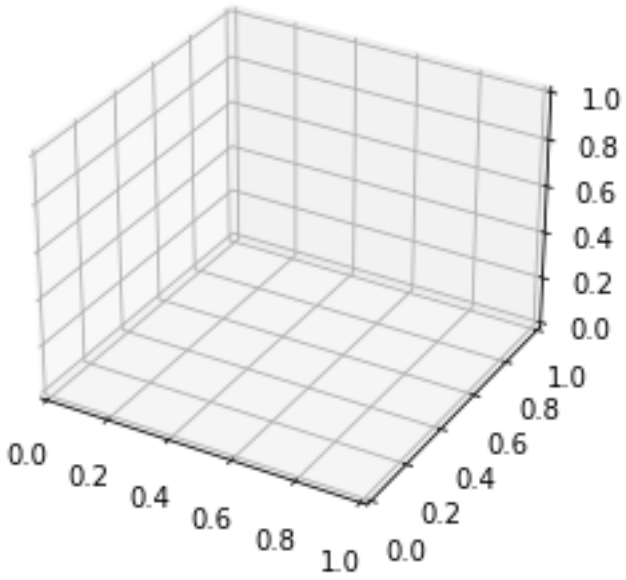
```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt  # To visualize

from mpl_toolkits.mplot3d import Axes3D

data = pd.read_csv('d:\\diabetes.csv')

data

data['Glucose'].plot(kind='hist')
```

Output




```
fig = plt.figure(figsize=(4,4))
ax = fig.add_subplot(111, projection='3d')
```

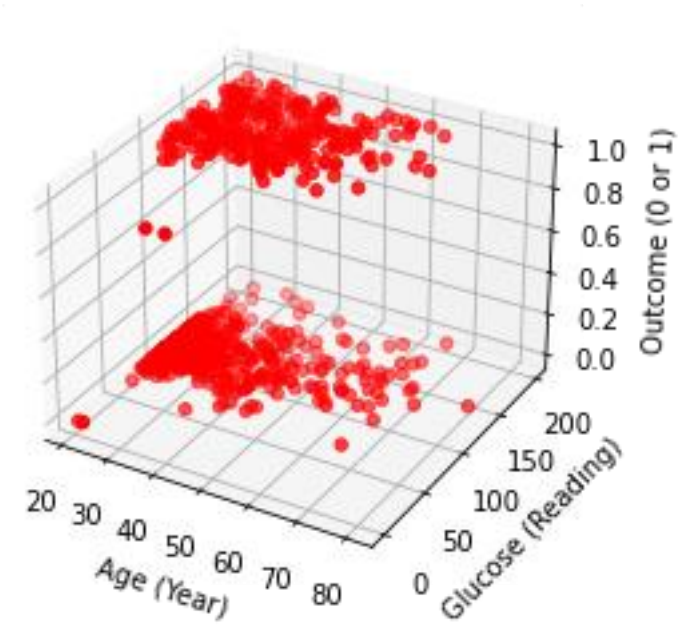
Output




```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
x = data['Age'].values
y = data['Glucose'].values
z = data['Outcome'].values
```

```
ax.set_xlabel("Age (Year)")
ax.set_ylabel("Glucose (Reading)")
ax.set_zlabel("Outcome (0 or 1)")
ax.scatter(x, y, z, c='r', marker='o')
plt.show()
```

Output



Result:

Above program was executed successfully.

7.Visualizing Geographic Data with Basemap

Aim

To create an insight Geographic Data with Basemap

Procedure

1.Install basemap is straightforward if your using conda you can type this and the package will be downloaded in Jupyter notebook.

$ conda install basemap

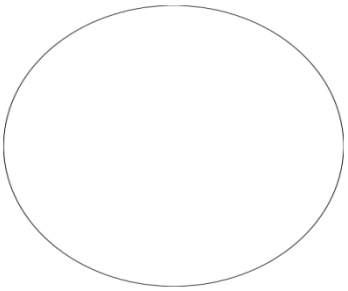
2.We add just a single new import to our standard boilerplate.

3. Once you have the Basemap toolkit installed and imported,geographic plots are just a few lines away install PIL package in Python2,or the pillow package in Python3.

Program

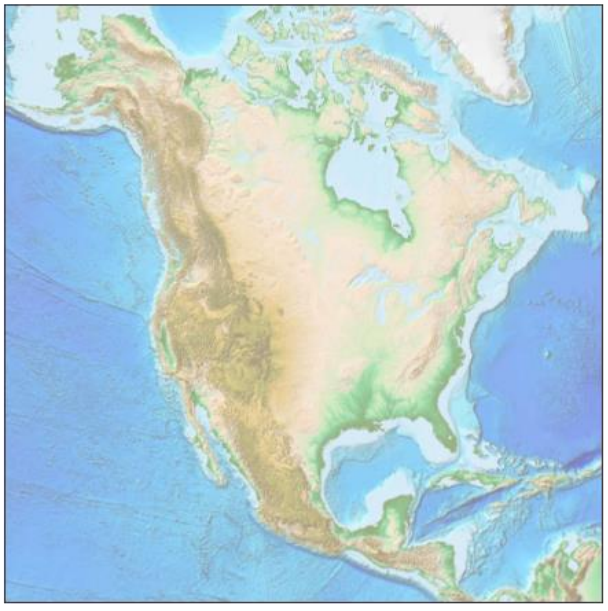
```
%matplotlib inline

import numpy as np

import matplotlib.pyplot as plt

from mpl_toolkits.basemap import Basemap

plt.figure(figsize=(8,8))  # Screen Size

m = Basemap(projection="ortho",resolution=None,lat_0=50,lon_0=-100)

m.bluemarble(scale=0.5)
```

Output



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

m = Basemap( projection="lcc", resolution=None, width=8E6,
height=8E6, lat_0= 45, lon_0= -100 )

m.etopo(scale=0.5,alpha=0.5)

x,y = m(-1223.3,47.6)

plt.plot(x,y,'ok',markersize=5)

plt.text(x,y,"Seattle",fontsize=12);
```

Output



```
from itertools import chain

def draw_map(m,scale=0.2):
```

```
    m.shadedrelief(scale=scale)
    lats = m.drawparallels(np.linspace(-90,90,13))
    lons = m.drawmeridians(np.linspace(-180,180,13))
    lat_lines = chain(*(tup[1][0] for tup in lats.items()))
    lon_lines = chain(*(tup[1][0] for tup in lons.items()))
    all_lines = chain(lat_lines,lon_lines)
    for line in all_lines:
        line.set(linestyle='-',alpha=0.3,color='w')


fig = plt.figure(figsize=(8,6),edgecolor='w')
m = Basemap( projection="cyl", resolution= None, llcrnrlat= -90,  urcrnrlat= 90, llcrnrlon= -180, urcrnrlon= 180 )
draw_map(m)
```

Output

```
fig = plt.figure(figsize=(8,6))
```

```
m =
Basemap(projection="ortho",resolution='l',lat_0=13.024767,lon_0=80.029663)

m.drawcoastlines(linewidth=0.25)

m.drawcountries(linewidth=0.25)

m.fillcontinents(color='coral',lake_color='coral')

draw_map(m);
```

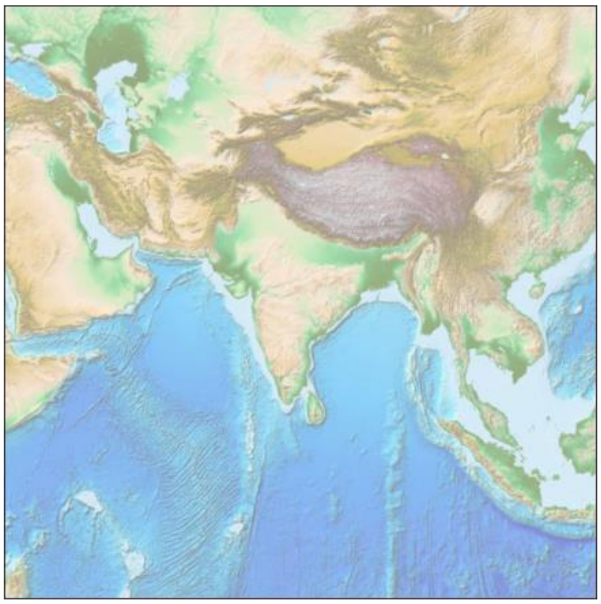
Output

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

m = Basemap(projection="lcc",resolution=None,
width=8E6,height=8E6,lat_0=45,lon_0=-100)

m.etopo(scale=0.5,alpha=0.5)

x,y = m(-122.3,47.6)

plt.plot(x,y,'ok',markersize=5)

plt.text(x,y,'Seattle',fontsize=12)
```

Output


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

m = Basemap(projection="lcc",resolution=None,
width=8E6,height=8E6,lat_0=20.593684,lon_0=78.96288) # india latitude
longitude

m.etopo(scale=0.5,alpha=0.5)

# x,y = m(13.024767,80.029663) # Dmi latitude , longitude

# plt.plot(x,y,'ok',markersize=5)

# plt.text(x,y,'DMI',fontsize=12)
```

Output



Result:

Above program was executed successfully.