Ex.No. 1 Date:	DOWNLOAD, INSTALL AND EXPLORE THE FEATURES OF NUMPY, SCIPY, JUPYTER, STATS MODELS AND PANDAS PACKAGES

How to Install Anaconda & Run Jupyter Notebook

Instructions To Install Anaconda and Run Jupyter Notebook

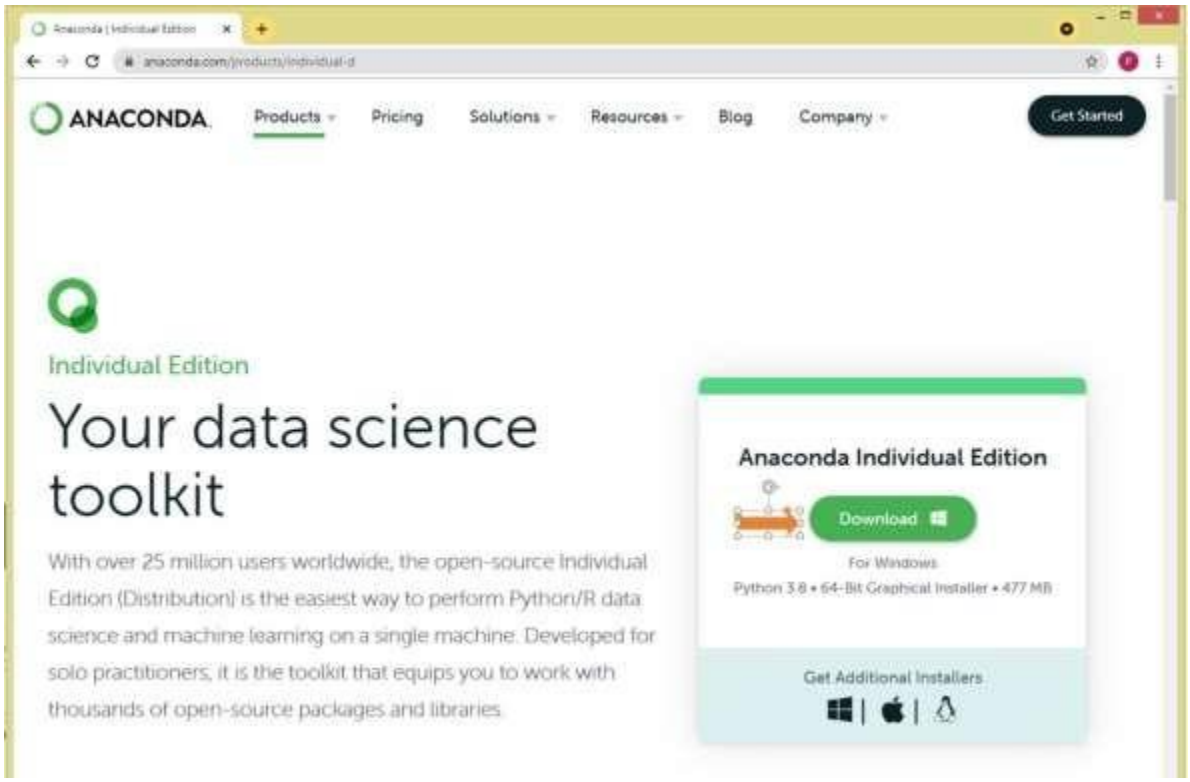
- Download & Install Anaconda Distribution
- Create Anaconda Environment
- Install and Run Jupyter Notebook

Download & Install Anaconda Distribution

Follow the below step-by-step instructions to install Anaconda distribution.

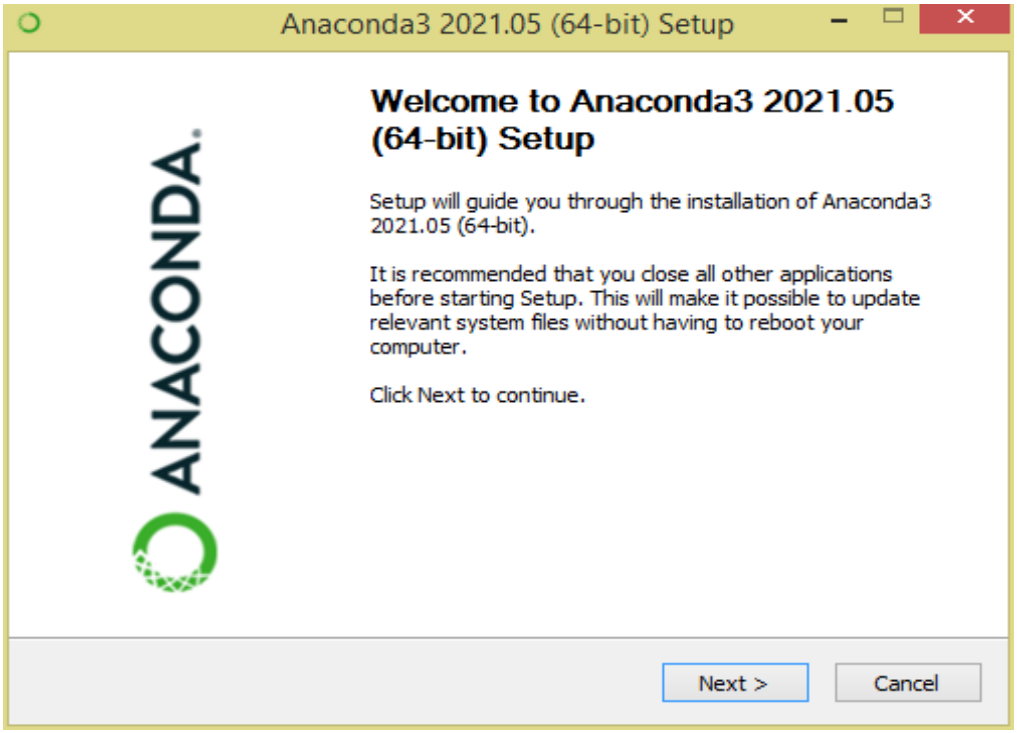
Download Anaconda Distribution

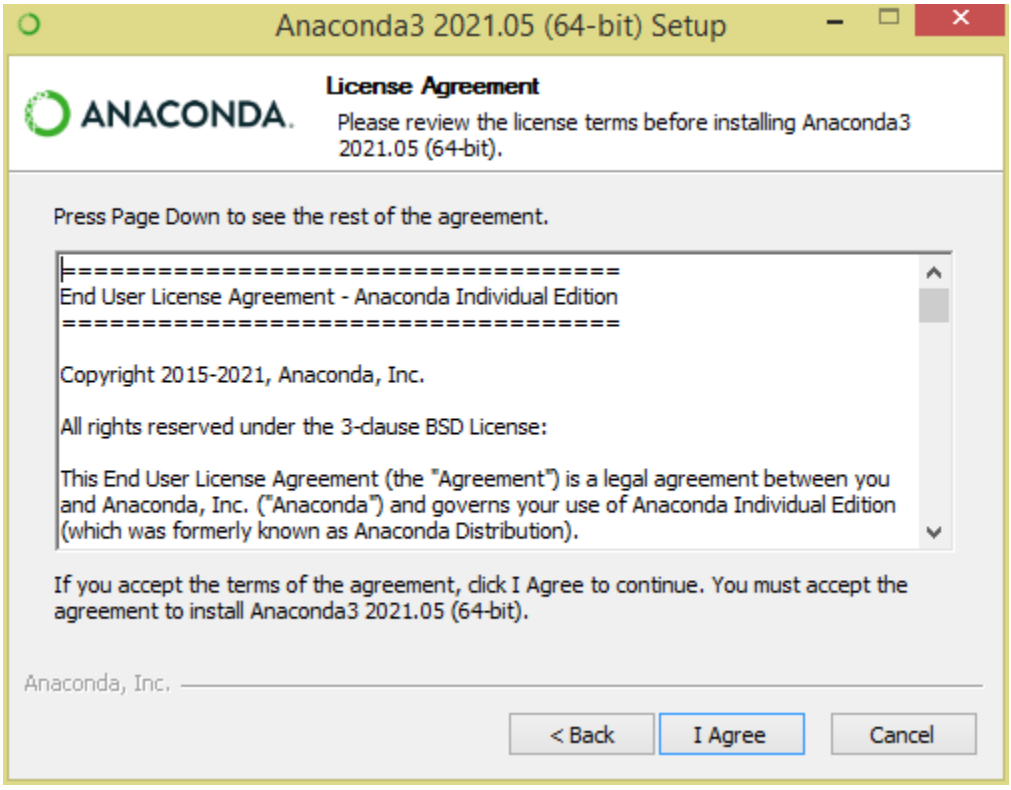
Go to https://anaconda.com/ and select **Anaconda Individual Edition** to download thelatest version of Anaconda. This downloads the .exe file to the windows download folder.

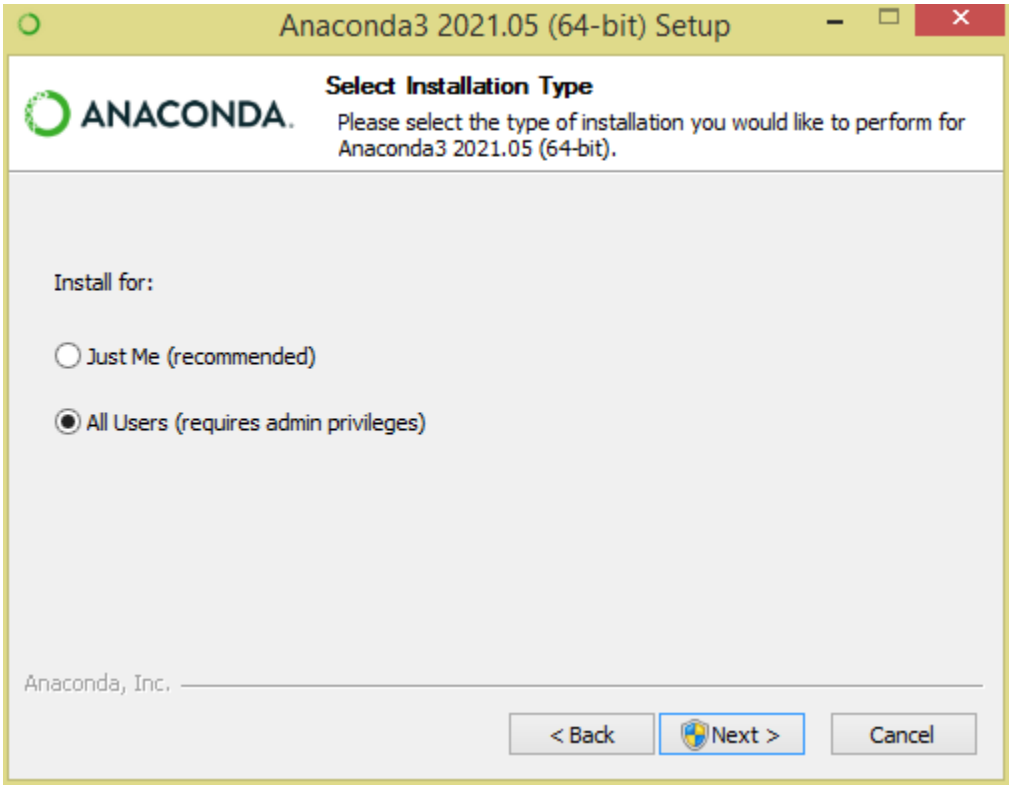


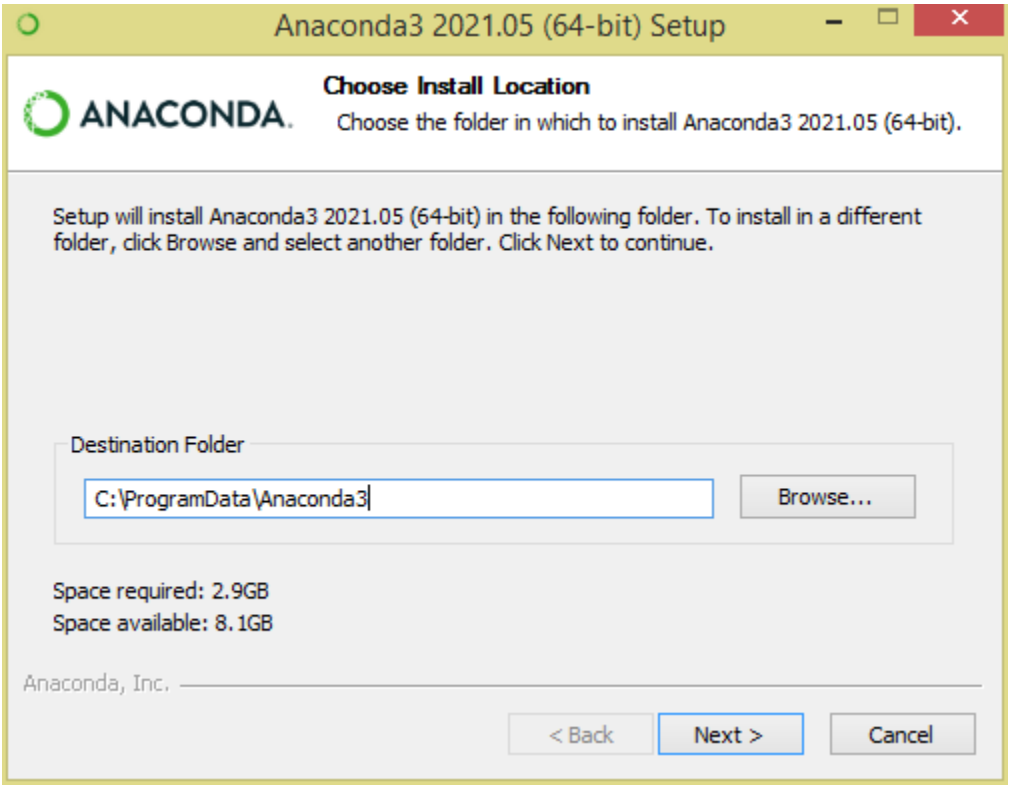
Install Anaconda

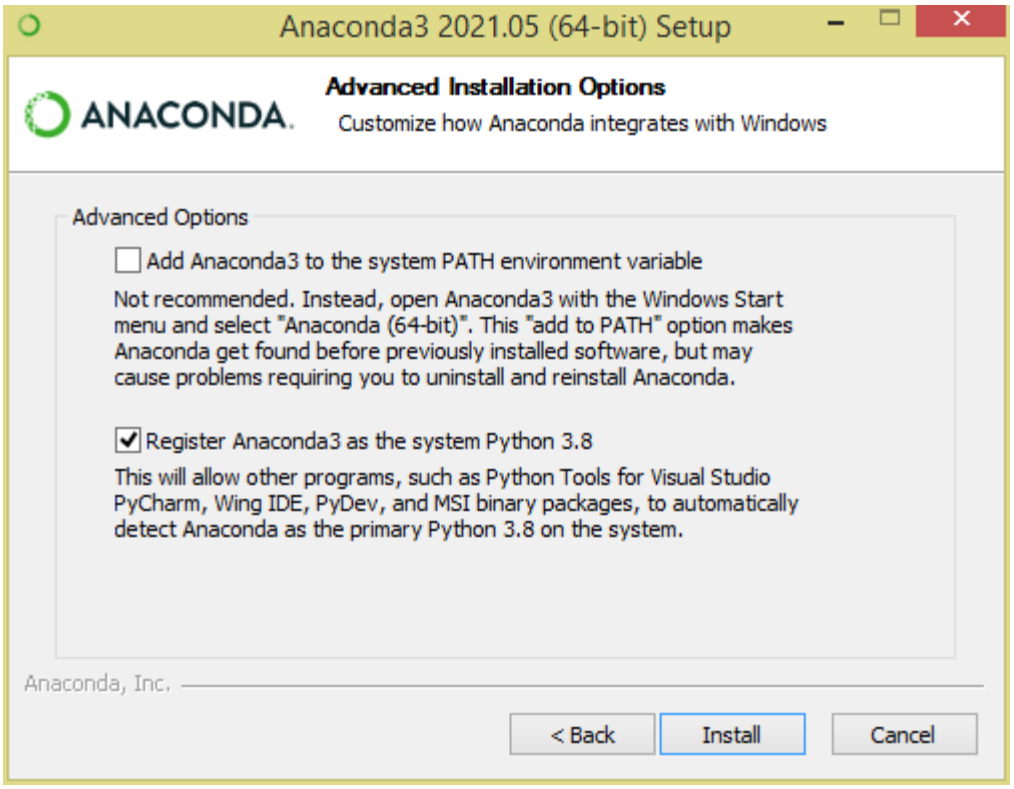
By double-clicking the .exe file starts the Anaconda installation. Follow the below screenshot's and complete the installation

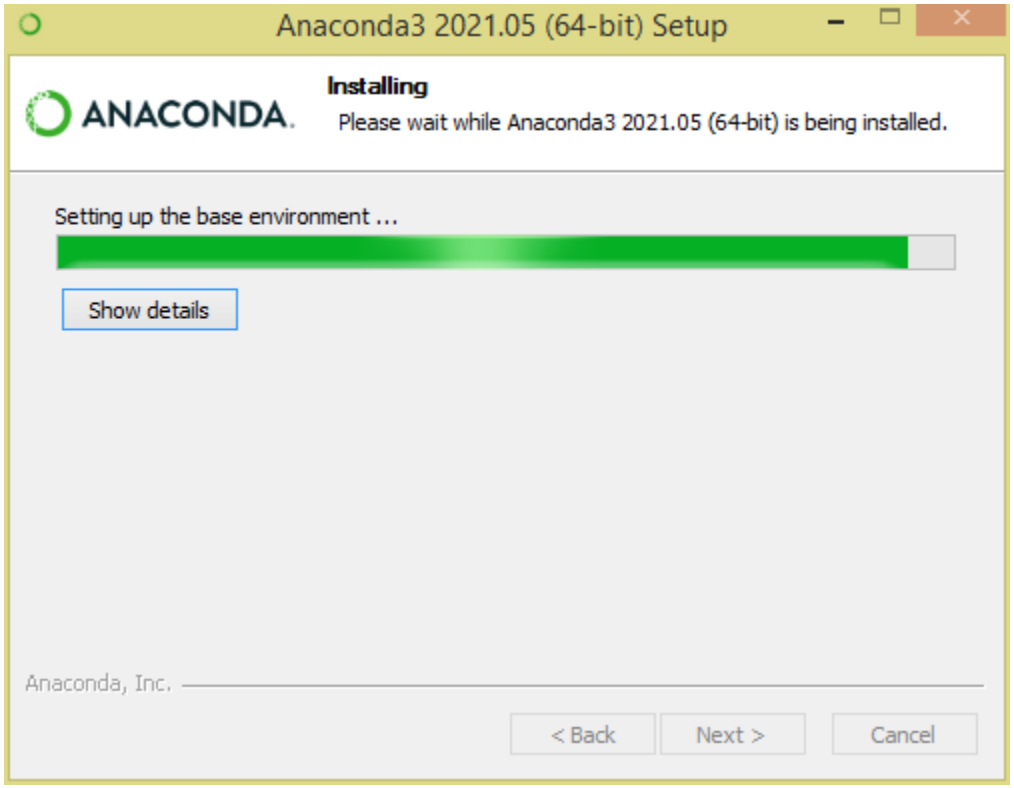


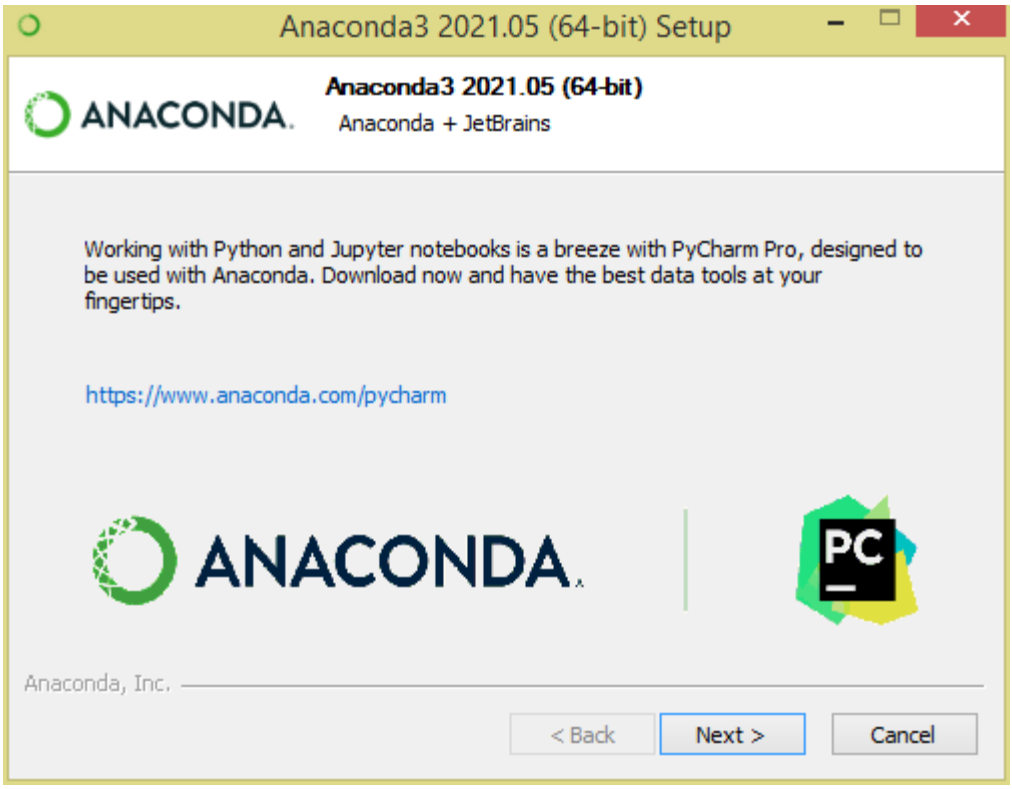


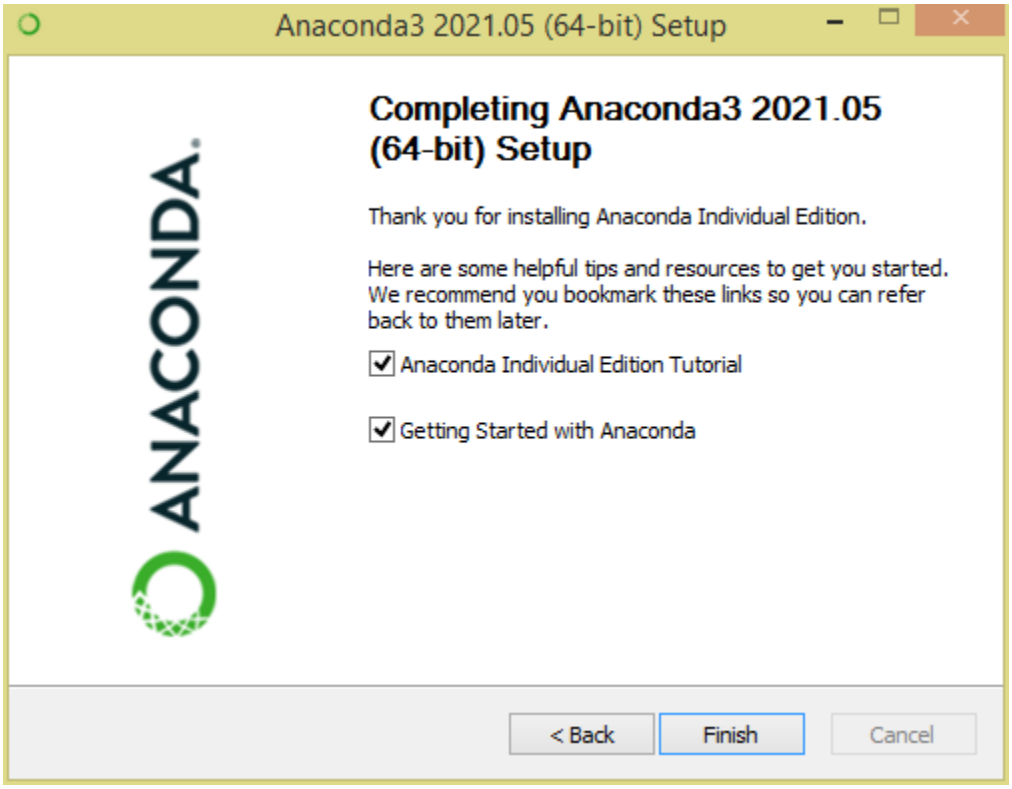

Anaconda3 2021.05 (64-bit) Setup
ANACONDA.
Select Installation Type
Please select the type of installation you would like to perform for Anaconda3 2021.05 (64-bit).
Install for:
Just Me (recommended)
All Users (requires admin privileges)
Anaconda, Inc.
< Back
Next >
Cancel


Anaconda3 2021.05 (64-bit) Setup
ANACONDA.
Choose Install Location
Choose the folder in which to install Anaconda3 2021.05 (64-bit).
Setup will install Anaconda3 2021.05 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
C:\ProgramData\Anaconda3
Browse...
Space required: 2.9GB
Space available: 8.1GB
Anaconda, Inc.
< Back
Next >
Cancel


Anaconda3 2021.05 (64-bit) Setup
ANACONDA.
Advanced Installation Options
Customize how Anaconda integrates with Windows
Advanced Options
Add Anaconda3 to the system PATH environment variable
Not recommended. Instead, open Anaconda3 with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.
Register Anaconda3 as the system Python 3.8
This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.8 on the system.
Anaconda, Inc.
< Back
Install
Cancel


Anaconda3 2021.05 (64-bit) Setup
ANACONDA.
Installing
Please wait while Anaconda3 2021.05 (64-bit) is being installed.
Setting up the base environment ...
Show details
Anaconda, Inc.
< Back
Next >
Cancel





This finishes the installation of Anaconda distribution, now let's see how to create an environment and install Jupyter Notebook.

Create Anaconda Environment from Navigator

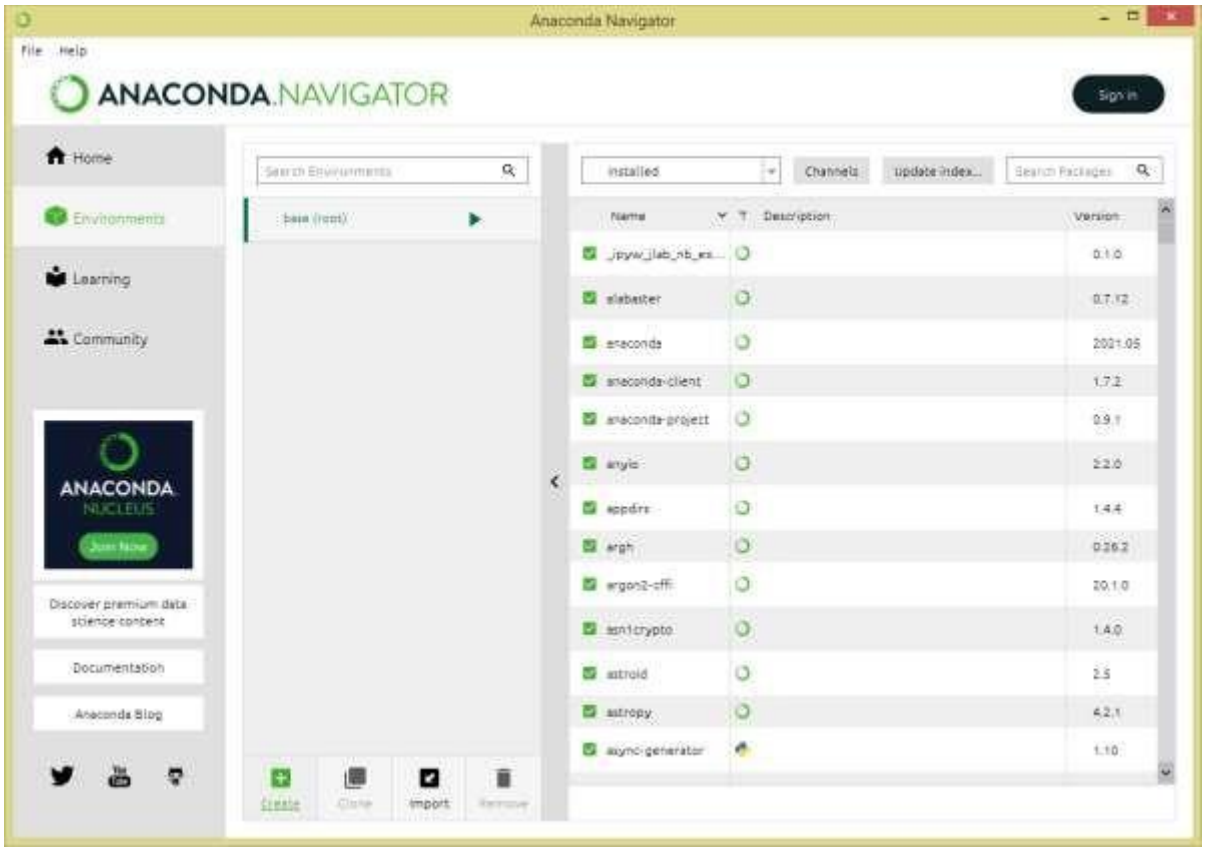
A conda environment is a directory that contains a specific collection of conda packages that you have installed. For example, you may have one environment with NumPy 1.7 and its dependencies, and another environment with NumPy 1.6 for legacy testing.
https://conda.io/docs/using/envs.html

Open Anaconda Navigator

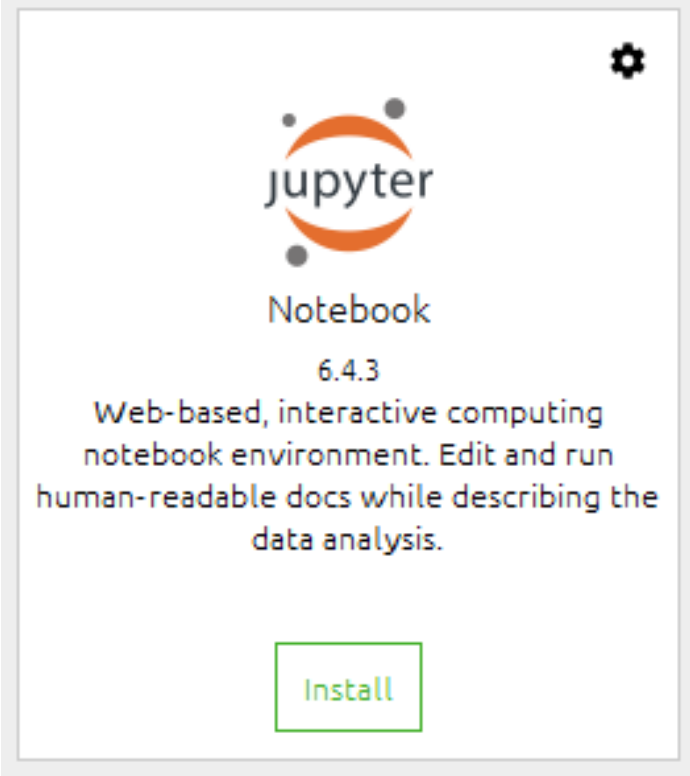
Open Anaconda Navigator from windows start or by searching it. Anaconda Navigatoris a UI application where you can control the Anaconda packages, environment etc.

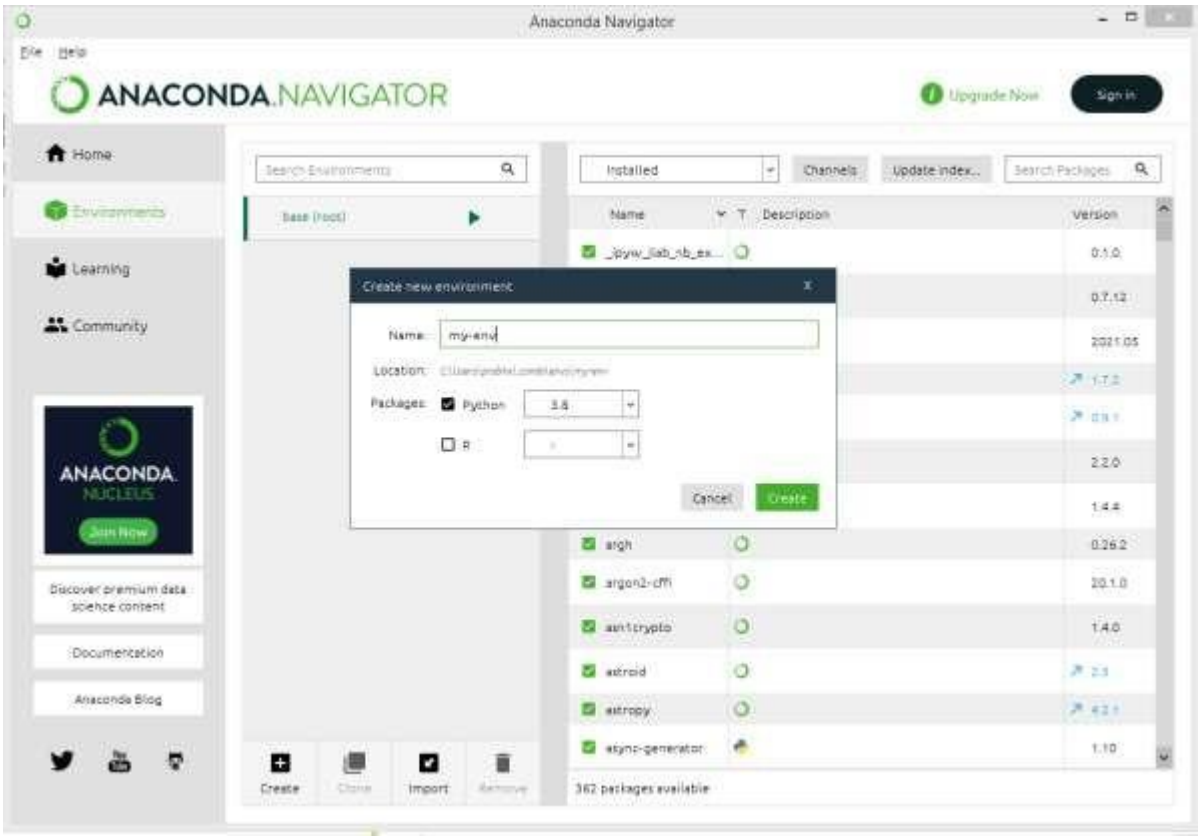
Create an Environment to Run Jupyter Notebook

This is optional but recommended to create an environment before you proceed. This gives complete segregation of different package installs for different projects you would be working on. If you already have an environment, you can use it too.



select + Create icon at the bottom of the screen to create an Anaconda environment.

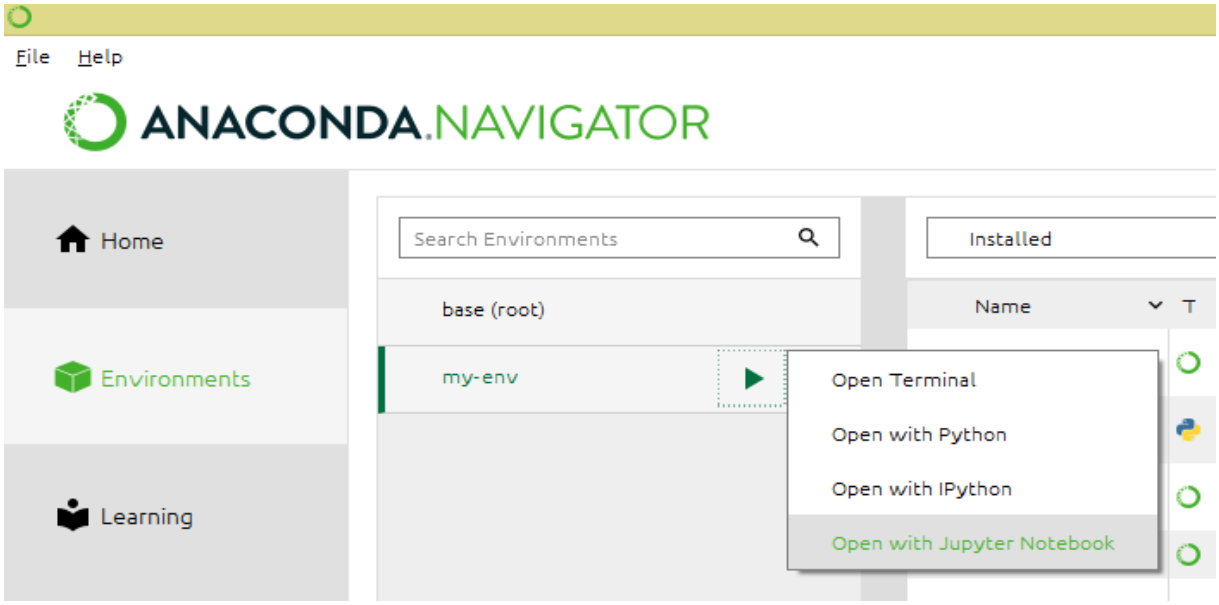




Install and Run Jupyter Notebook

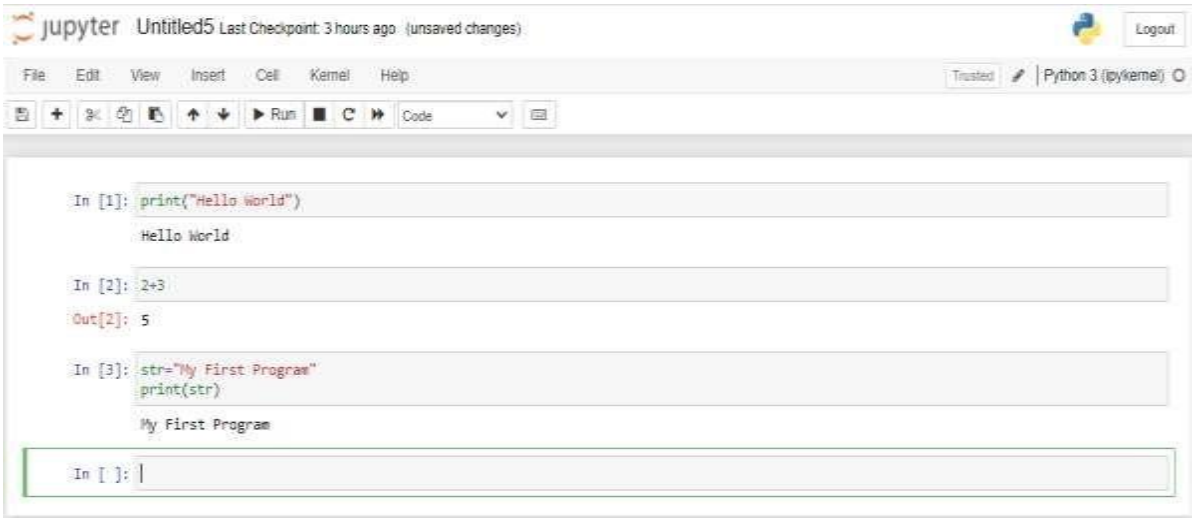
Once you create the anaconda environment, go back to the Home page on Anaconda Navigator and install Jupyter Notebook from an application on the right panel.

It will take a few seconds to install Jupyter to your environment, once the install completes, you can open Jupyter from the same screen or by accessing Anaconda Navigator -> Environments -> your environment (mine pandas-tutorial) -> select Open with JupyterNotebook.



This opens up Jupyter Notebook in the default browser.

Now select **New** -> **PythonX** and enter the below lines and select Run. On Jupyter, each cell is a statement, so you can run each cell independently when there are no dependencies on previous cells.



This completes installing Anaconda and running Jupyter Notebook.

```
C:\WINDOWS\system32>pip install statsmodels
Collecting statsmodels
  Downloading statsmodels-0.13.2-cp38-cp38-win32.whl (8.7 MB)
     ---------------------------------------- 8.7/8.7 MB 3.4 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17 in c:\program files (x86)\python38-32\lib\site-packages (from statsmodels) (1.23.2)
Requirement already satisfied: scipy>=1.3 in c:\program files (x86)\python38-32\lib\site-packages (from statsmodels) (1.9.1)
Collecting patsy>=0.5.2
  Downloading patsy-0.5.2-py2.py3-none-any.whl (233 kB)
     ---------------------------------------- 233.7/233.7 kB 3.5 MB/s eta 0:00:00
Collecting packaging>=21.3
  Downloading packaging-21.3-py3-none-any.whl (40 kB)
     ---------------------------------------- 40.8/40.8 kB 984.2 kB/s eta 0:00:00
Collecting pandas>=0.25
  Downloading pandas-1.4.3-cp38-cp38-win32.whl (9.4 MB)
     ---------------------------------------- 9.4/9.4 MB 2.3 MB/s eta 0:00:00
Collecting pyparsing!=3.0.5,>=2.0.2
  Downloading pyparsing-3.0.9-py3-none-any.whl (98 kB)
     ---------------------------------------- 98.3/98.3 kB 2.8 MB/s eta 0:00:00
Collecting pytz>=2020.1
  Downloading pytz-2022.2.1-py2.py3-none-any.whl (500 kB)
     ---------------------------------------- 500.6/500.6 kB 4.5 MB/s eta 0:00:00
Collecting python-dateutil>=2.8.1
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
     ---------------------------------------- 247.7/247.7 kB 2.2 MB/s eta 0:00:00
Collecting six
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: pytz, six, pyparsing, python-dateutil, patsy, packaging, pandas, statsmodels
Successfully installed packaging-21.3 pandas-1.4.3 patsy-0.5.2 pyparsing-3.0.9 python-dateutil-2.8.2 pytz-2022.2.1 six-1.16.0 statsmodels-0.13.2
```

```
Administrator: C:\Windows\System32\cmd.exe

C:\WINDOWS\system32>pip install numpy
Collecting numpy
  Downloading numpy-1.23.2-cp38-cp38-win32.whl (12.2 MB)
     ---------------------------------------- 12.2/12.2 MB 3.3 MB/s eta 0:00:00
Installing collected packages: numpy
Successfully installed numpy-1.23.2

[notice] A new release of pip available: 22.1.2 -> 22.2.2
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\WINDOWS\system32>pip install scipy
Collecting scipy
  Downloading scipy-1.9.1-cp38-cp38-win32.whl (34.5 MB)
     ---------------------------------------- 34.5/34.5 MB 3.0 MB/s eta 0:00:00
Requirement already satisfied: numpy<1.25.0,>=1.18.5 in c:\program files (x86)\python38-32\lib\site-packages (from scipy) (1.23.2)
Installing collected packages: scipy
Successfully installed scipy-1.9.1
```

```
C:\WINDOWS\system32>pip install pandas
Collecting pandas
  Using cached pandas-1.4.3-cp38-cp38-win32.whl (9.4 MB)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\program files (x86)\python38-32\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: numpy>=1.18.5 in c:\program files (x86)\python38-32\lib\site-packages (from pandas) (1.23.2)
Requirement already satisfied: pytz>=2020.1 in c:\program files (x86)\python38-32\lib\site-packages (from pandas) (2022.2.1)
Requirement already satisfied: six>=1.5 in c:\program files (x86)\python38-32\lib\site-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
Installing collected packages: pandas
Successfully installed pandas-1.4.3
```

```
C:\WINDOWS\system32>pip install scipy
Collecting scipy
  Downloading scipy-1.9.1-cp38-cp38-win32.whl (34.5 MB)
     ---------------------------------------- 34.5/34.5 MB 3.0 MB/s eta 0:00:00
Requirement already satisfied: numpy<1.25.0,>=1.18.5 in c:\program files (x86)\python38-32\lib\site-packages (from scipy) (1.23.2)
Installing collected packages: scipy
Successfully installed scipy-1.9.1
```

RESULT:

Thus, Jupyter Notebook environment has been successfully installed with all thenecessary packages using Anaconda distribution.

Ex.No. 2 Date:	WORKING WITH NUMPY ARRAYS

Aim

To implement array object using Numpy module in Python Coding

Algorithm

Step 1: Start the Code

Step 2: Import the required packages

Step 3: Read the elements through list/tuple/dictionary

Step 4: Convert List/tuple/dictionary into array using built-inmethods

Step 5: Check the number of dimensions in an array

Step 6: Compute the shape of an array or if it's required reshape an array

Step 7: Do the required operations like slicing, iterating, searching,concatenating and splitting an array element.

Step 8: Stop the Code

Create a NumPy ND array Object
Code

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])

print(arr)

print(type(arr))
```

Output

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

Dimensions in Arrays 0-D Arrays
Code

```
import numpy as np

arr = np.array(42)

print(arr)
```

Output

```
42
```

1-D Arrays
Code

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])

print(arr)
```

Output

```
[1 2 3 4 5]
```

2-D Arrays
Code

```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

print(arr)
```

Output

```
[[1 2 3]
[4 5 6]]
```

3-D arrays
Code

```
import numpy as np

arr = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(arr)
```

Output

```
[[[1 2 3]
[4 5 6]]
[[1 2 3]
[4 5 6]]]
```

Check Number of Dimensions

Code

```
import numpy as np

a = np.array(42)

b = np.array([1, 2, 3, 4, 5])

c = np.array([[1, 2, 3], [4, 5, 6]])

d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(a.ndim)

print(b.ndim)

print(c.ndim)

print(d.ndim)
```

Output

```
0
1
2
3
```

Access Array Elements

Code

```
import numpy as np

arr = np.array([1, 2, 3, 4])

print(arr[0])
```

Output

```
1
```

Code

```
import numpy as np

arr = np.array([1, 2, 3, 4])

print(arr[2] + arr[3])
```

Output

```
7
```

Slicing arrays
Code

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:5])
```

Output

```
[2 3 4 5]
```

NumPy Array Shape
Code

```
import numpy as np

arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])

print(arr.shape)
```

Output (2, 4)

Reshaping arrays
Code

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])

newarr = arr.reshape(4, 3)

print(newarr)
```

Output

```
[[ 1 2 3]
[ 4 5 6]
[ 7 8 9]
[10 11 12]]
```

Iterating Arrays
Code

```
import numpy as np

arr = np.array([1, 2, 3])

for x in arr:print(x)
```

Output

```
1
2
3
```

Joining NumPy Arrays
Code

```
import numpy as np
```

```
arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

arr = np.concatenate((arr1, arr2))

print(arr)
```

Output
[1 2 3 4 5 6]
,
Splitting NumPy Arrays
Code
```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6])

newarr = np.array_split(arr, 3)

print(newarr)
```

Output
[array([1, 2]), array([3, 4]), array([5, 6])]

Searching Arrays
Code
```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 4, 4])

x = np.where(arr == 4)

print(x)
```

Output
(array([3, 5, 6]),)

Sorting Arrays
Code
```
import numpy as np

arr = np.array([3, 2, 0, 1])

print(np.sort(arr))
```

Output
[0 1 2 3]

RESULT:

Thus, Array object has been explored using Numpy module in Python Coding successfully.

Ex.No. 3 Date:	WORKING WITH PANDAS DATA FRAMES

Aim:
To work with DataFrame object using Pandas module in Python Coding

Algorithm:

Step 1: Start the Code

Step 2: Import the required packages

Step 3: Create a DataFrame using built in method.

Step 4: Load data into a DataFrame object otherwise Load Files(excel/csv) into a DataFrame

Step 5: Display the rows and describe the data set using built in method.

Step 6: Display the last 5 rows of the DataFrame.

Step 7: Check the number of maximum returned rows

Step 8: Stop the Code

(i) Create a simple Pandas DataFrame:
Code

```
import pandas as pd
data = {
"calories": [420, 380, 390],
"duration": [50, 40, 45]
}
#load data into a DataFrame Object
df = pd.DataFrame(data)
print(df)
```

Output

```
calories duration
0   420      50
1   380      40
2   390      45
```

(ii) Locate Row Code
Code

```
print(df.loc[0])
```

Output

```
 calories    420
duration     50
Name: 0, dtype: int64
```

(iii) use a list of indexes:

Code
```
print(df.loc[[0, 1]])
```

Output
```
   calories  duration
0       420        50
1       380        40
```

(iv) Named Indexes
Code
```
import pandas as pd

data = {

"calories": [420, 380, 390],

"duration": [50, 40, 45]

}

df = pd.DataFrame(data, index = ["day1", "day2", "day3"])

print(df)
```

Output
```
        calories   duration
day1    420              50
day2     380             40
day3     390             45
```

(v) Locate Named Indexes
Code
```
print(df.loc["day2"])
```

Output
```
calories 380
duration 40
Name: day2, dtype: int64
```

(vi) Load Files into a DataFrame
Code
```
 import pandas as pd

 df = pd.read_csv('data.csv')

print(df)
```

Output
```
Duration Pulse Maxpulse Calories
0 60 110 130 409.1
1 60 117 145 479.0
2 60 103 135 340.0
3 45 109 175 282.4
4 45 117 148 406.0
.. ... ... ... ...
164 60 105 140 290.8
```

165 60 110 145 300.4
166 60 115 145 310.2
167 75 120 150 320.4
168 75 125 150 330.4

[169 rows x 4 columns]

(vii) Check the number of maximum returned rows:
Code

```
import pandas as pd

print(pd.options.display.max_rows)

import pandas as pd

pd.options.display.max_rows = 9999

df = pd.read_csv('data.csv')

 print(df)
```

(viii) Viewing
Code

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df.head(4))
```

Output
Duration Pulse Maxpulse Calories
0 60 110 130 409.1
1 60 117 145 479.0
2 60 103 135 340.0
3 45 109 175 282.4
4 45 117 148 406.0

(ix) Print the last 5 rows of the DataFrame:
Code

```
print(df.tail())

print(df.info())
```

Output
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 4 columns):
Column Non-Null Count Dtype
--- ------ -------------- -----
0 Duration 169 non-null int64
1 Pulse 169 non-null int64
2 Maxpulse 169 non-null int64
3 Calories 164 non-null float64
dtypes: float64(1), int64(3)
memory usage: 5.4 KB
None

RESULT:

Thus, Data Frame object using Pandas module in Python Coding has beensuccessfully explored.

Ex.No. 4 Date:	READING DATA FROM TEXT FILES, EXCEL AND THE WEB AND EXPLORING VARIOUSCOMMANDS FOR DOING DESCRIPTIVE ANALYTICS ON THE IRIS DATA SET

Aim:

To perform descriptive analytics on Iris dataset using Python Coding

Algorithm:

Step 1: Start the Code

Step 2: Import the required packages

Step 3: Load Files (excel/csv/ text) into a DataFrame from Iris data set

Step 4: Display the rows and describe the data set using built in methods

Step 5: Compare Petal Length and Petal Width

Step 6: Visualize the data set using histogram with distplot, heatmaps box plots methods

Step 7: Check Missing Values, Duplicates and remove outliers

Step 8: Stop the Code

Code

```
import pandas as pd
# Reading the CSV file
 df = pd.read_csv("Iris.csv")
 # Printing top 5 rows
df.head()
```

Output:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

Getting Information about the Dataset

```
df.shape
```

Output:

(150, 6)

```
df.info ()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

df.describe()

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

Checking Missing Values
df.isnull().sum()

```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

Checking Duplicates
data = df.drop_duplicates(subset="Species")

Output:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
50	51	7.0	3.2	4.7	1.4	Iris-versicolor
100	101	6.3	3.3	6.0	2.5	Iris-virginica

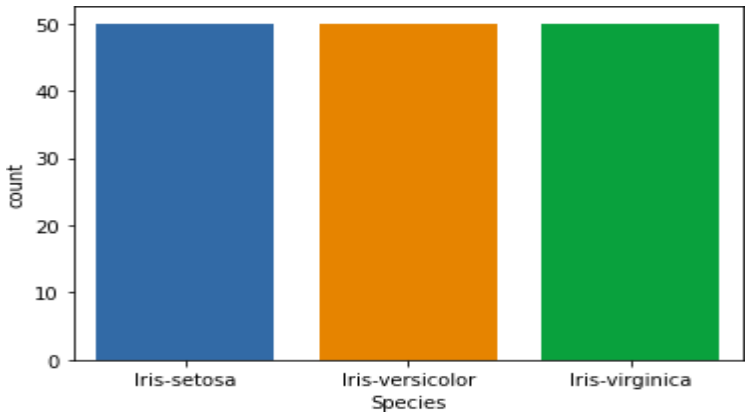
df.value_counts("Species")

```
Species
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

Data Visualization

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(x='Species',data=df)
 plt.show()
```

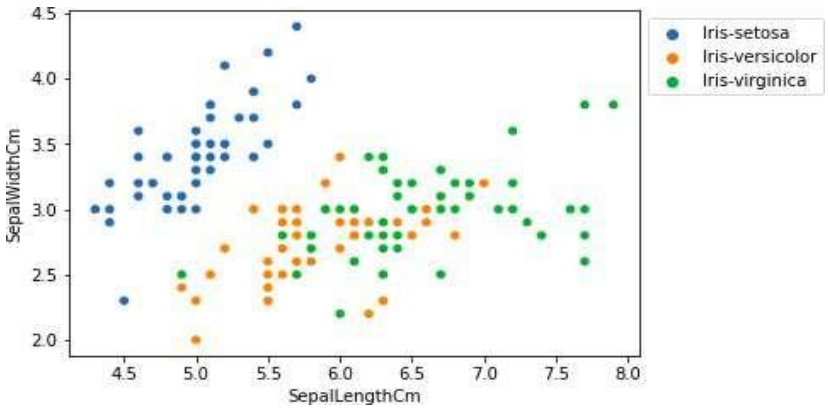
Output:



Comparing Sepal Length and Sepal Width

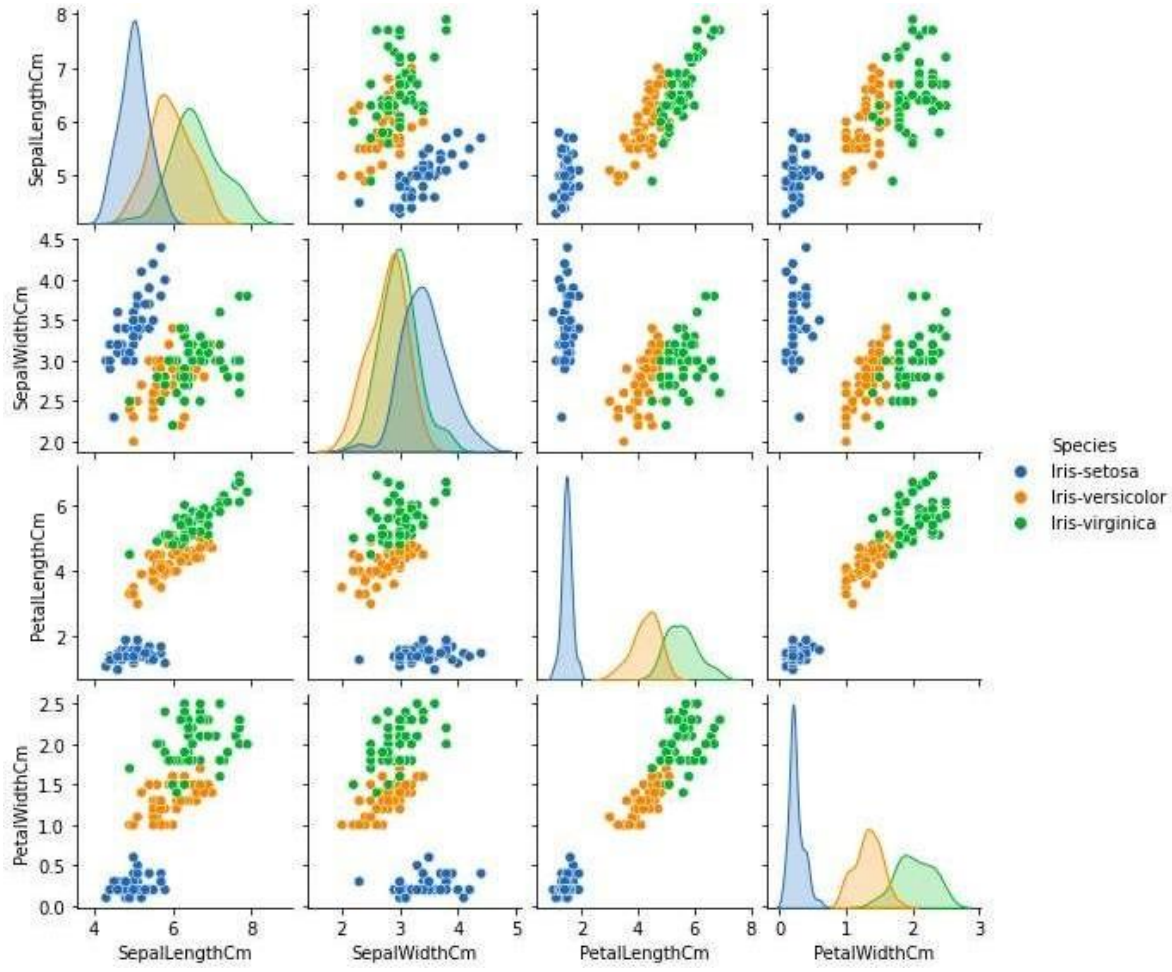
```
 # importing packages
import seaborn as sns
import matplotlib.pyplot as plt
sns.scatterplot(x='SepalLengthCm',y='SepalWidthCm', hue='Species',data=df, )
# Placing Legend outside the Figure
plt.legend(bbox_to_anchor=(1, 1), loc=2)
plt.show()
```

Output:



```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
sns.pairplot(df.drop(['Id'], axis =1), hue='Species', height=2)
```

Output:

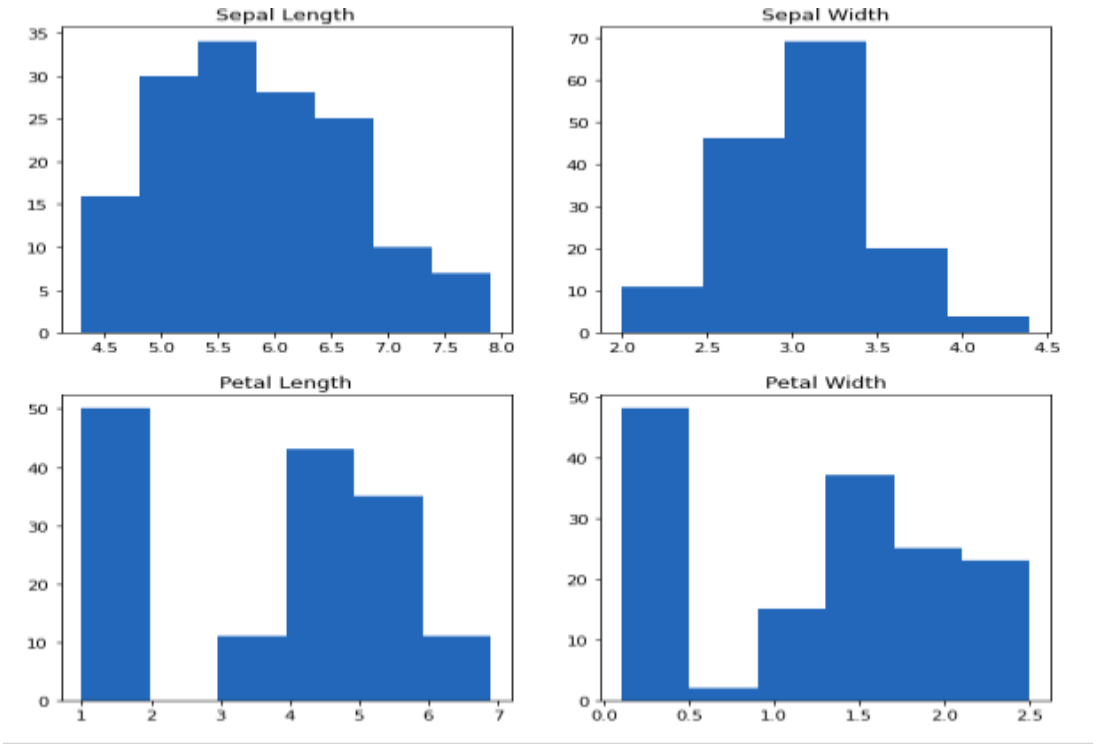


Histogram

Code

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
fig, axes = plt.subplots(2, 2, figsize=(10,10))
axes[0,0].set_title("Sepal Length")
axes[0,0].hist(df['SepalLengthCm'], bins=7)
axes[0,1].set_title("Sepal Width")
axes[0,1].hist(df['SepalWidthCm'], bins=5)
axes[1,0].set_title("Petal Length")
axes[1,0].hist(df['PetalLengthCm'], bins=6)
axes[1,1].set_title("Petal Width")
axes[1,1].hist(df['PetalWidthCm'], bins=6)
```

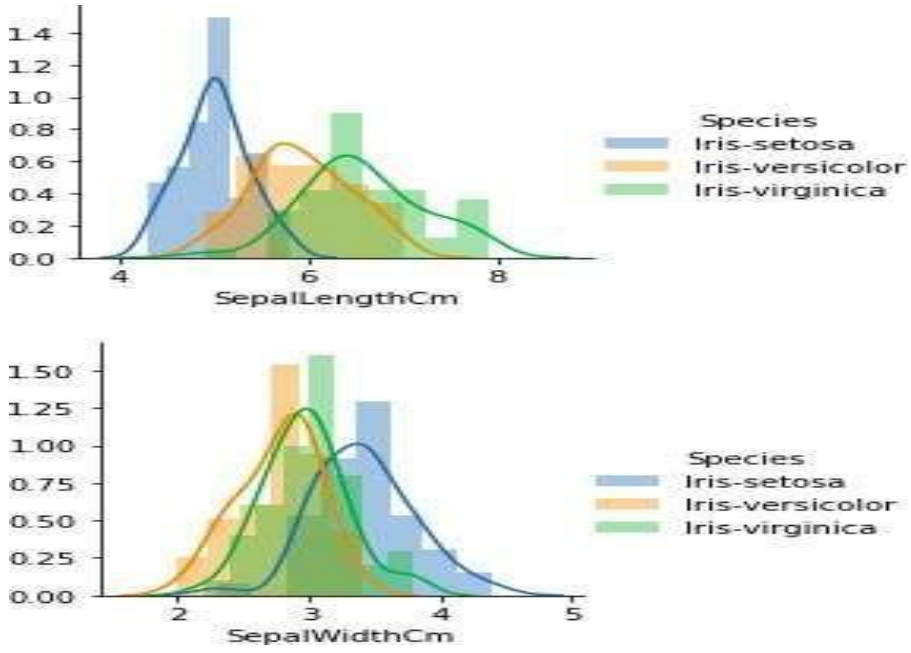
Output:

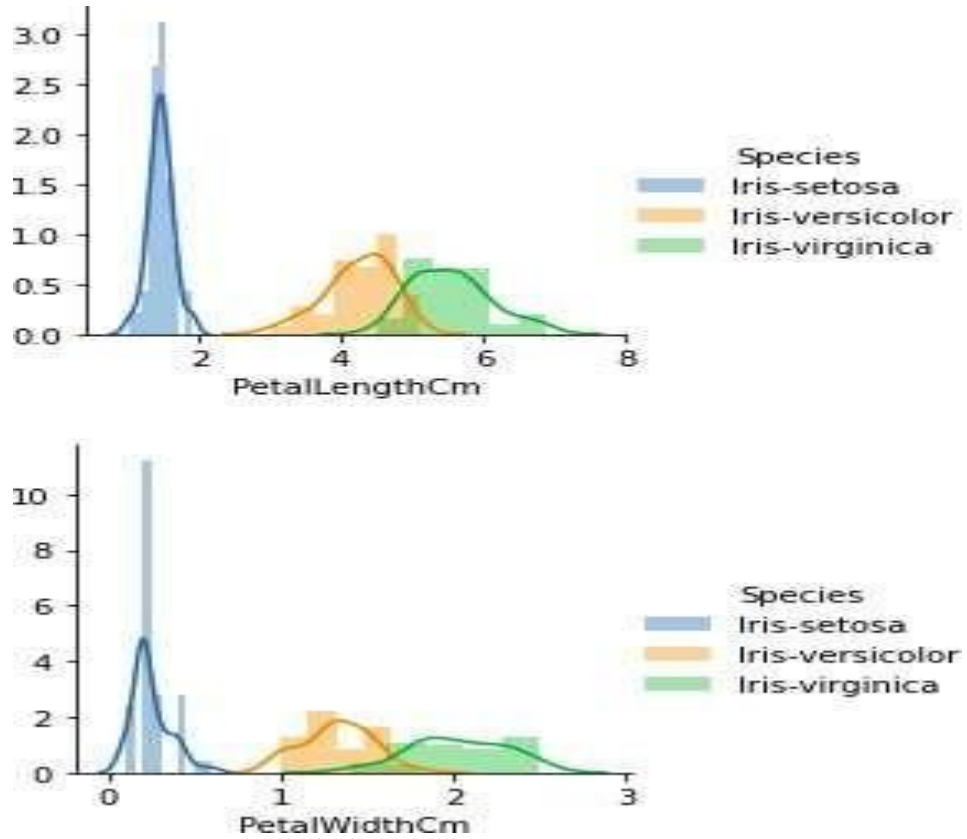


Histograms with Distplot Plot
Code

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
plot = sns.FacetGrid(df, hue="Species")
plot.map(sns.distplot, "SepalLengthCm").add_legend()
plot = sns.FacetGrid(df, hue="Species")
plot.map(sns.distplot, "SepalWidthCm").add_legend()
plot = sns.FacetGrid(df, hue="Species")
plot.map(sns.distplot, "PetalLengthCm").add_legend()
plot = sns.FacetGrid(df, hue="Species")
plot.map(sns.distplot, "PetalWidthCm").add_legend()
plt.show()
```

Output:





Handling Correlation

data.corr(method='pearson')

Output:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
Id	1.000000	0.624413	-0.654654	0.969909	0.999685
SepalLengthCm	0.624413	1.000000	-0.999226	0.795795	0.643817
SepalWidthCm	-0.654654	-0.999226	1.000000	-0.818999	-0.673417
PetalLengthCm	0.969909	0.795795	-0.818999	1.000000	0.975713
PetalWidthCm	0.999685	0.643817	-0.673417	0.975713	1.000000

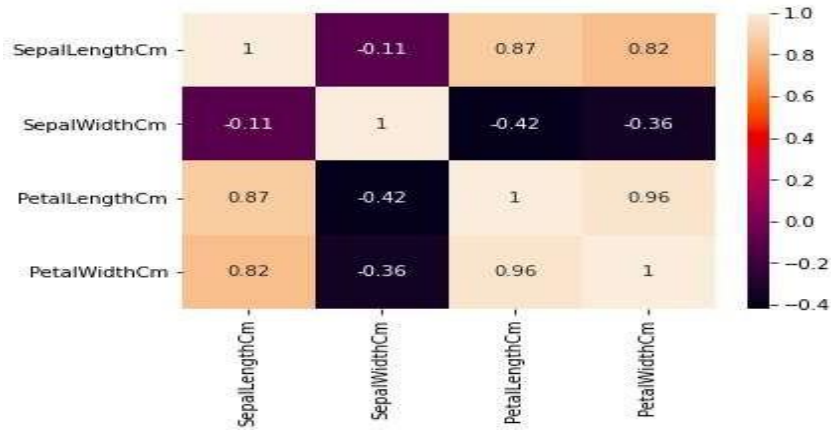
Heatmaps

Code

```
# importing packages
import seaborn as sns
```

```
import matplotlib.pyplot as plt
sns.heatmap(df.corr(method='pearson').drop( ['Id'], axis=1).drop(['Id'], axis=0),annot =True)
plt.show()
```

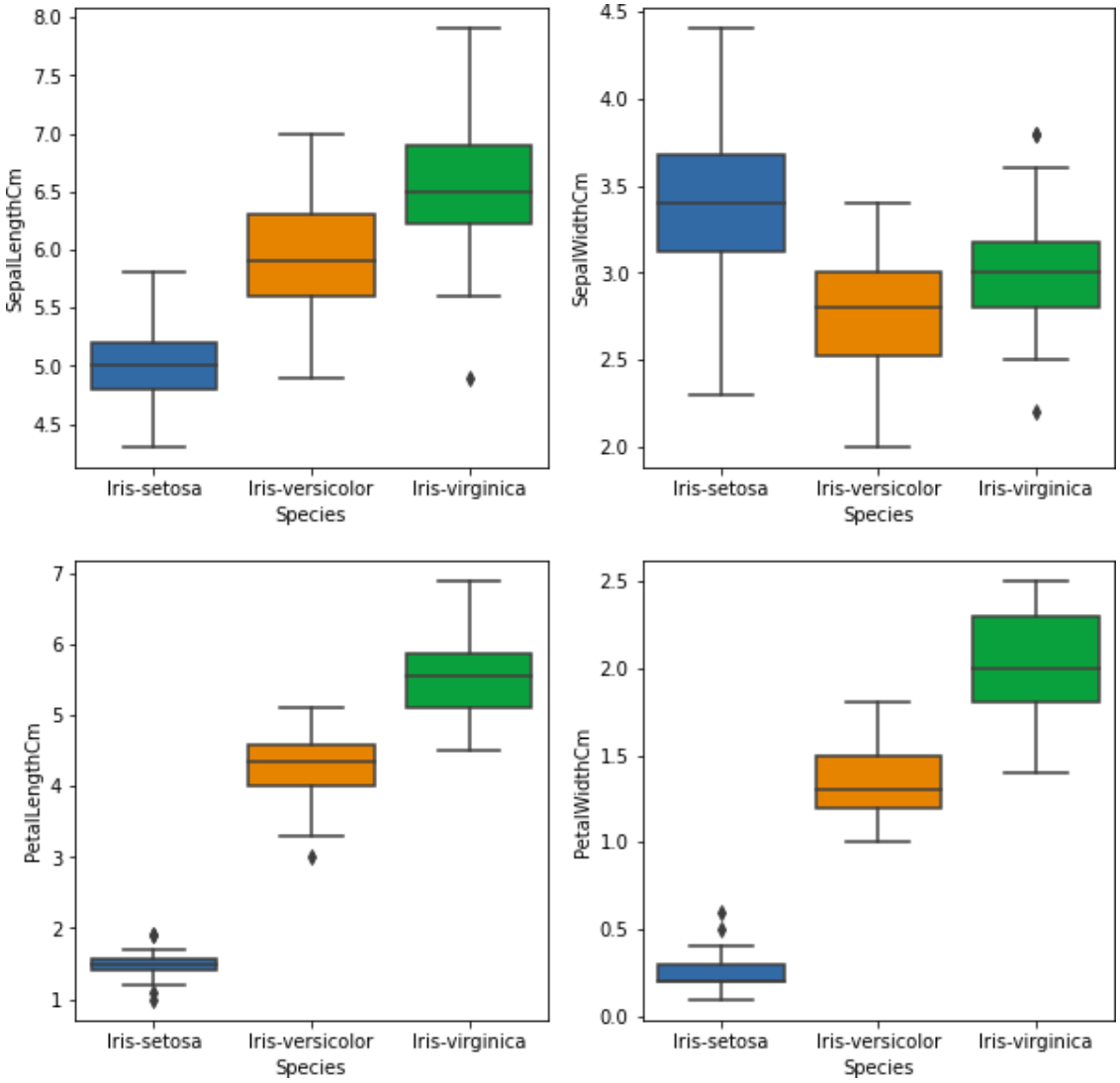
Output:



Box Plots

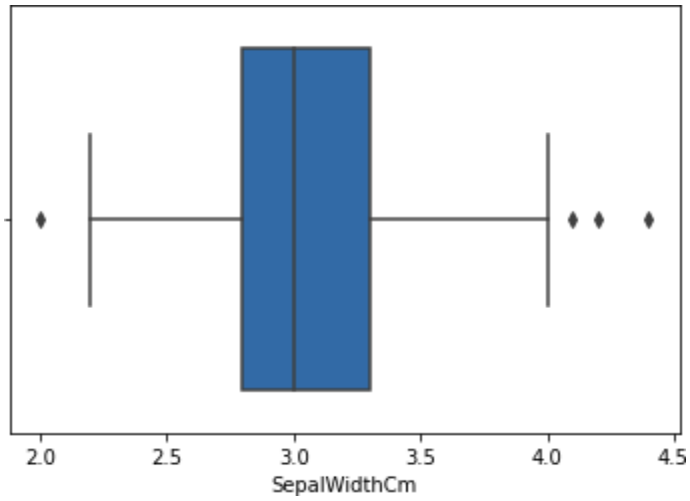
```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
def graph(y):
        sns.boxplot(x="Species", y=y, data=df)
        plt.figure(figsize=(10,10))
# Adding the subplot at the specified
# grid position
plt.subplot(221)
graph('SepalLengthCm')
plt.subplot(222)
graph('SepalWidthCm')
plt.subplot(223)
graph('PetalLengthCm')
plt.subplot(224)
graph('PetalWidthCm')
plt.show()
```

Output:




```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
# Load the dataset
df = pd.read_csv('Iris.csv')
sns.boxplot(x='SepalWidthCm', data=df)
```

Output:



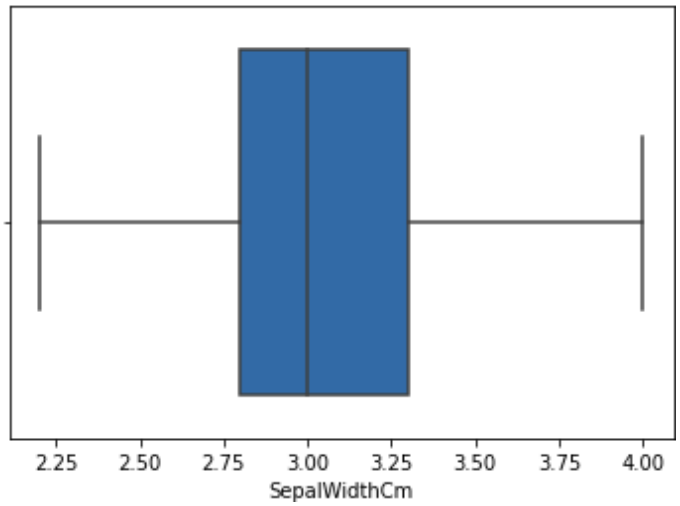
Removing OutliersCode

```
# Importing
import sklearn
import pandas as pd
import seaborn as sns
# Load the dataset
df = pd.read_csv('Iris.csv')
# IQR
Q1 = np.percentile(df['SepalWidthCm'], 25, interpolation = 'midpoint')
Q3 = np.percentile(df['SepalWidthCm'], 75, interpolation = 'midpoint')
IQR = Q3 - Q1
print("Old Shape: ",df.shape)
# Upper bound
upper = np.where(df['SepalWidthCm'] >= (Q3+1.5*IQR))
# Lower bound
lower = np.where(df['SepalWidthCm'] <= (Q1-1.5*IQR))

# Removing the Outliers
df.drop(upper[0], inplace = True)
df.drop(lower[0], inplace = True)
print("New Shape: ", df.shape)
```

```
sns.boxplot(x='SepalWidthCm', data=df)
```

Output:

```
Old Shape:  (150, 6)
New Shape:  (146, 6)

<AxesSubplot:xlabel='SepalWidthCm'>
```



RESULT:

Thus, Iris dataset has been explored and descriptively analyzed using PythonCoding

Ex.No. 5 Date:	PERFORM UNIVARIATE, BIVARIATE AND REGRESSION ANALYSIS USING THE DIABETES DATA SET FROM UCI AND PIMA INDIANS' DIABETES DATA SET

Aim:

To perform various exploratory data analysis on Pima Indians Diabetes dataset using Python Coding.

- Univariate analysis: Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewness and Kurtosis.
- Bivariate analysis: Linear and logistic regression modeling
- Multiple Regression analysis

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

Algorithm

Step 1: Start the Code

Step 2: Import the required packages

Step 3: Load Files (excel/csv/ text) into a Data Frame from UCI and Pima Indians Diabetes dataset

Step 4: Display the rows and describe the data set using built in methods

Step 5: Compute Frequency, Mean, Median, Mode, Variance, Standard Deviation, Skewnessand Kurtosis

Step 6: Visualize the data set using histogram with distplot, heatmaps, box plots methods

Step 7: Check Missing Values, Duplicates and remove outliers using built in method

Step 8: Stop the Code

Code
Import libraries for data handling, visualization, and modeling
Load the dataset
Frequency count of Glucose values

```
: import pandas as pd
  import seaborn as sns
  import matplotlib.pyplot as plt
  %matplotlib inline
  from sklearn.linear_model import LogisticRegression
  df = pd.read_csv('diabetes.csv')
  count = df['Glucose'].value_counts()
  print(count)
```

```
99     17
100    17
111    14
129    14
125    14
       ..
191     1
177     1
44      1
62      1
190     1
Name: Glucose, Length: 136, dtype: int64
```

Display first 5 rows of the dataset

```
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Descriptive statistics summary (count, mean, std, min, quartiles, max)

```
df.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Mean of each column

```
df.mean()
```

```
Pregnancies                 3.845052
Glucose                   120.894531
BloodPressure              69.105469
SkinThickness              20.536458
Insulin                    79.799479
BMI                        31.992578
DiabetesPedigreeFunction    0.471876
Age                        33.240885
Outcome                     0.348958
dtype: float64
```

Mode of each column

```
df.mode()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	1.0	99	70.0	0.0	0.0	32.0	0.254	22.0	0.0
1	NaN	100	NaN	NaN	NaN	NaN	0.258	NaN	NaN

Variance of each column

```
df.var()
```

```
Pregnancies                 11.354056
Glucose                   1022.248314
BloodPressure              374.647271
SkinThickness              254.473245
Insulin                  13281.180078
BMI                         62.159984
DiabetesPedigreeFunction     0.109779
Age                        138.303046
Outcome                      0.227483
dtype: float64
```

Standard Deviation of each column

```
df.std()
```

```
Pregnancies                 3.369578
Glucose                    31.972618
BloodPressure              19.355807
SkinThickness              15.952218
Insulin                   115.244002
BMI                         7.884160
DiabetesPedigreeFunction    0.331329
Age                        11.760232
Outcome                     0.476951
dtype: float64
```

Skewness of each column

```
df.skew()
```

```
Pregnancies                 0.901674
Glucose                     0.173754
BloodPressure              -1.843608
SkinThickness               0.109372
Insulin                     2.272251
BMI                        -0.428982
DiabetesPedigreeFunction    1.919911
Age                         1.129597
Outcome                     0.635017
dtype: float64
```

Kurtosis of each column

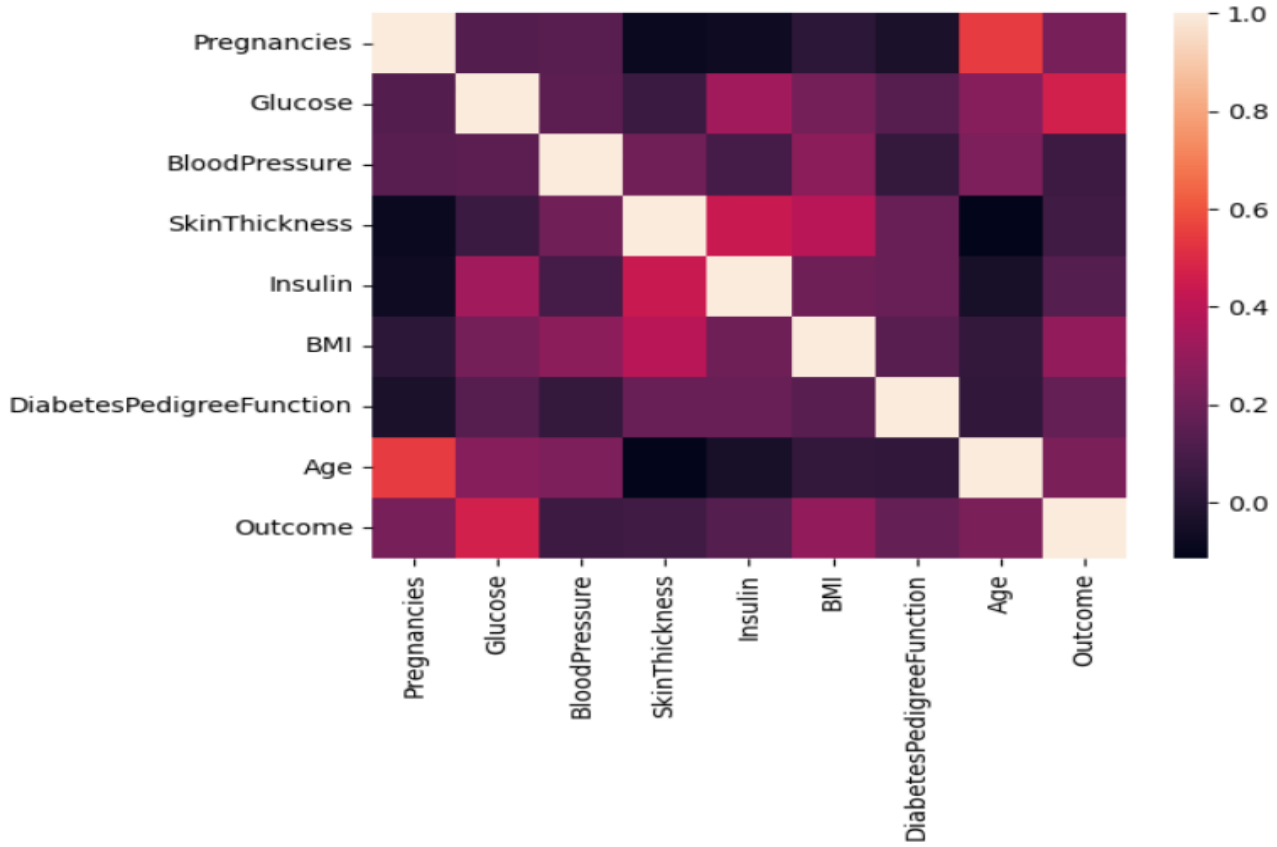
```
df.kurtosis()
```

```
Pregnancies                 0.159220
Glucose                     0.640780
BloodPressure               5.180157
SkinThickness              -0.520072
Insulin                     7.214260
BMI                         3.290443
DiabetesPedigreeFunction    5.594954
Age                         0.643159
Outcome                    -1.600930
dtype: float64
```

Correlation matrix and Heat Map of Correlations

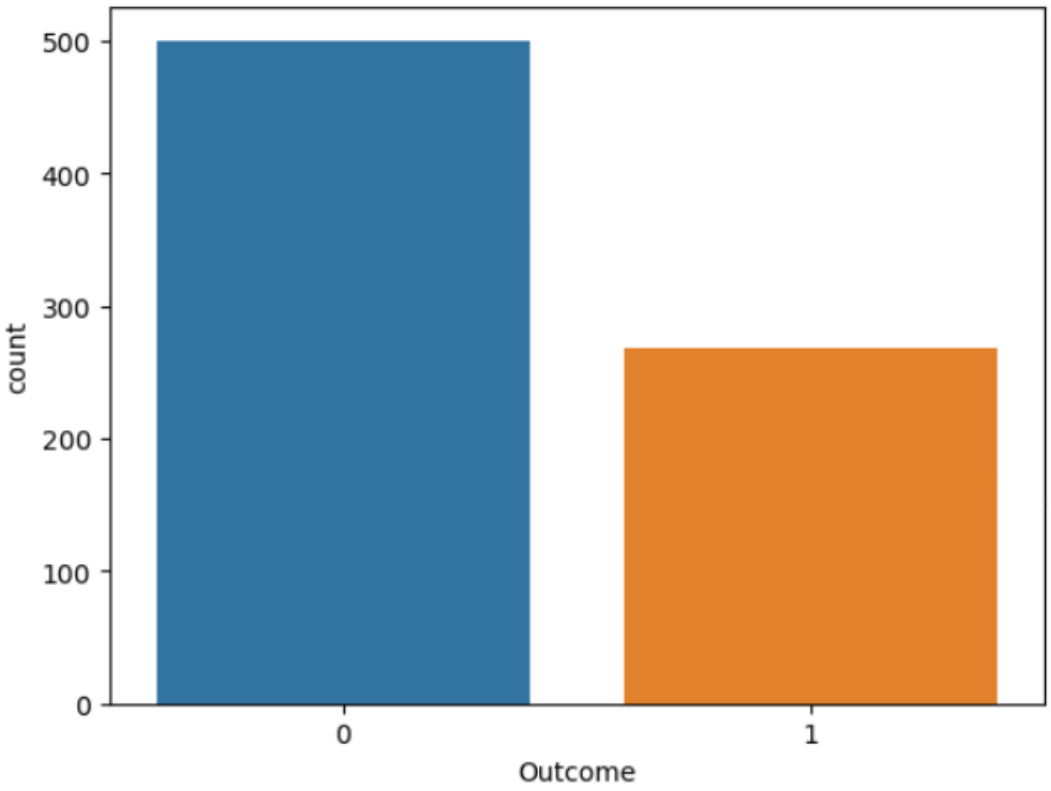
```
corr = df.corr()
sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns)
```

```
<AxesSubplot:>
```



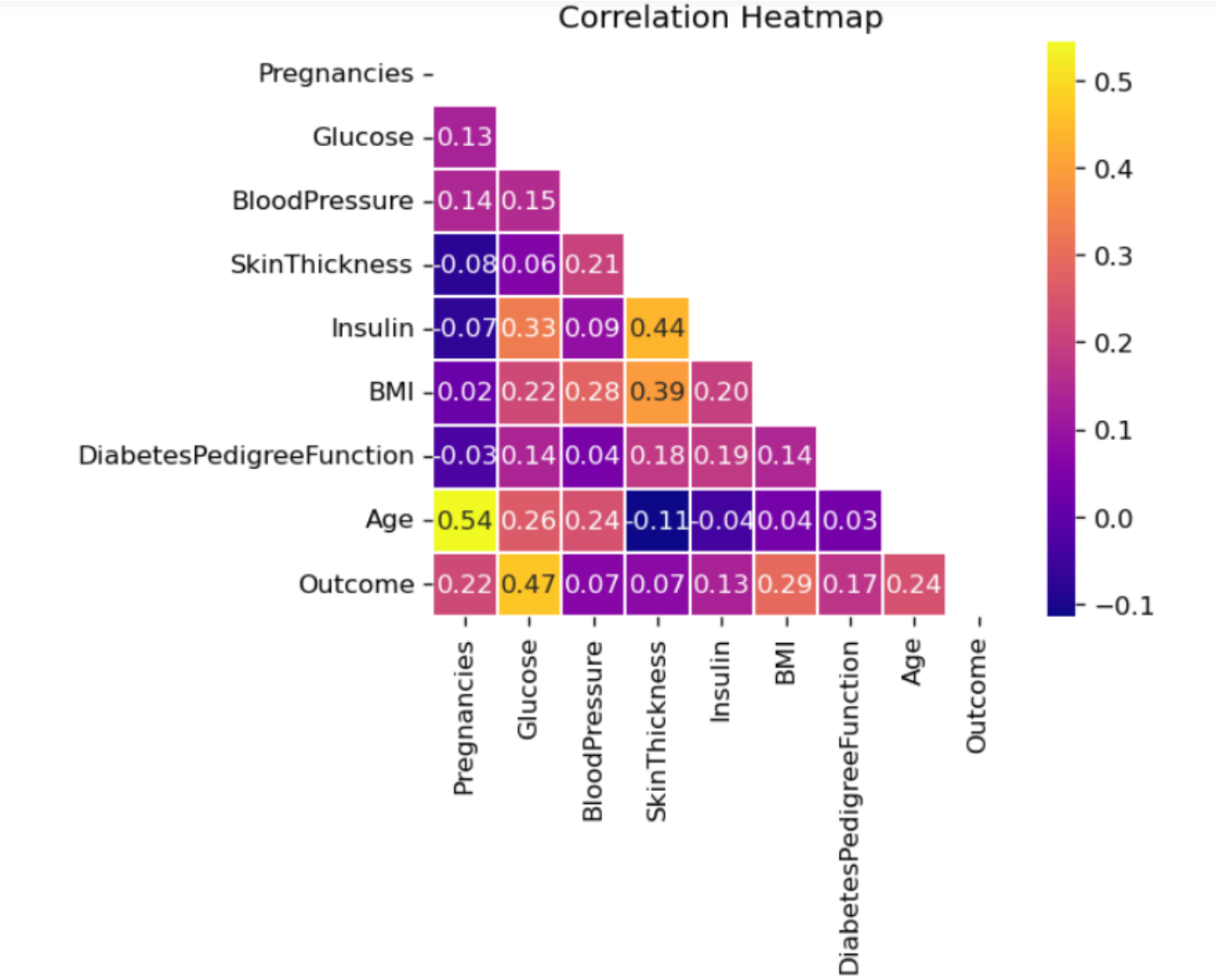
Count plot of diabetic vs non-diabetic outcomes

```
sns.countplot('Outcome', data=df)
plt.show()
```



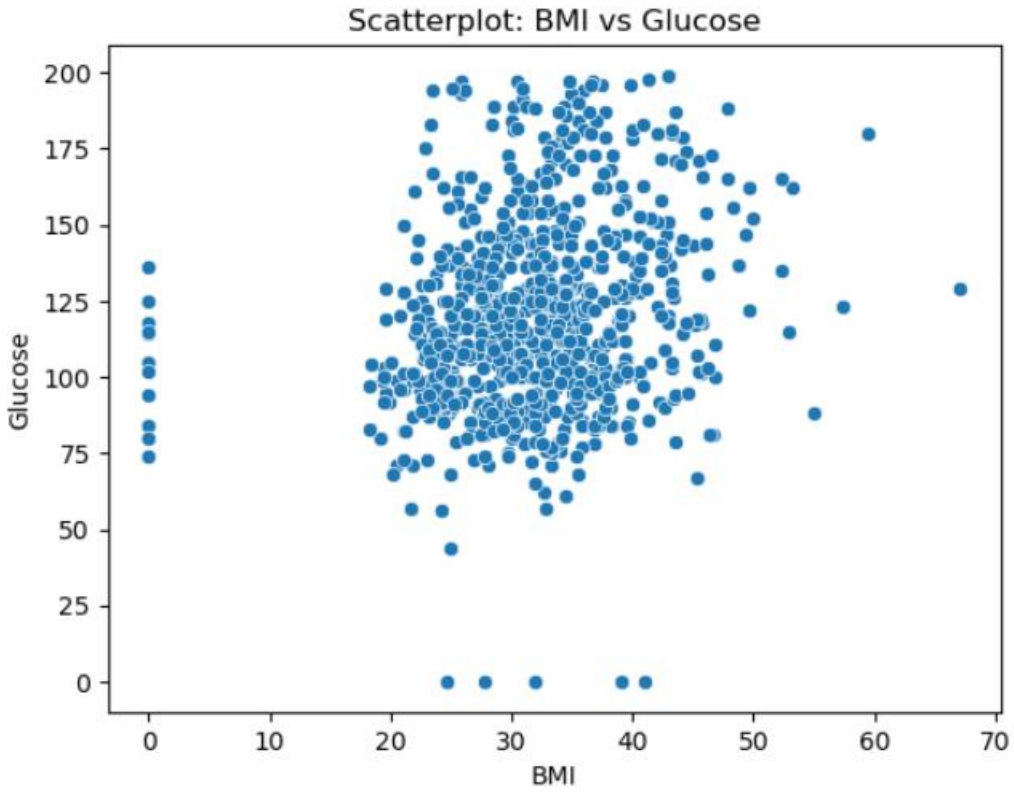
Compute percentage of diabetic (1) and non-diabetic (0) samples

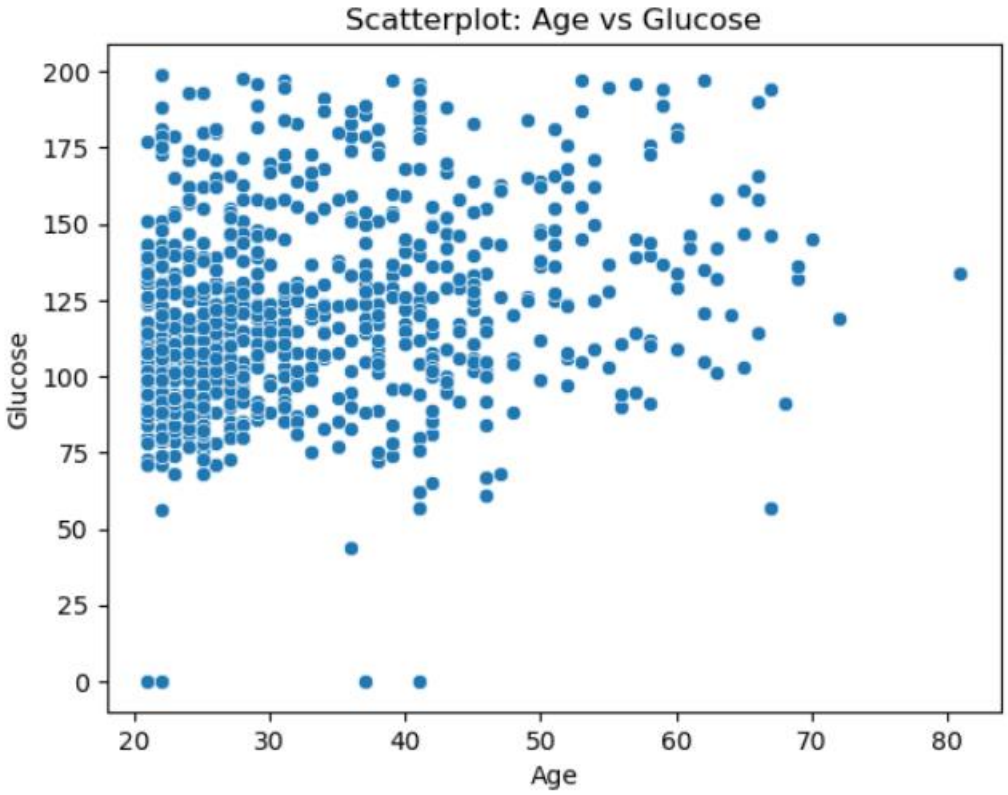
```
import numpy as np
# Computing the %age of diabetic and non-diabetic in the sample
Out0=len([df.Outcome==1])
Out1=len([df.Outcome==0])
Total=Out0+Out1
PC_of_1 = Out1*100/Total
PC_of_0 = Out0*100/Total
plt.figure(dpi = 120,figsize= (5,4))
mask = np.triu(np.ones_like(df.corr(),dtype = bool))
sns.heatmap(df.corr(),mask = mask, fmt = ".2f",annot=True,lw=1,cmap = 'plasma')
plt.yticks(rotation = 0)
plt.xticks(rotation = 90)
plt.title('Correlation Heatmap')
plt.show()
```



Bivariate Analysis using scatterplot

```
:  sns.scatterplot(x='BMI', y='Glucose', data=df)
   plt.title("Scatterplot: BMI vs Glucose")
   plt.show()
   sns.scatterplot(x='Age', y='Glucose', data=df)
   plt.title("Scatterplot: Age vs Glucose")
   plt.show()
```





Linear and logistic regression modeling

```
: from sklearn.linear_model import LinearRegression

X_lin = df[['BMI']]   # Independent variable
y_lin = df['Glucose'] # Dependent variable

lin_model = LinearRegression()
lin_model.fit(X_lin, y_lin)

print("Linear Regression Coefficient (BMI → Glucose):", lin_model.coef_[0])
print("Intercept:", lin_model.intercept_)
print("R² Score:", lin_model.score(X_lin, y_lin))
```

```
Linear Regression Coefficient (BMI → Glucose): 0.8965090270343352
Intercept: 92.2128961628363
R² Score: 0.04887241775173845
```

```python
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score

# Define predictors and target
X_log = df[['Glucose','BMI','Age','Pregnancies']]
y_log = df['Outcome']

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X_log, y_log, test_size=0.3, random_state=42, stratify=y_log)

# Fit logistic regression
log_model = LogisticRegression(max_iter=1000)
log_model.fit(X_train, y_train)

# Predictions
y_pred = log_model.predict(X_test)
y_prob = log_model.predict_proba(X_test)[:,1]
```

```python
print("\nLogistic Regression Coefficients:")
print(pd.DataFrame({'Feature': X_log.columns, 'Coefficient': log_model.coef_[0]}))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("ROC-AUC Score:", roc_auc_score(y_test, y_prob))
```

```
Logistic Regression Coefficients:
       Feature  Coefficient
0      Glucose     0.035302
1          BMI     0.086988
2          Age     0.006661
3  Pregnancies     0.130721

Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.85      0.81       150
           1       0.66      0.54      0.59        81

    accuracy                           0.74       231
   macro avg       0.72      0.69      0.70       231
weighted avg       0.73      0.74      0.73       231

Confusion Matrix:
[[127  23]
 [ 37  44]]
ROC-AUC Score: 0.8223045267489713
```

RESULT:

Thus, various exploratory data analysis has been performed on Pima Indians Diabetesdataset using Python Coding successfully.

Ex.No. 6 Date:	APPLY AND EXPLORE VARIOUS PLOTTING FUNCTIONS ON UCI DATA SETS

Aim:
To apply various plotting functions on UCI data set using Python Coding

- Normal curves
- Density and contour plots
- Correlation and scatter plots
- Histograms
- Three-dimensional plotting

Algorithm

Step 1: Start the Code

Step 2: Import the required packages

Step 3: Load Files (excel/csv/ text) into a Data Frame from UCI data set

Step 4: Describe the data set using built in method

Step 5: Compute Frequency, Mean, Median, Mode, Variance, Standard Deviation,

Step 6: Visualize the data set using Explore various plotting functions on UCI datasets forthe following

- Normal curves
- Density and contour plots
- Correlation and scatter plots
- Histograms
- Three-dimensional plotting

Step 7: Analyze the sample data and do the required operations

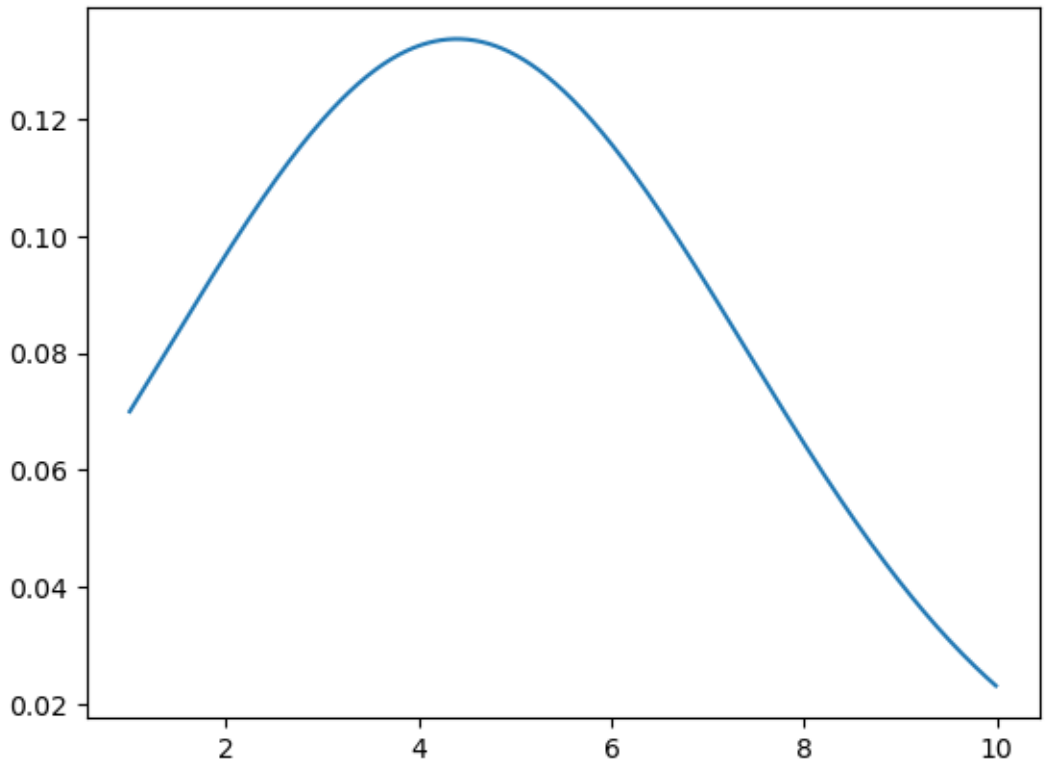
Step 8: Stop the Code

Normal curves

Code

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm
df=pd.read_csv("diabetic_data.csv")
df.head()
mean=df['time_in_hospital'].mean()
std =df['time_in_hospital'].std()
x_axis = np.arange(1, 10, 0.01)
plt.plot(x_axis, norm.pdf(x_axis, mean, std))
plt.show()
```

Output

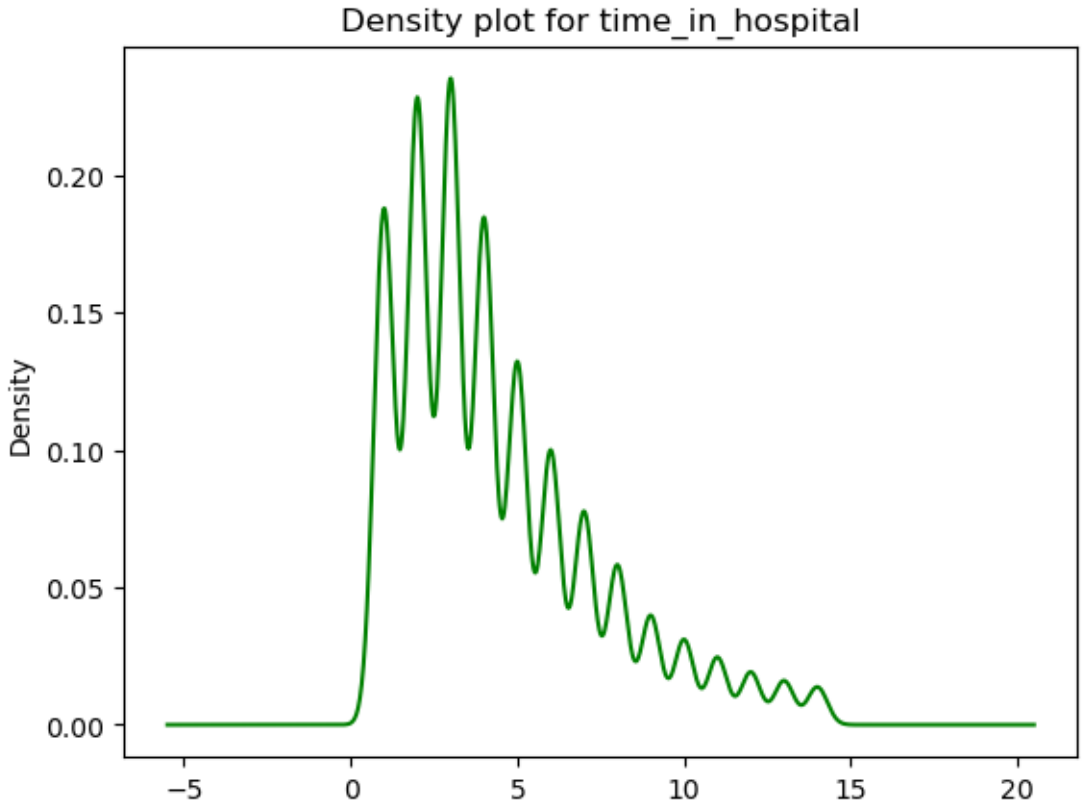


Density and contour plots

Code

```
df.time_in_hospital.plot.density(color='green')
plt.title('Density plot for time_in_hospital')
plt.show()
```

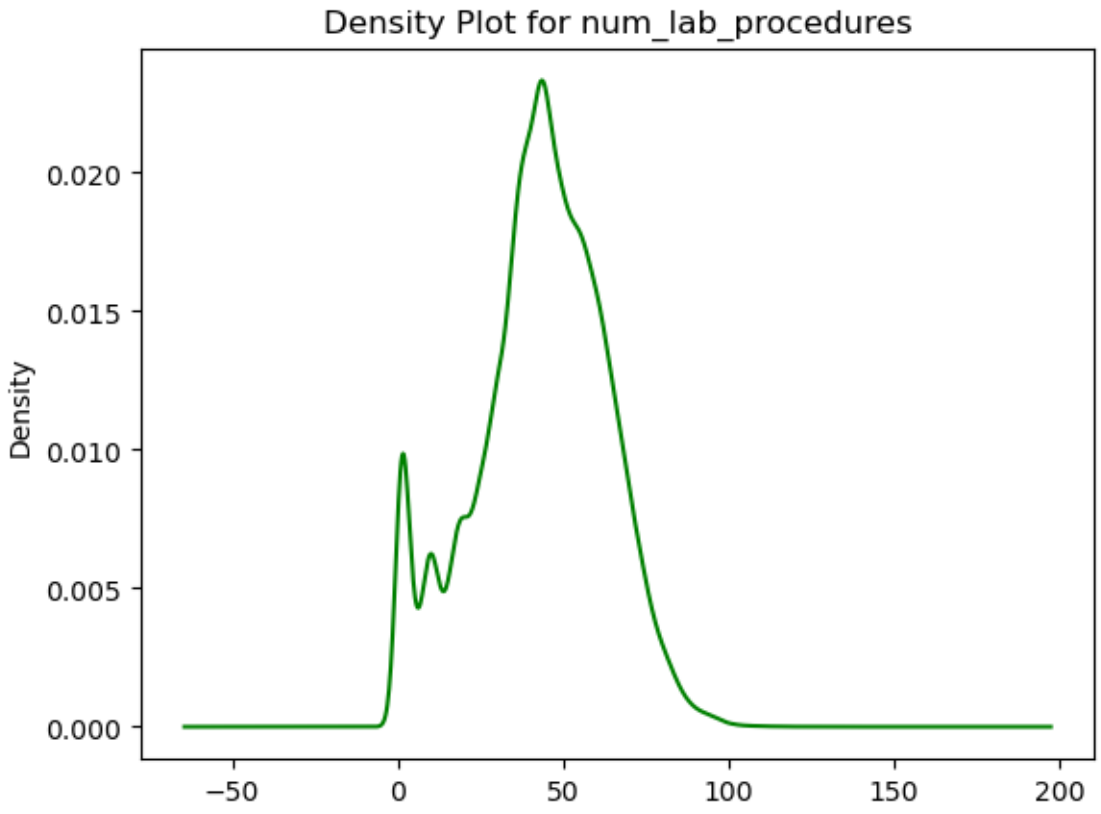
Output



Code

```
df.num_lab_procedures.plot.density(color='green')
plt.title('Density Plot for num_lab_procedures')
plt.show()
```

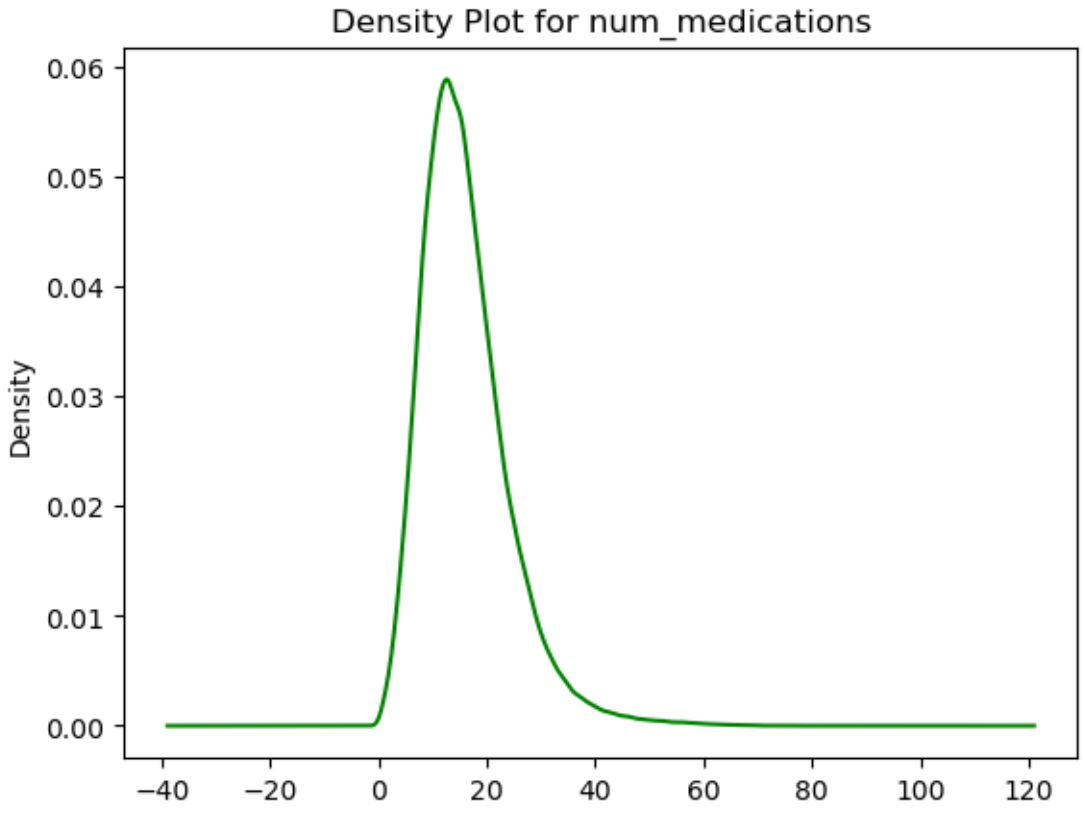
Output



Code

```
df.num_medications.plot.density(color='green')
plt.title('Density Plot for num_medications')
plt.show()
```

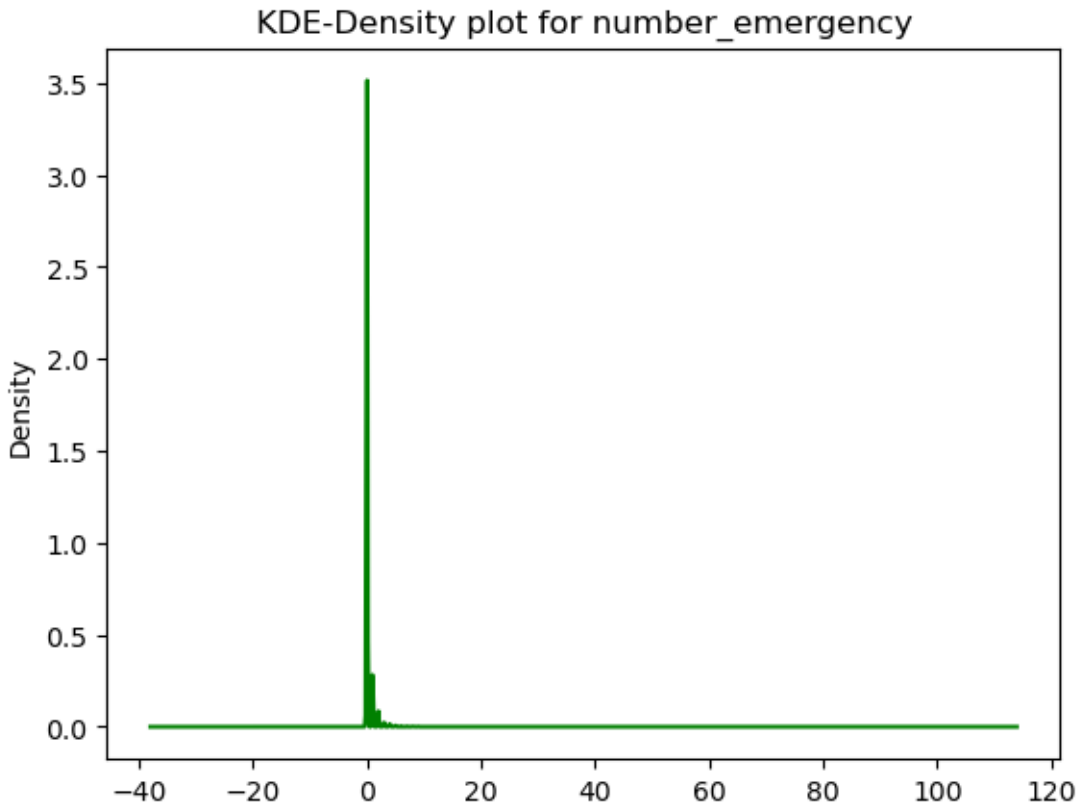
Output



Code

```
df.number_emergency.plot.kde(color='green')
plt.title('KDE-Density plot for number_emergency')
plt.show()
```

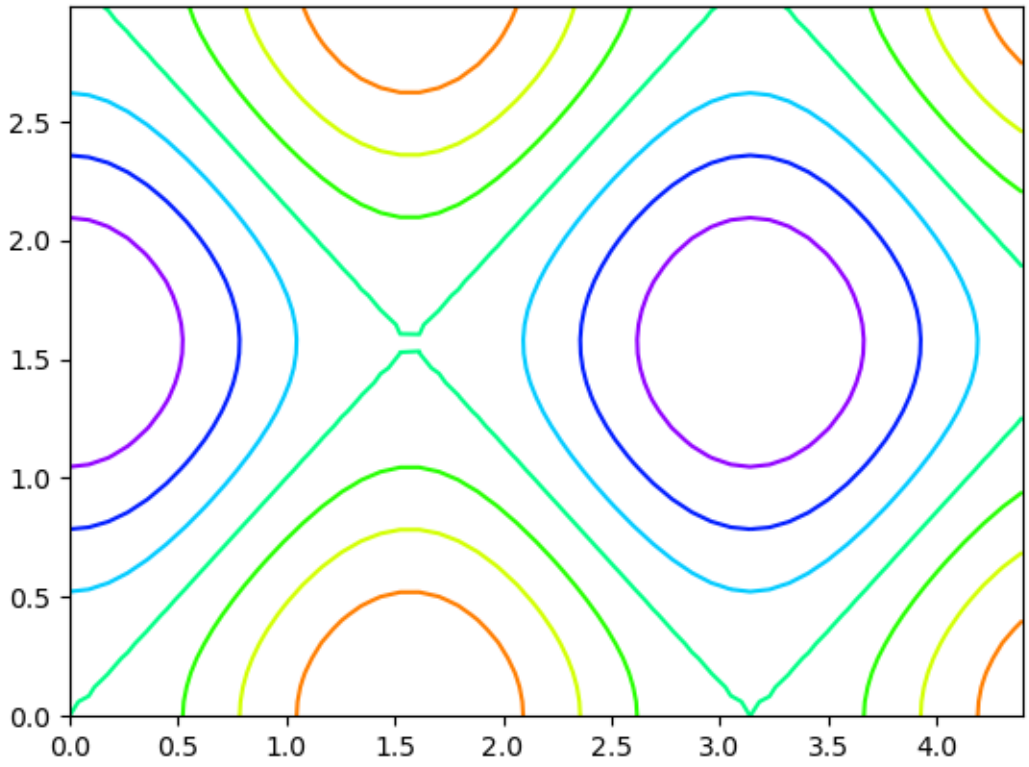
Output



Code

```
def func(x, y):
  return np.sin(x) ** 2 +np.cos(y) **2 # generate 50 values b/w 0 a5
mean=df['time_in_hospital'].mean()
std =df['time_in_hospital'].std()
x = np.linspace(0,mean)
y =np.linspace(0, std)
# Generate combination of grids
X, Y = np.meshgrid(x,y)
Z = func(X, Y)
# Draw rectangular contour plot
plt.contour(X, Y, Z, cmap='gist_rainbow_r');
```

Output

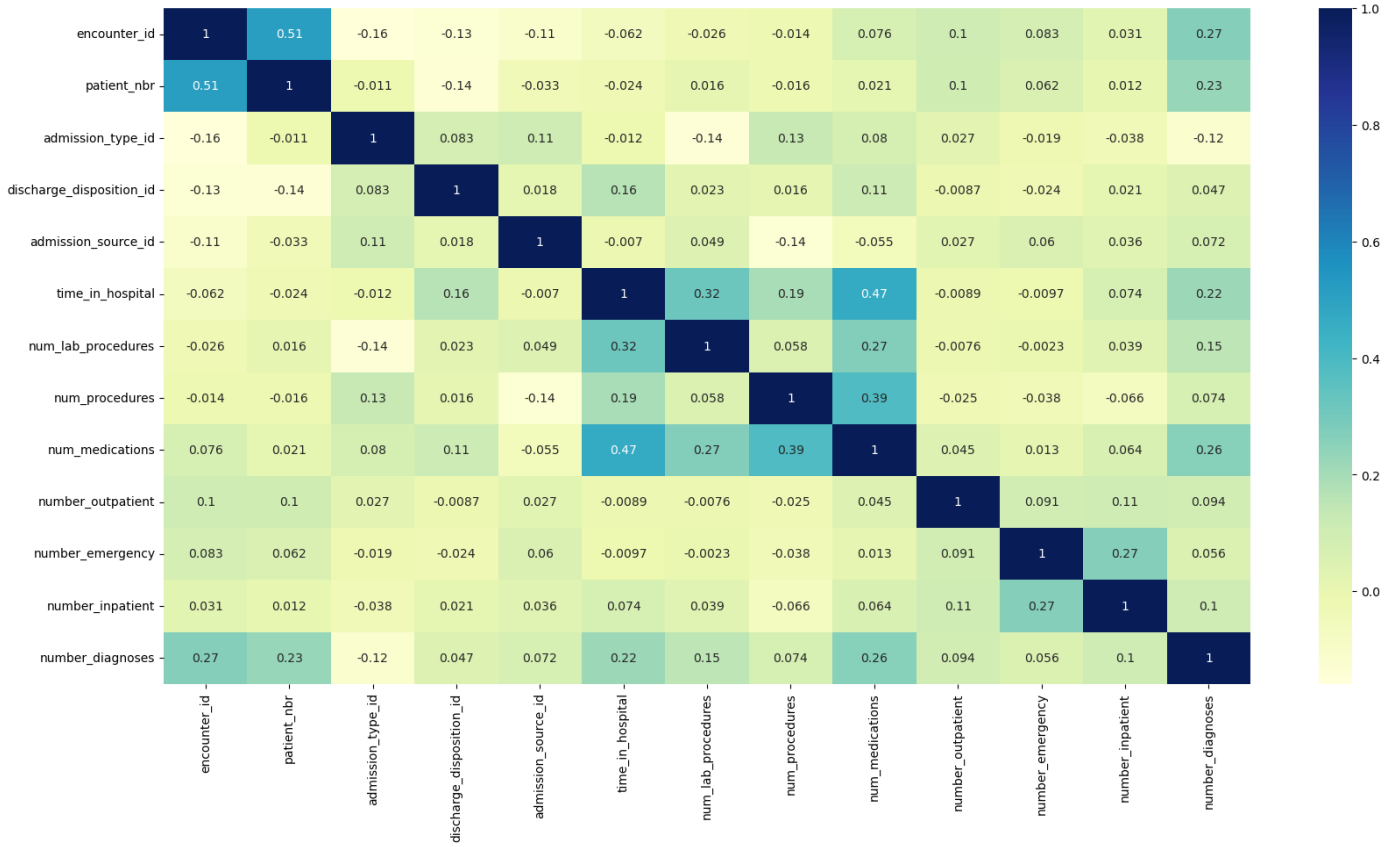


Correlation and scatter plots

Code

```
import seaborn as sb
plt.figure(figsize=(20,10))
dataplot = sb.heatmap(df.corr(), cmap="YlGnBu", annot=True)
```

Output

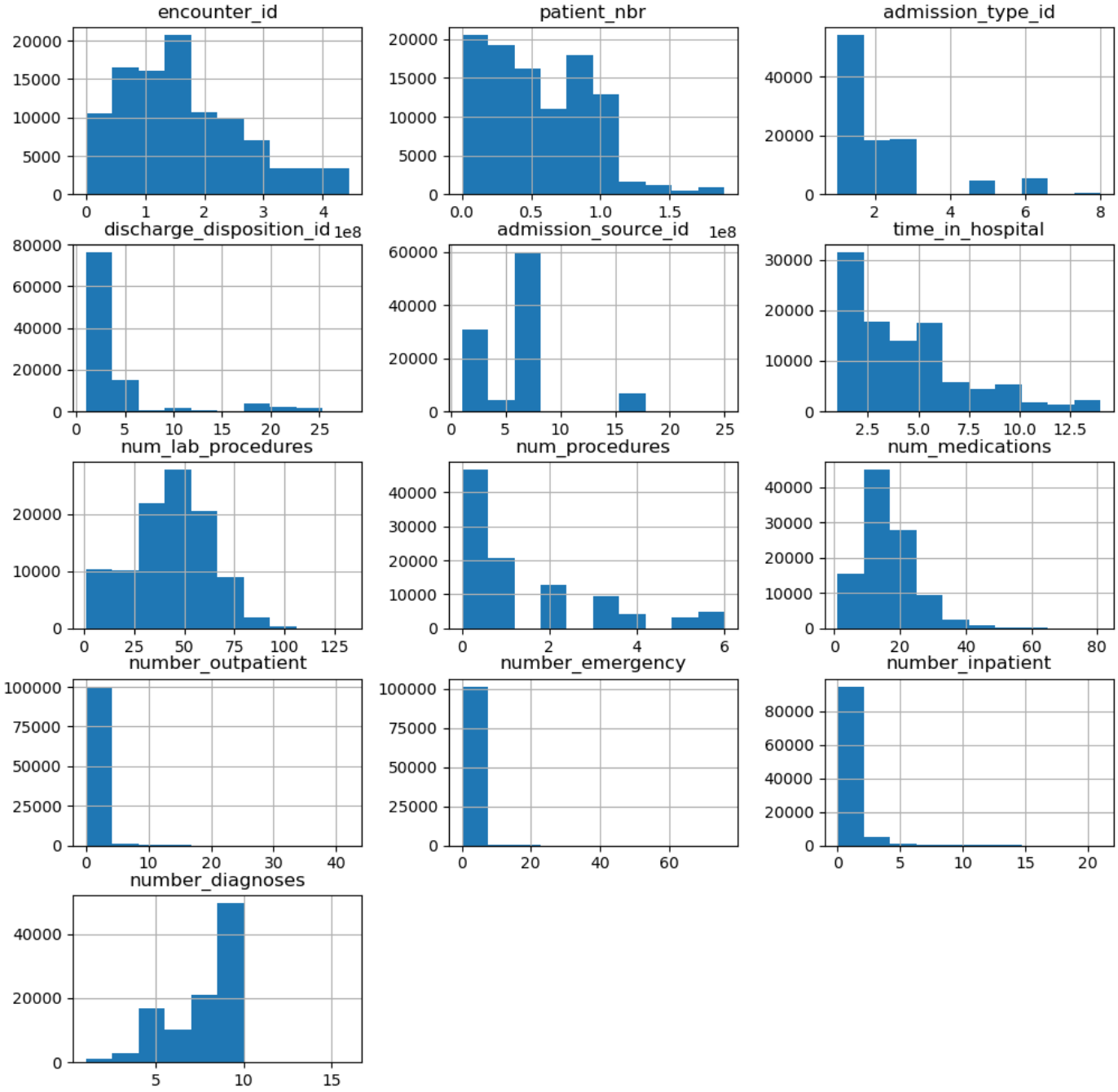


Histograms

Code

```
df.hist(figsize=(12,12),layout=(5,3))
```

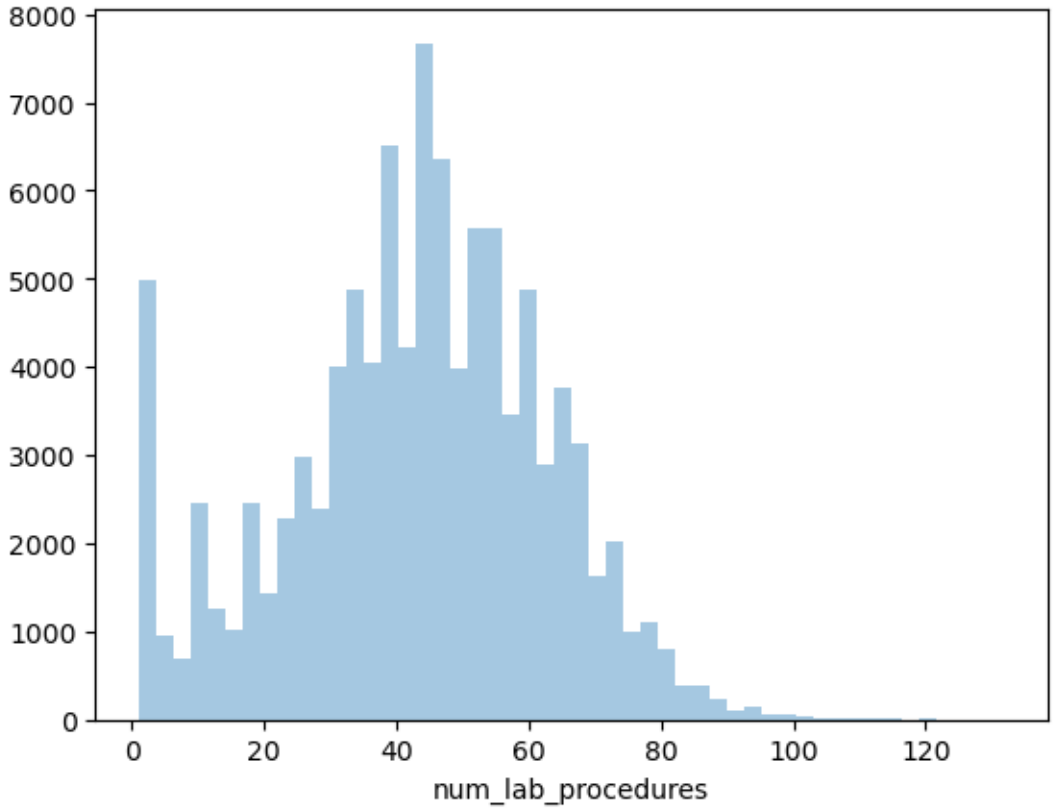
Output



Code

```
# plotting histogram for carat using distplot()
sb.distplot(a=df.num_lab_procedures, kde=False)
# visualizing plot using matplotlib.pyplot library
plt.show()
```

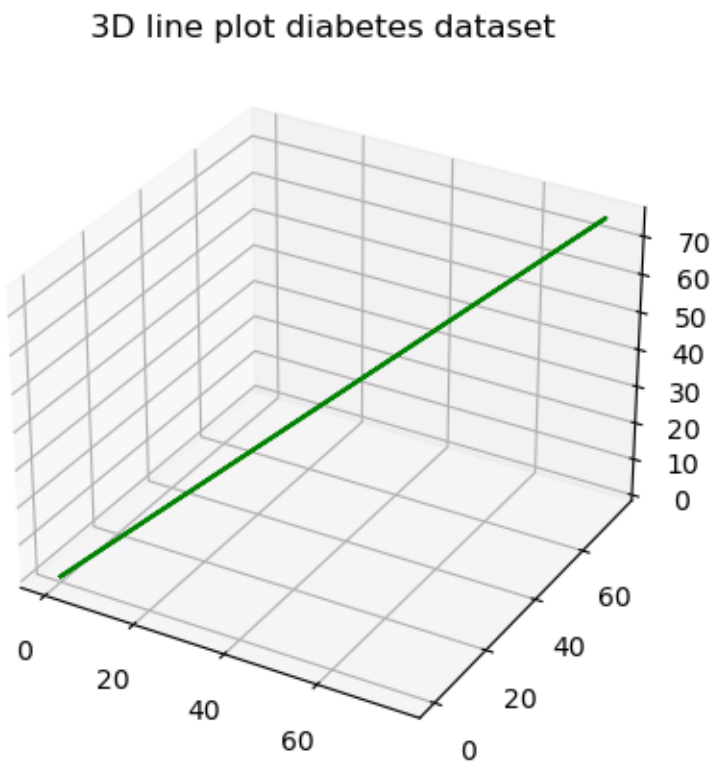
Output



Three-dimensional plotting
Code

```
fig = plt.figure()
ax = plt.axes(projection ='3d')
x =df['number_emergency']
x= pd.Series(x, name= '')
y =df['number_inpatient']
y = pd.Series(x, name='')
z =df['number_outpatient']
z = pd.Series(x, name='')
ax.plot3D(x, y, z,'green')
ax.set_title('3D line plot diabetes dataset')
plt.show()
```

Output



RESULT:

Thus apply various plotting functions on UCI data set using Python Coding.

Ex.No. 7 Date:	VISUALIZING GEOGRAPHIC DATA WITH BASEMAP

Aim:

To visualize Geographic Data using BaseMap module in Python Programming

Algorithm:

Step 1: Start the program

Step 2: Import the required packages

Step 3: Visualize Geographic Data with Basemap

Step 4: Display the Base map using built in method like basemap along with latitude andlongitude parameters

Step 5: Display the Coastal lines meters and Country boundaries using built in methods

Step 6: Fill the Coastal lines meters and Country boundaries with suitable colors

Step 7: Create a global map with a Cylindrical Equidistant Projection, OrthographicProjection, Robinson Projection

Step 8: Stop the program

Create a global map with a Ortho Projection

Code

```
%matplotlib
inline    import
numpy as np
import matplotlib.pyplot as plt from
mpl_toolkits.basemap importBasemap
plt.figure(figsize=(8, 8))
m = Basemap(projection='ortho', resolution=None, lat_0=50, lon_0=-100)
m.bluemarble(scale=0.5);
```

Output

Code

```
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)
# Map (long, lat) to (x, y) for
plotting x, y = m(-122.3, 47.6)
plt.plot(x, y, 'ok', markersize=5)
plt.text(x, y, ' INDIA', fontsize=12);
```

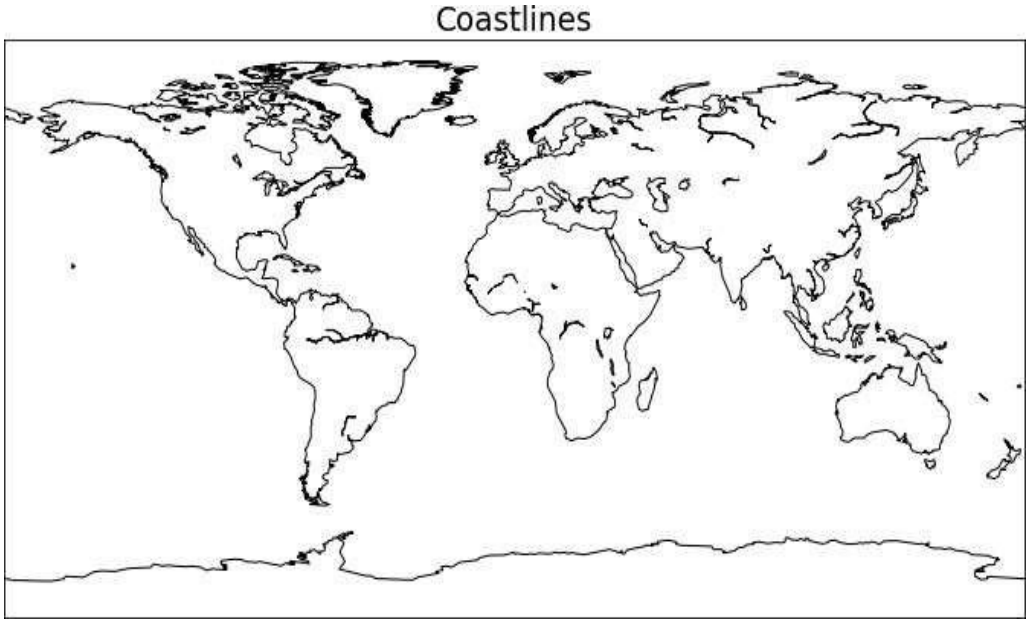
Output

Create a global map with a Coastlines

Code

```
fig = plt.figure(figsize =
(12,12)) m = Basemap()
m.drawcoastlines()
plt.title("Coastlines",
fontsize=20) plt.show()
```

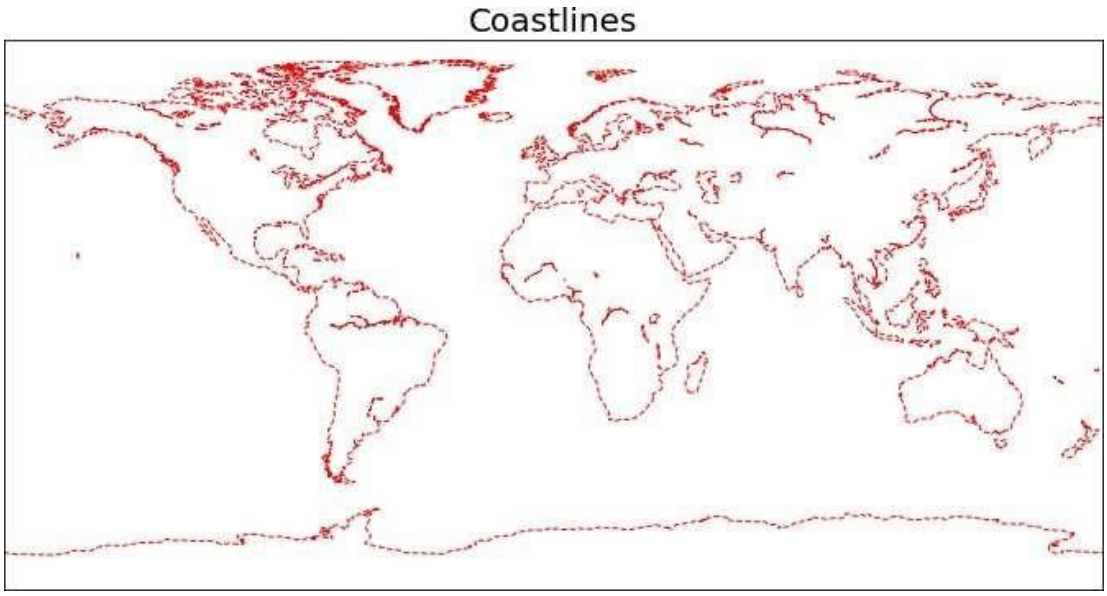
Output



Code

```
fig = plt.figure(figsize =
(12,12)) m = Basemap()
m.drawcoastlines(linewidth=1.0, linestyle='dashed', color='red')plt.title("Coastlines",
fontsize=20)
plt.show()
```

Output



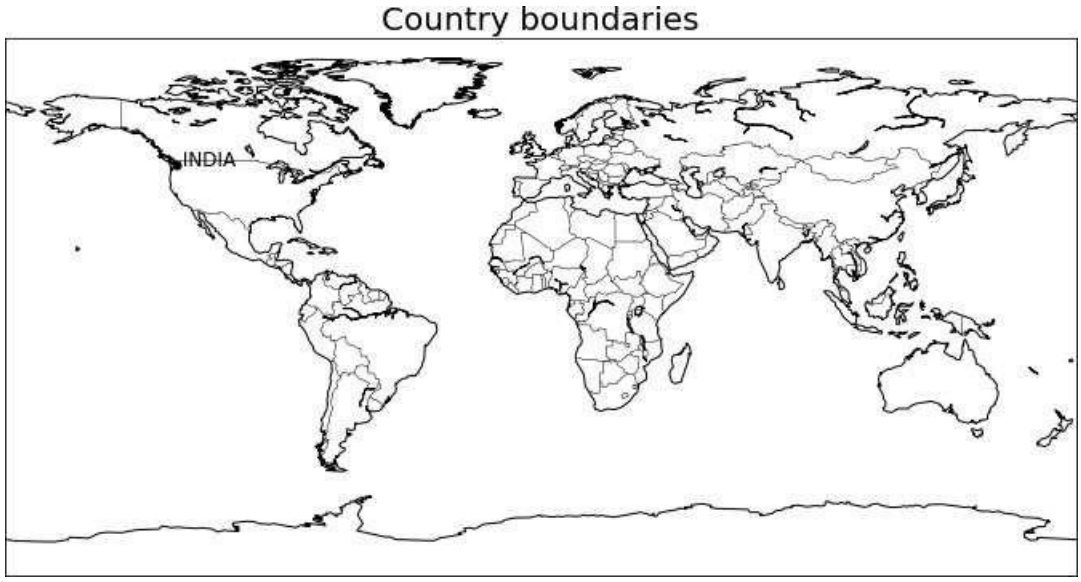
Create a global map with a Country

boundaries

Code

```
fig = plt.figure(figsize =
(12,12)) m = Basemap()
m.drawcoastlines(linewidth=1.0, linestyle='solid', color='black')m.drawcountries()
plt.title("Country boundaries",
fontsize=20) x, y = m(-122.3, 47.6)
plt.plot(x, y, 'ok',
markersize=5) plt.text(x, y, '
INDIA', fontsize=12);
plt.show()
```

Output

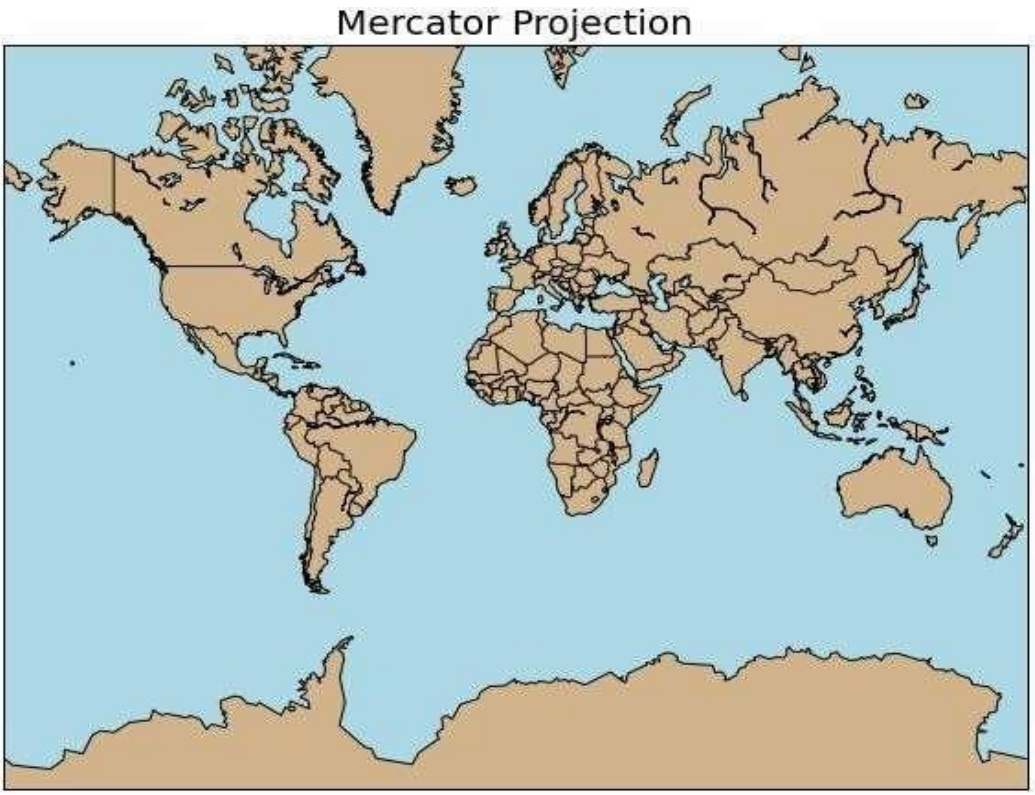


Create a global map with a Mercator Projection

Code

```
fig = plt.figure(figsize = (10,8))
m = Basemap(projection='merc',llcrnrlat=-80,urcrnrlat=80,llcrnrlon=-180,urcrnrlon=180)
m.drawcoastlines()
m.fillcontinents(color='tan',lake_color='lightblue')
m.drawcountries(linewidth=1, linestyle='solid', color='k' )
m.drawmapboundary(fill_color='lightblue')
plt.title("Mercator Projection", fontsize=20)
```

Output



Create a global map with a Cylindrical Equidistant Projection

Code

```
fig = plt.figure(figsize = (10,8))
m = Basemap(projection='cyl',llcrnrlat=-80,urcrnrlat=80,llcrnrlon=-180,urcrnrlon=180)
m.drawcoastlines() m.fillcontinents(color='tan',lake_color='lightblue')
m.drawcountries(linewidth=1, linestyle='solid', color='k' )
m.drawmapboundary(fill_color='lightblue')
plt.title(" Cylindrical Equidistant Projection", fontsize=20)
```

Output

Create a global map with Orthographic Projection

Code

```
fig = plt.figure(figsize = (10,8))
m = Basemap(projection='ortho', lon_0 = 25, lat_0 =10)
m.drawcoastlines()
m.fillcontinents(color='tan',lake_color='lightblue')
m.drawcountries(linewidth=1, linestyle='solid', color='k' )
m.drawmapboundary(fill_color='lightblue')
plt.title("Orthographic Projection", fontsize=18)
```

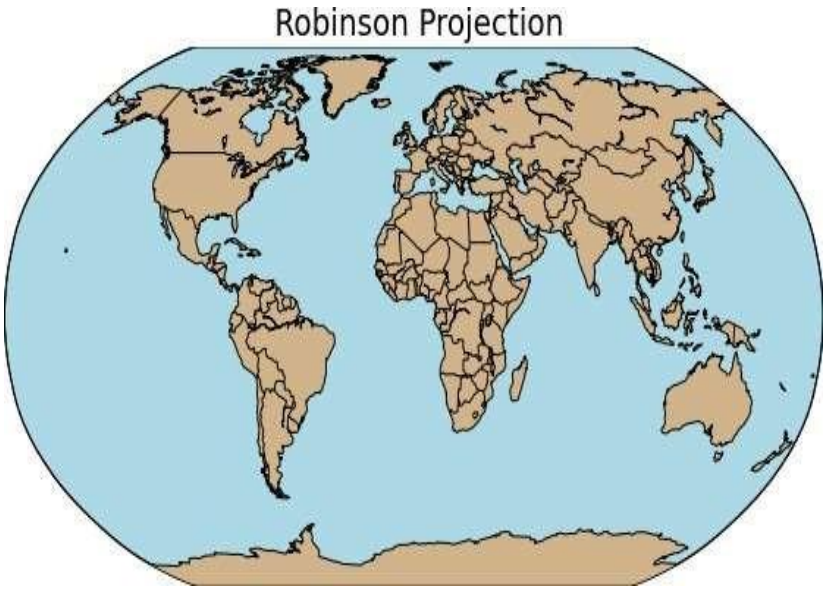
Output

Create a global map with a Robinson Projection

Code

```
fig = plt.figure(figsize = (10,8))
m = Basemap(projection='robin',llcrnrlat=-80,urcrnrlat=80,llcrnrlon=-
180,urcrnrlon=180, lon_0 = 0, lat_0 = 0)
m.drawcoastlines()
m.fillcontinents(color='tan',lake_color='lightbl
ue') m.drawcountries(linewidth=1,
linestyle='solid', color='k' )
m.drawmapboundary(fill_color='lightblue')
plt.title(" Robinson Projection", fontsize=20)
```

Output



RESULT

Thus Geographic Data has been visualized using Base Map module in Python Programming successfully.