KNOWLEDGE INSTITUTE OF TECHNOLOGY (AN AUTONOMOUS INSTITUTION)

Approved by AICTE, New Delhi and Affiliated to Anna University, Chennai

Kakapalayam (Po), Salem – 637 504

RECORD NOTE BOOK

REG. NO.

Certified that this is the bonafide record of work done by Selvan/Selvi ……..…… of the ……...… Semester ……...…… Branch during the year ……..… in the ……..……... Laboratory.

Staff – In charge Head of the Department Submitted for the University Practical

Examination on …….

CONTENTS

S. NO.	DATE	NAME OF THE EXPERIMENT	PAGE NO.	MARKS	SIGNATURE

1		Implementation of various selection and control statements in Python.			
2		Implementation of string operations and functions in Python.			
3		Implementation of List, Tuples and Dictionary in Python.			
4		NumPy, Pandas, SciPy, Seaborn, Stats models, and Matplotlib packages can be downloaded and explored for their features.			
5		Working on NumPy Packages.			
6		Working on Pandas Packages.			
7		The following tasks can be done using the real-time data set from Kaggle.			
8		Explore and apply various plotting functions to Kaggle real-time data sets.			
TOTAL					
AVERAGE					

Ex-No:1	**Implementation of various selection and control statements in Python.**

Aim:

To implement various selection and control statements in Python and understand how they are used to control the flow of a program.

Algorithm:

1. Start.
2. Input the necessary data for the program.
3. Use conditional statements (if, elif, else) to execute specific blocks of code based on certain conditions.
4. Implement loop control statements (for, while) to iterate over sequences or perform repeated tasks.
5. Use break and continue statements to control the flow within loops.

6. Print the output based on the operations performed.

7. End.

(i) If-Else Statement: Check if a number is positive, negative, or zero. Program:

```
# Program to check if a number is positive, negative, or zero
number = int(input("Enter a number: "))
if number > 0:
    print(f"{number} is a positive number.")
elif number < 0:
    print(f"{number} is a negative number.")
else:
    print(f"{number} is zero.")
```

Sample Input:

Enter a number: 5

Sample Output:

5 is a positive number.

(ii) For Loop: Print the sum of the first n natural numbers. Program:

```
# Program to calculate the sum of first n natural
numbers n = int(input("Enter the value of n: "))
sum = 0
for i in range(1, n + 1):
    sum += i
print(f"The sum of the first {n} natural numbers is
{sum}.")
```

Sample Input:

Enter the value of n: 5

Sample Output:

The sum of the first 5 natural numbers is 15.

(iii) While Loop: Find the factorial of a number. Program:

```
# Program to calculate the factorial of a
number number = int(input("Enter a number:
"))
factorial = 1
while number > 0:
    factorial *= number
    number -= 1
print(f"The factorial is {factorial}.")
```

Sample Input:

Enter a number: 4

Sample Output:

The factorial is 24.

(iv) Break Statement: Find the first multiple of 5 in a list. Program:

```
# Program to find the first multiple of 5 in a
list numbers = [1, 22, 18, 25, 33, 40]
```



```
for number in numbers:
    if number % 5 == 0:
        print(f"The first multiple of 5 in the list is {number}.")
        break
```

Sample Input:

```
numbers = [1, 22, 18, 25, 33, 40]
```

Sample Output:

The first multiple of 5 in the list is 25.

(v) Continue Statement: Print all even numbers from 1 to 10, skipping odd numbers. Program:

```
# Program to print all even numbers from 1 to 10, skipping odd
numbers for i in range(1, 11):
    if i % 2 != 0:
        continue
    print(i)
```

Sample Output:

2

4

6

8

10

PERFORMANCE	30	
OBSERVATION	30	
RECORD	40	
TOTAL	100	

Result:

The implementation of various selection and control statements in Python has been successfully demonstrated.

Ex-No:2	Implementation of string operations and functions in Python.

Aim:

To implement various string operations and functions in Python to manipulate and analyze string data.

Algorithm:

1. Start.
2. Input a string from the user.

3. Perform basic string operations such as concatenation, slicing, and repetition.
4. Use string functions like len(), upper(), lower(), find(), replace(), split(), and join() to manipulate the string.
5. Print the results of each operation.
6. End.

(i) String Concatenation: Combine two strings.

Program:

```
# Program to concatenate two strings
str1 = input("Enter the first string: ")
str2 = input("Enter the second string: ")
result = str1 + " " + str2
print(f"The concatenated string is: {result}")
```

Sample Input:

Enter the first string: Hello

Enter the second string: World

Sample Output:

The concatenated string is: Hello World

(ii) String Slicing: Extract a substring from a string.

Program:

```
# Program to extract a substring from a string
string = input("Enter a string: ")
start = int(input("Enter the start index: "))
end = int(input("Enter the end index: "))
```



```
substring = string[start:end]
print(f"The extracted substring is: {substring}")
```

Sample Input:

Enter a string: OpenAI

Enter the start index: 1

Enter the end index: 4

Sample Output:

The extracted substring is: pen

(iii) String Functions: Convert to uppercase and find the length of the string. Program:

```
# Program to convert a string to uppercase and find its
length string = input("Enter a string: ")
upper_string = string.upper()
length = len(string)
print(f"The uppercase string is: {upper_string}")
print(f"The length of the string is: {length}")
```

Sample Input:

Enter a string: Python

Sample Output:

The uppercase string is: PYTHON

The length of the string is: 6

(iv) String Replacement: Replace a substring in a string.

Program:

```
# Program to replace a substring in a string
string = input("Enter a string: ")
old_substring = input("Enter the substring to be replaced: ")
new_substring = input("Enter the new substring: ")

replaced_string = string.replace(old_substring, new_substring)
```



```
print(f"The modified string is: {replaced_string}")
```

Sample Input:

Enter a string: I love programming.

Enter the substring to be replaced: programming

Enter the new substring: Python

Sample Output:

The modified string is: I love Python.

(v) String Splitting and Joining: Split a string and then join it back.

Program:

```
# Program to split a string and then join it back
string = input("Enter a string: ")
words = string.split()
joined_string = "-".join(words)
print(f"The split words are: {words}")
print(f"The joined string is: {joined_string}")
```

Sample Input:

Enter a string: Learn Python Programming

Sample Output:

The split words are: ['Learn', 'Python', 'Programming']

The joined string is: Learn-Python-Programming

PERFORMANCE	30	
OBSERVATION	30	
RECORD	40	
TOTAL	100	

Result:

The implementation of string operations and functions in Python has been successfully demonstrated.

Ex-No: 3	**Implementation of List, Tuples and Dictionary in Python.**

Aim:

To implement and manipulate data using List, Tuple, and Dictionary in Python and understand their unique properties and use cases.

Algorithm:

1. Start.
2. Create a list, tuple, and dictionary with sample data.
3. Perform basic operations on each data structure, such as accessing elements,

adding/removing items, and iterating through elements.

4. Use specific methods associated with each data structure to modify and analyze the data.
5. Print the results of the operations.
6. End.

(i) List Operations: Create a list, add elements, remove elements, and iterate through the list.

Programs:

```
# Program to demonstrate list operations
my_list = [10, 20, 30, 40, 50]
# Adding elements
my_list.append(60)
my_list.insert(2, 25)
# Removing elements
my_list.remove(40)
popped_element = my_list.pop()
# Iterating through the list
print("List elements:")
for item in my_list:
    print(item)
print(f"Popped element: {popped_element}")
```


Sample Output:

```
List elements:
10
20
25
30
50
Popped element: 60
```

(ii) Tuple Operations: Create a tuple, access elements, and iterate through the

tuple. Program:

```
# Program to demonstrate tuple operations
my_tuple = (1, 2, 3, 4, 5)
# Accessing elements
first_element = my_tuple[0]
last_element = my_tuple[-1]
# Iterating through the tuple
print("Tuple elements:")
for item in my_tuple:
    print(item)
print(f"First element: {first_element}")
print(f"Last element: {last_element}")
```

Sample Output:

```
Tuple elements:
1
2
3
4
5
First element: 1
Last element: 5
```


(iii) Dictionary Operations: Create a dictionary, add key-value pairs, remove key-value pairs, and iterate through the dictionary.

Program:

```
# Program to demonstrate dictionary operations
my_dict = {'name': 'John', 'age': 25, 'city': 'New York'}
# Adding key-value pairs
my_dict['email'] = 'john@example.com'
# Removing key-value pairs
del my_dict['age']
# Iterating through the dictionary
```

```
print("Dictionary elements:")
for key, value in my_dict.items():
    print(f"{key}: {value}")
```

Sample Output:

```
Dictionary elements:
name: John
city: New York
email: john@example.com
```

PERFORMANCE	30	
OBSERVATION	30	
RECORD	40	
TOTAL	100	

Result:

The implementation of List, Tuple, and Dictionary in Python has been successfully demonstrated.

Ex –No: 4	**NumPy, Pandas, SciPy, Seaborn, Stats models, and Matplotlib packages can be downloaded and explored for their features.**

How to Install Anaconda & Run Jupyter Notebook

Instructions To Install Anaconda and Run Jupyter Notebook

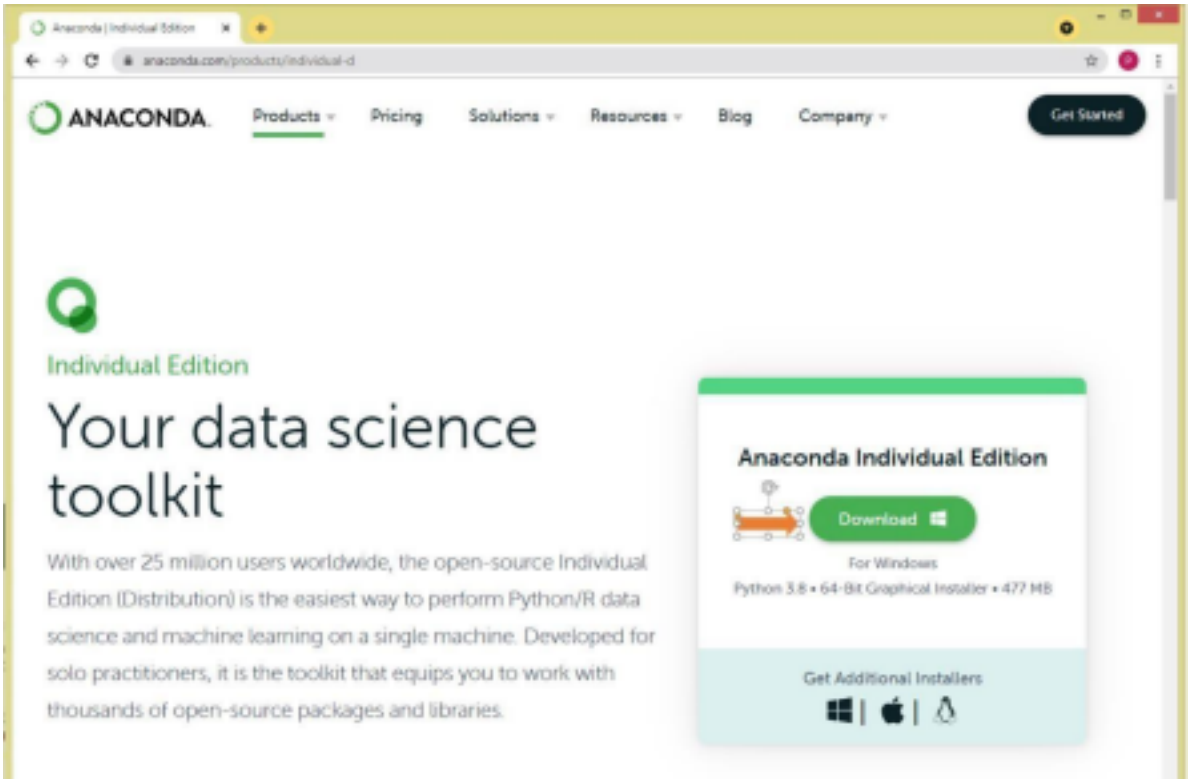
- Download & Install Anaconda Distribution
- Create Anaconda Environment
- Install and Run Jupyter Notebook

Download & Install Anaconda Distribution

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

Download Anaconda Distribution

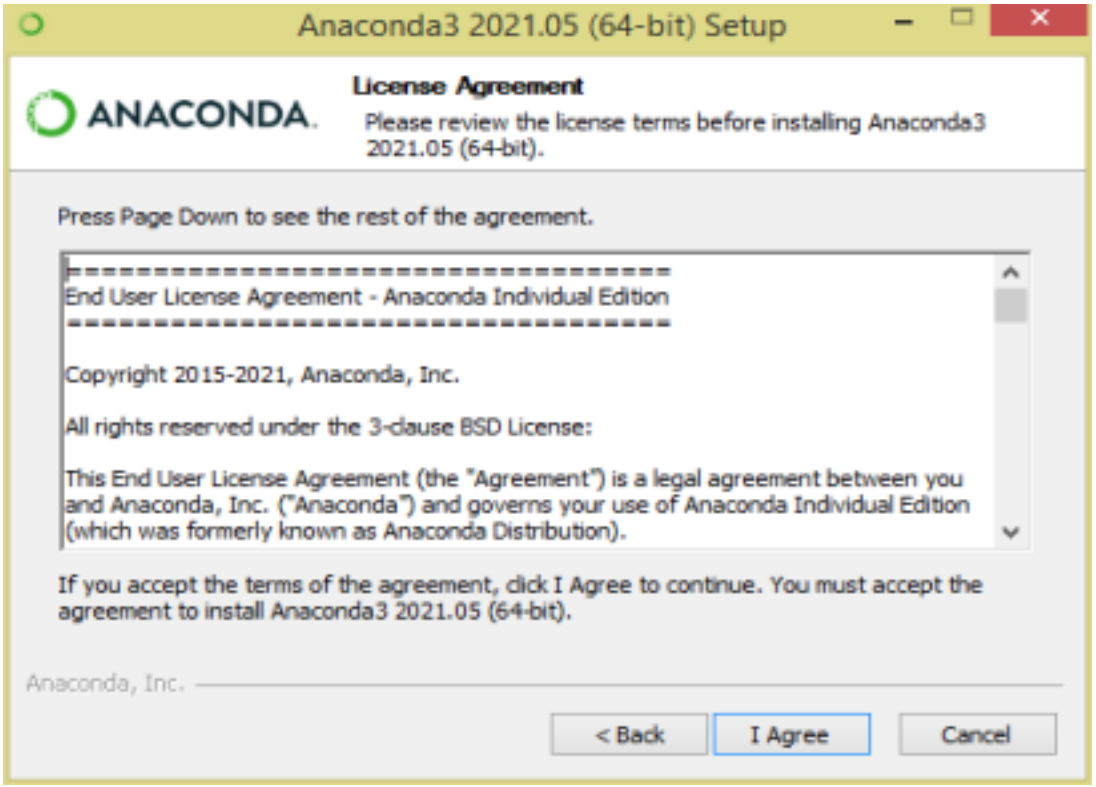
Go to https://anaconda.com/ and select **Anaconda Individual Edition** to download the latest version of Anaconda. This downloads the .exe file to the windows download folder.

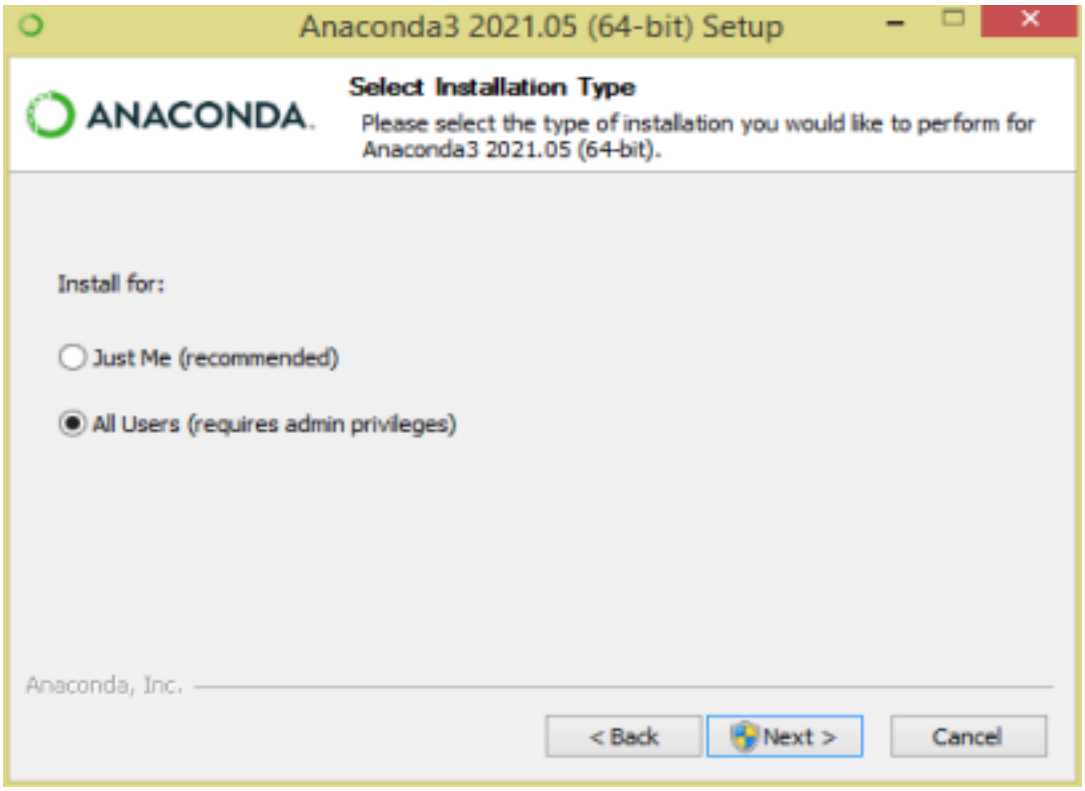


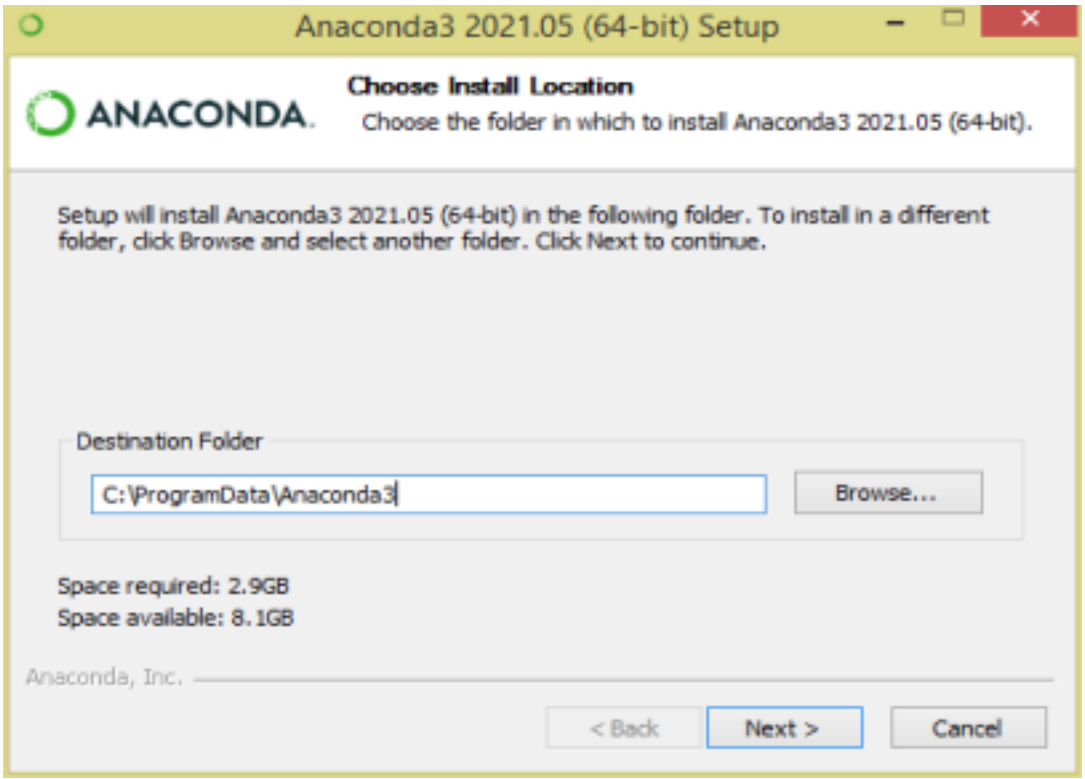
Install Anaconda

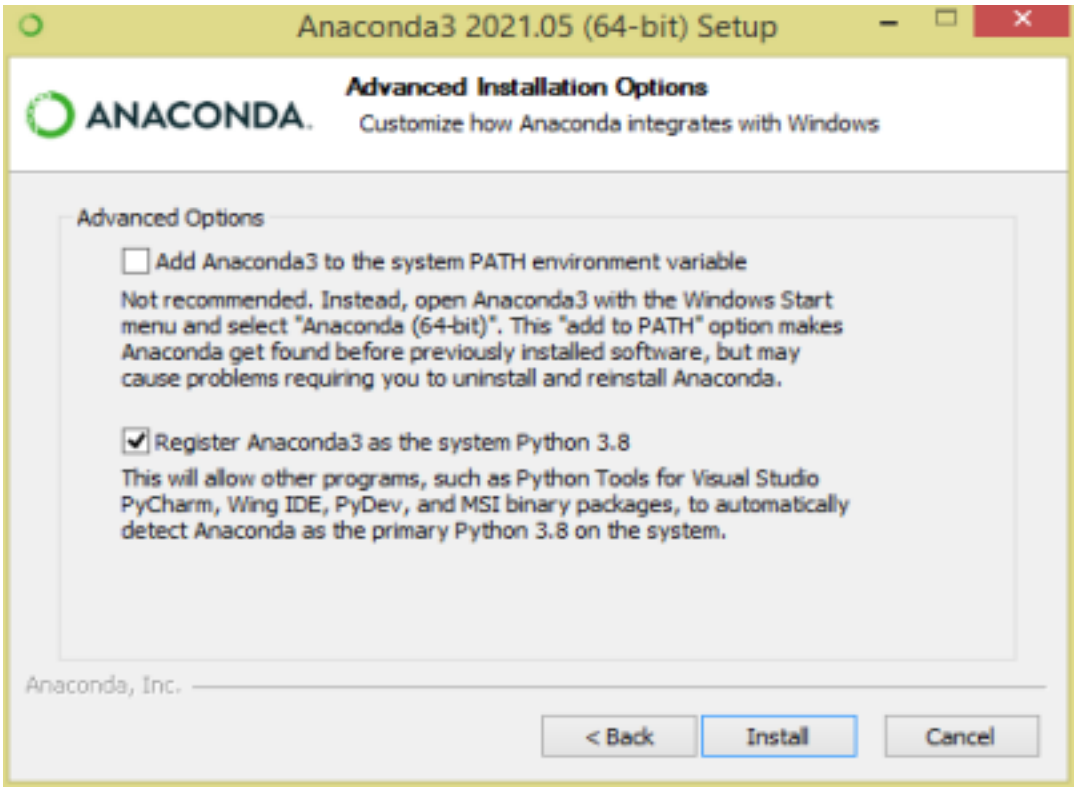
By double-clicking the .exe file starts the Anaconda installation. Follow the below screen shot's and complete the installation

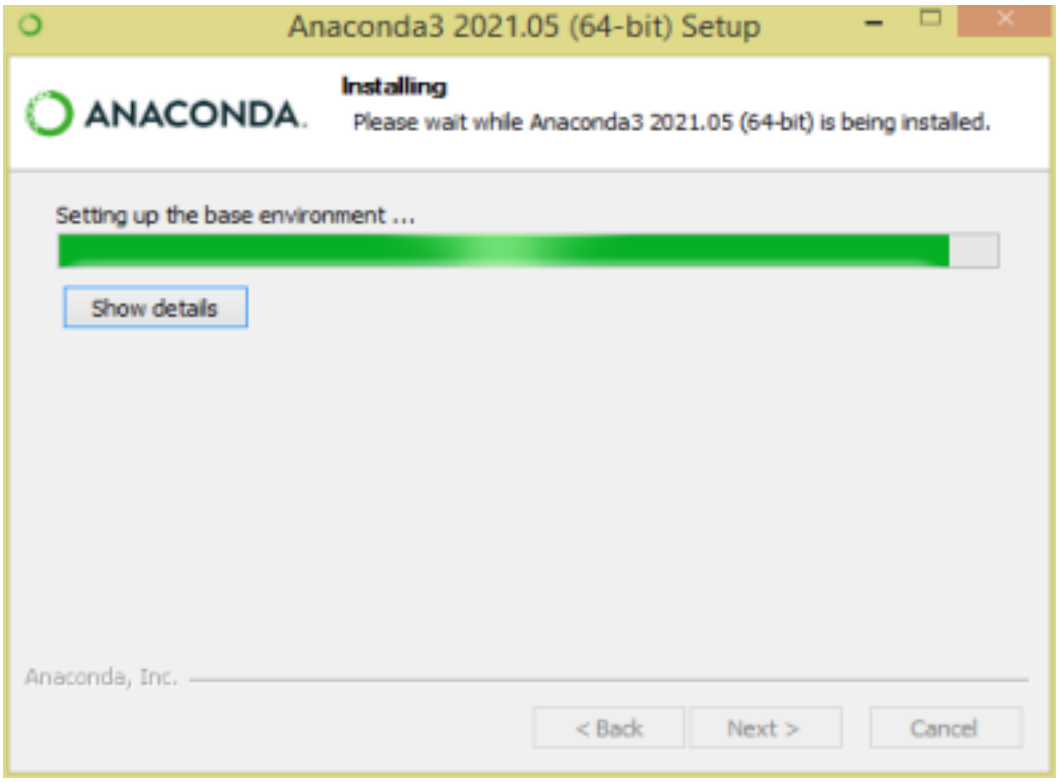

Anaconda3 2021.05 (64-bit) Setup
ANACONDA.
Welcome to Anaconda3 2021.05 (64-bit) Setup
Setup will guide you through the installation of Anaconda3 2021.05 (64-bit).
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Next >
Cancel

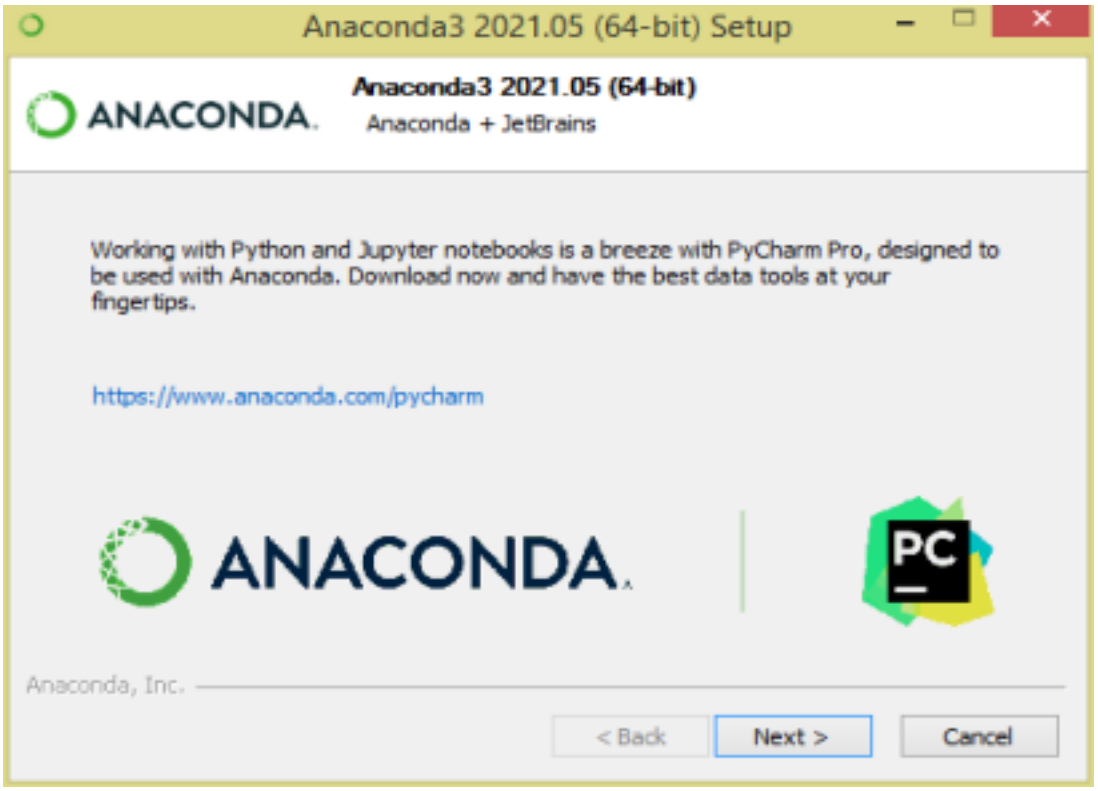

Anaconda3 2021.05 (64-bit) Setup
ANACONDA.
License Agreement
Please review the license terms before installing Anaconda3 2021.05 (64-bit).
Press Page Down to see the rest of the agreement.
End User License Agreement - Anaconda Individual Edition
Copyright 2015-2021, Anaconda, Inc.
All rights reserved under the 3-clause BSD License:
This End User License Agreement (the "Agreement") is a legal agreement between you and Anaconda, Inc. ("Anaconda") and governs your use of Anaconda Individual Edition (which was formerly known as Anaconda Distribution).
If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Anaconda3 2021.05 (64-bit).
Anaconda, Inc.
< Back
I Agree
Cancel

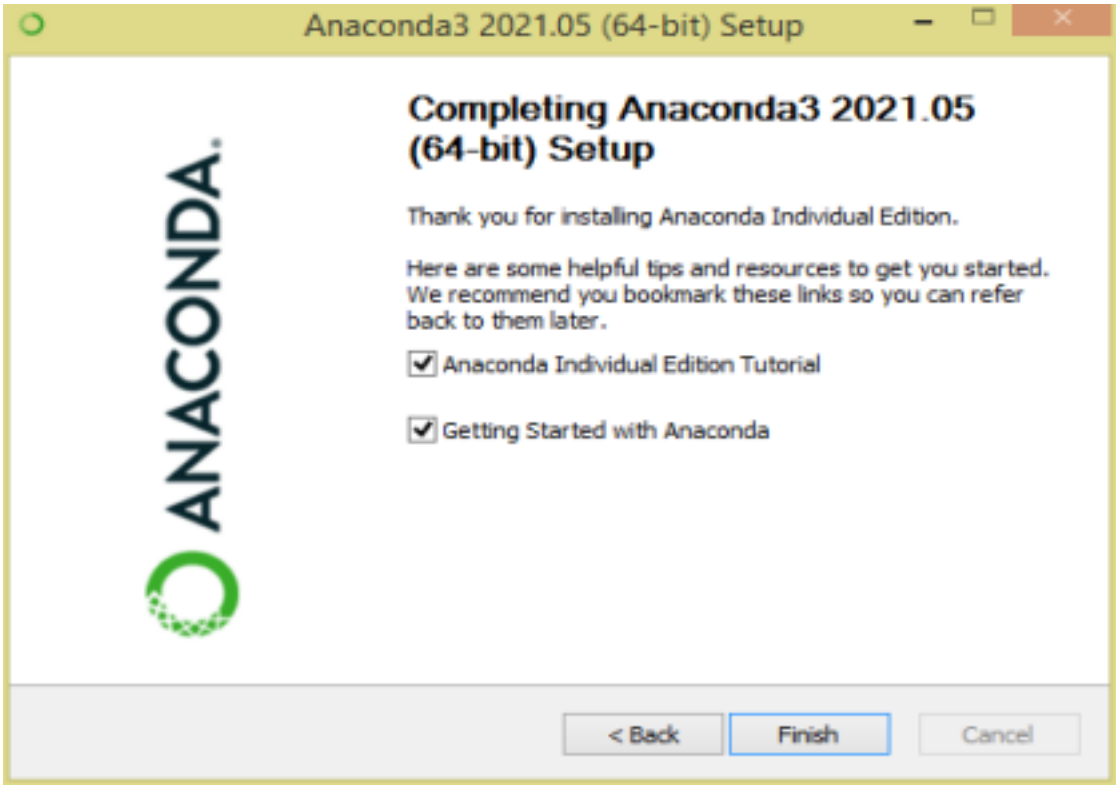

Anaconda3 2021.05 (64-bit) Setup
ANACONDA.
Select Installation Type
Please select the type of installation you would like to perform for Anaconda3 2021.05 (64-bit).
Install for:
Just Me (recommended)
All Users (requires admin privileges)
Anaconda, Inc.
< Back
Next >
Cancel


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.

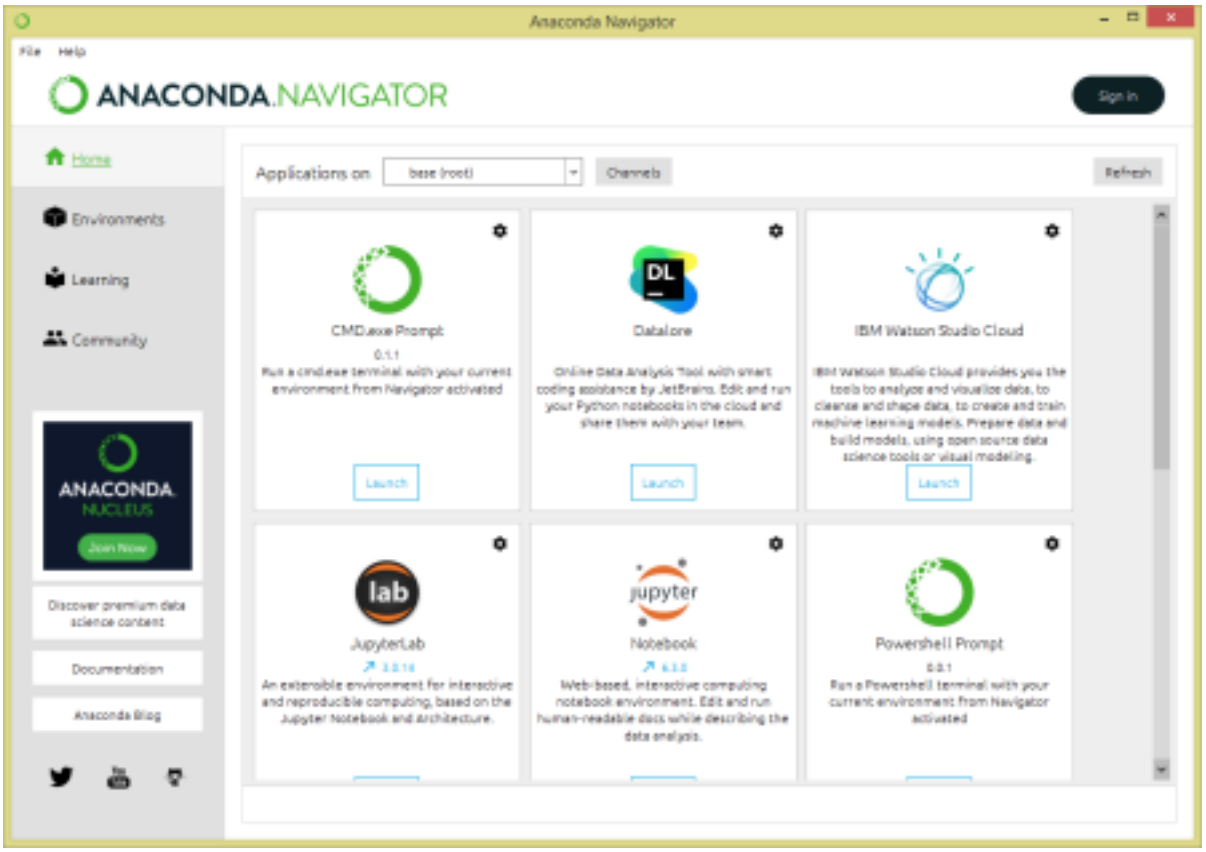
Create Anaconda Environment from Navigator

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

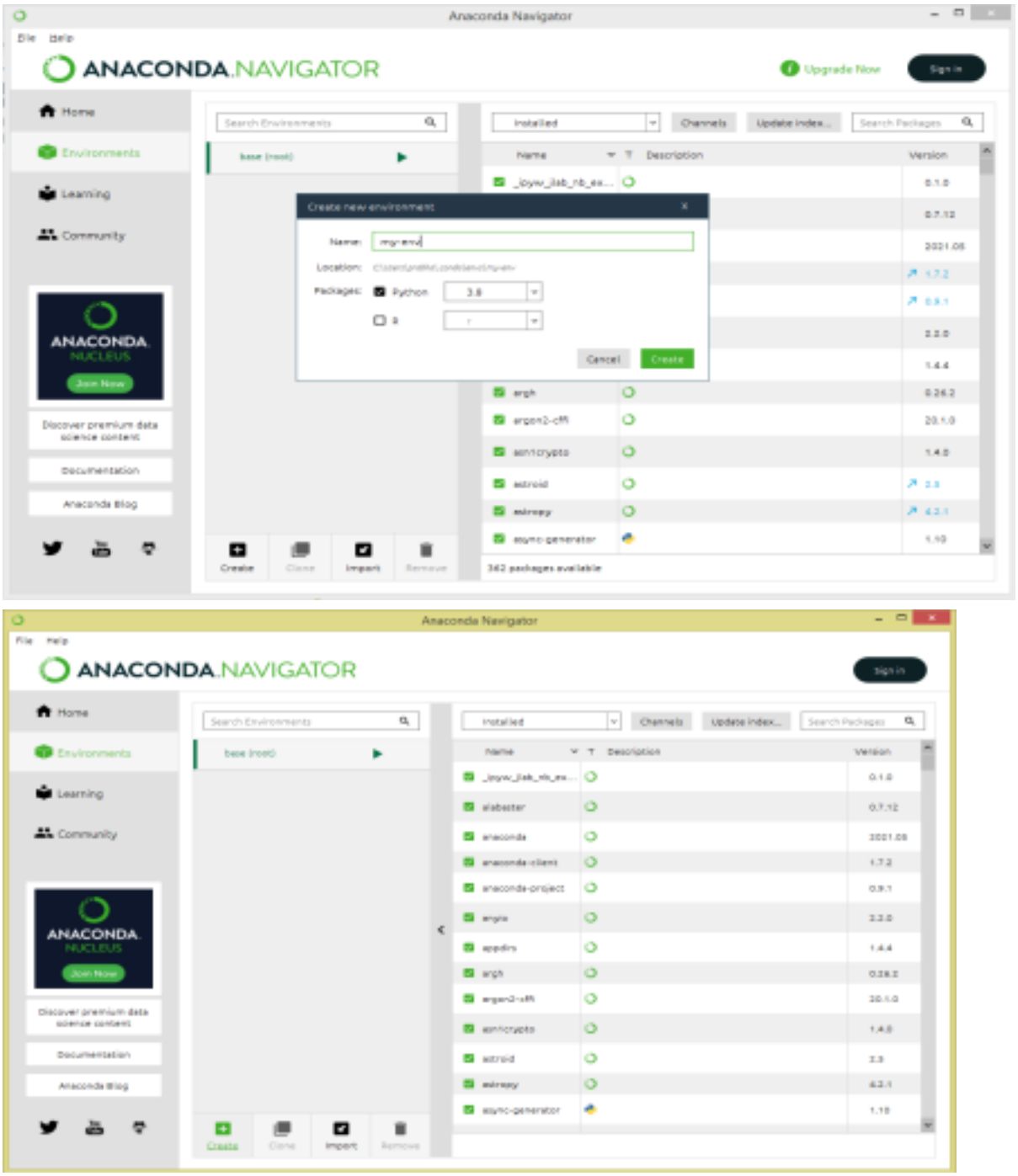
Open Anaconda Navigator

Open Anaconda Navigator from windows start or by searching it.



Create an Environment to Run Jupyter Notebook

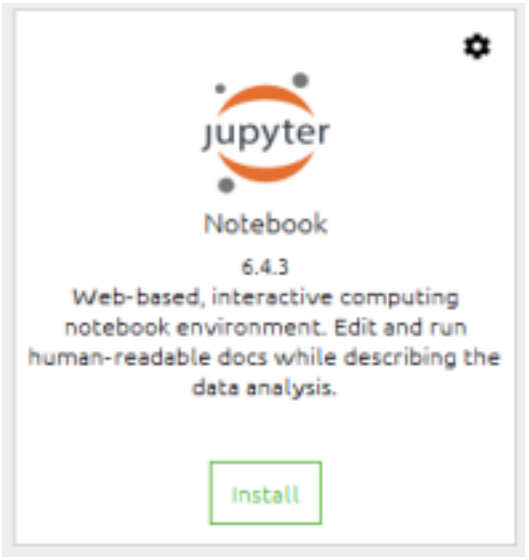
This is optional but recommended to create an environment before you proceed. This gives complete segregation of different package installs for different projects you would be working on. If you already have an environment, you can use it too.



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

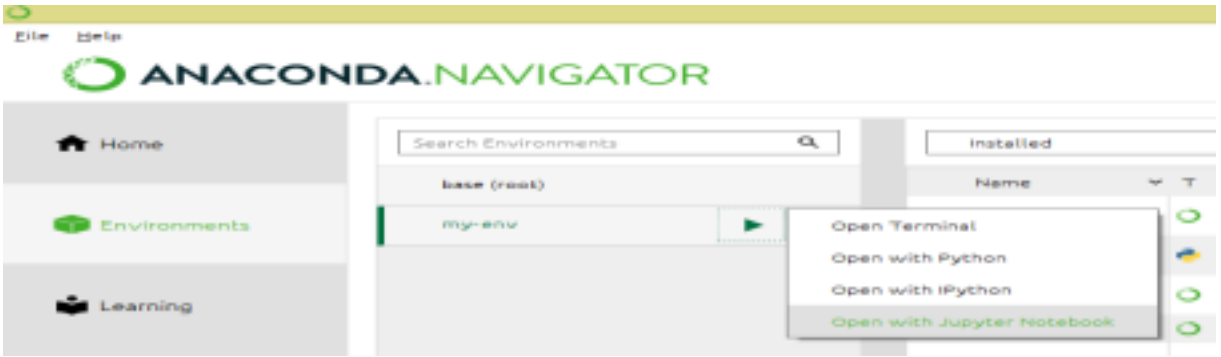
Install and Run Jupyter Notebook

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.



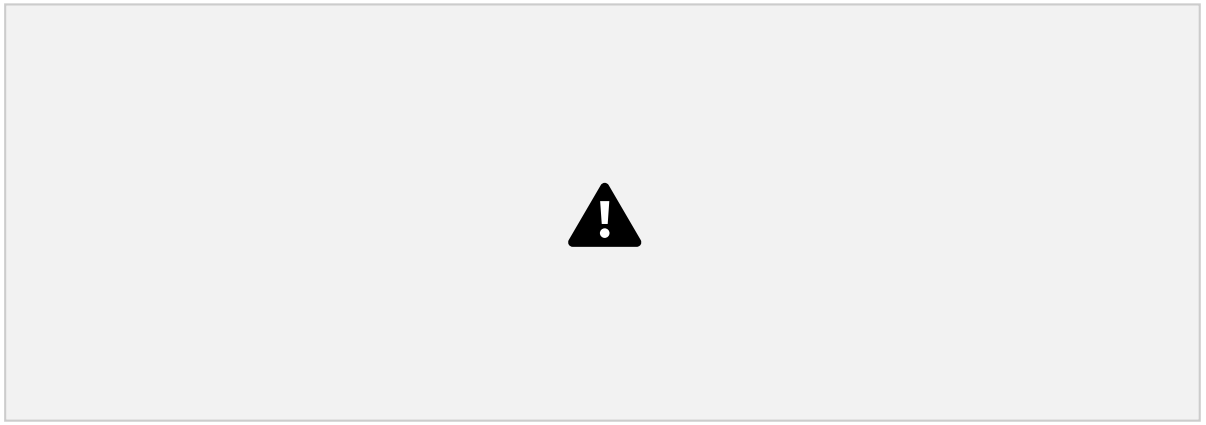
It will take a few seconds to install Jupyter to your environment, once the install completes, you can open Jupyter from the same screen or by accessing Anaconda Navigator -> Environments -> your

environment (mine pandas-tutorial) -> select Open With Jupyter Notebook.

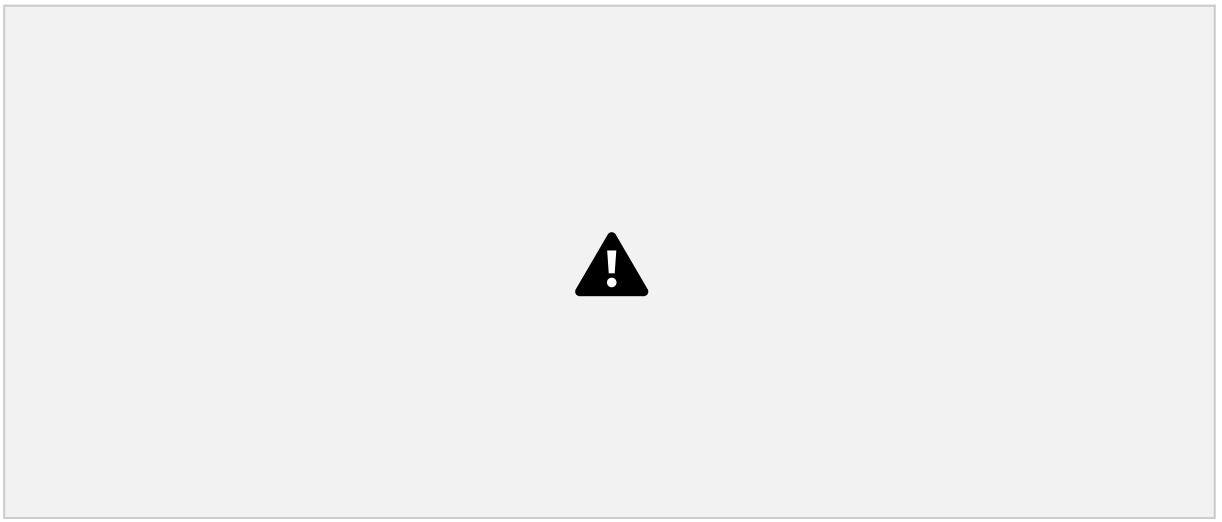


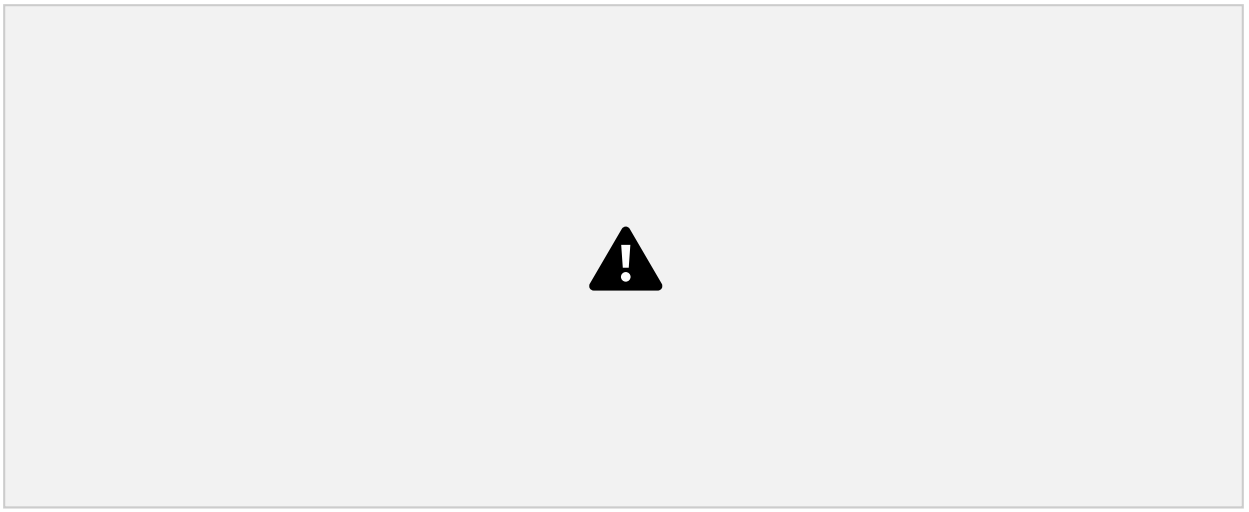
This opens up Jupyter Notebook in the default browser.

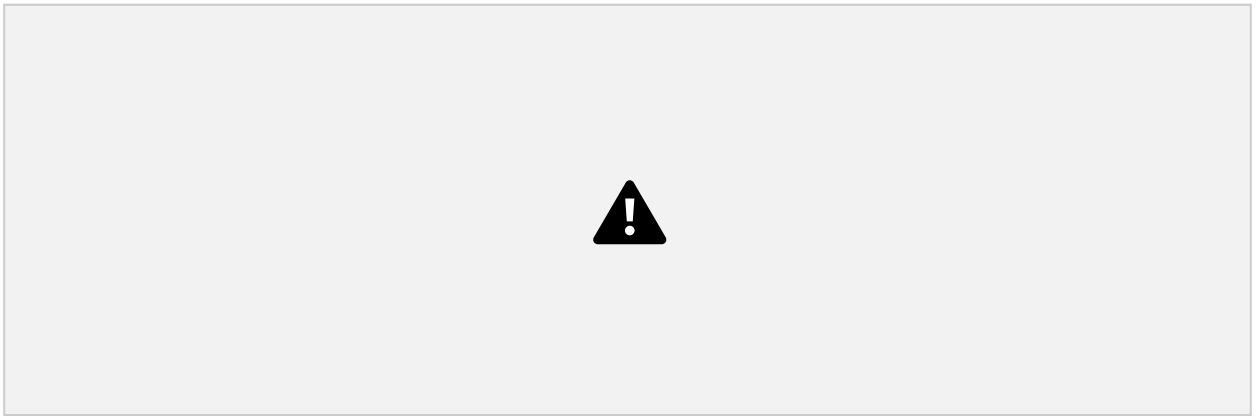
Now select New -> PythonX and enter the below lines and select Run. On Jupyter, each cell is a statement, so you can run each cell independently when there are no dependencies on previous cells.

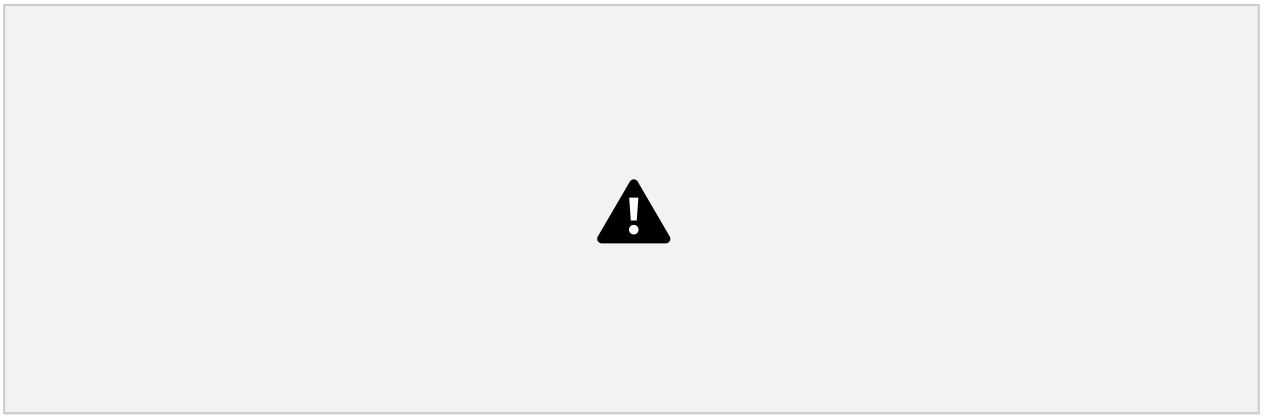


This completes installing Anaconda and running Jupyter Notebook.









PERFORMANCE	30	
OBSERVATION	30	
RECORD	40	
TOTAL	100	

RESULT:

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

Ex-No: 5	Working on NumPy Packages.

Aim

To implement array object using Numpy module in Python

programming **Algorithm**

Step 1: Start the program

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-in methods 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 program

(i) Create a NumPy ndarray Object Program

```
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'>
```

(ii) Dimensions in Arrays 0-D Arrays

Program

```
import numpy as np

arr = np.array(42)

print(arr)
```

Output

42

1-D Arrays Program

```
import numpy as np
```



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

print(arr)
```

Output

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

2-D Arrays Program

```
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 Program

```
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]]]
```

(iii) Check Number of Dimensions?

Program

```
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
```

(iv) Access Array Elements

Program import numpy as np

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

print(arr[0])

Output

1

Program

import numpy as np

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

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

Output

7

(v) Slicing arrays

import numpy as np

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

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

Output

[2 3 4 5]

(vi) NumPy Array Shape

Program import numpy as np

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

8]]) print(arr.shape)

Output

(2, 4)

(vii) Reshaping arrays Program

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]]

(viii) Iterating Arrays Program

```
import numpy as np
arr = np.array([1, 2, 3])
for x in arr:
  print(x)
```

Output

1

2

3

(ix) Joining NumPy Arrays Program

```
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]

(x) Splitting NumPy Arrays Program

```
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])]

(xi) Searching Arrays Program


```
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]),)

(xii) Sorting Arrays

Program import numpy as np

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

print(np.sort(arr))

Output

[0 1 2 3]

PERFORMANCE	30	
OBSERVATION	30	
RECORD	40	
TOTAL	100	

Result:
The implementation of array object using Numpy module in Python programming was successfully verified.

Ex-No: 6	**Working on Pandas Packages.**

Aim:

To work with DataFrame object using Pandas module in Python

Programming. **Algorithm:**

Step 1: Start the program

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 program

(i) Create a simple Pandas DataFrame:

Program

```
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

(ii)Locate Row Program

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


Output

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

Note: This example returns a Pandas Series.

(iii)use a list of indexes:

Program

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

Output

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

Note: When using [], the result is a Pandas DataFrame.

(iv) Named IndexesProgram

```
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

```
print(df.loc["day2"])
```

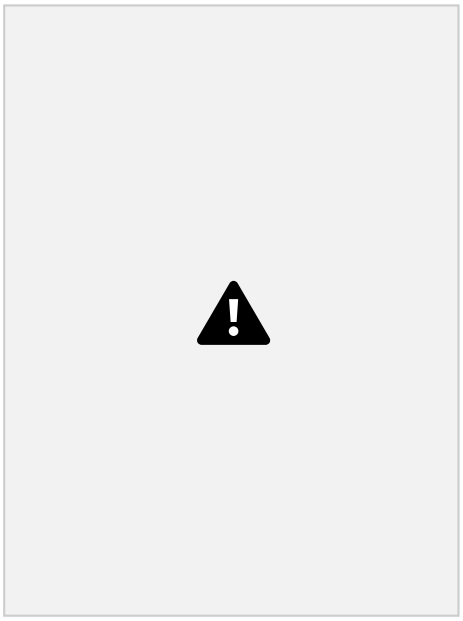
Output

```
calories 380
duration 40
Name: 0, dtype: int64
```


(vi) Load Files Into a DataFrameProgram

```
import pandas as pd
df = pd.read_csv('data.csv')
print(df)
```

Output



(vii) Check the number of maximum returned rows:

Program

```
import pandas as pd
print(pd.options.display.max_rows)
```

In my system the number is 60, which meansthat if the DataFrame contains more than 60 rows, the print(df)statement will return only the headers and the first andlast 5 rows.

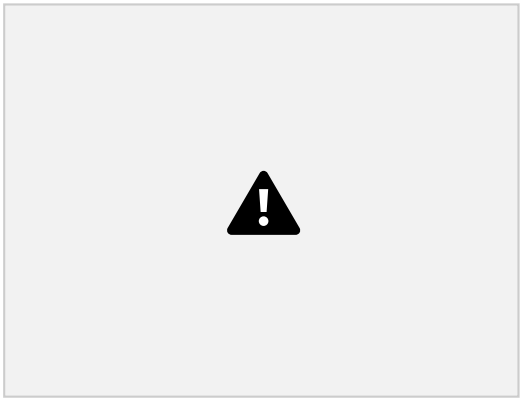
```
import pandas as pd
pd.options.display.max_rows = 9999
df = pd.read_csv('data.csv')
print(df)
```


(vii) Viewing the Data

Program

```
import pandas as pd
df = pd.read_csv('data.csv')
print(df.head(4))
```

Output



(viii) Print the last 5 rows of the DataFrame: 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

PERFORMANCE	30	
OBSERVATION	30	
RECORD	40	
TOTAL	100	

RESULT:

Thus, data frame object using Pandas module in Python Programming has been successfully explored.

Ex-No: 7	The following tasks can be done using the real-time data set from Kaggle

The following tasks can be done using the real-time data set fromKaggle a. Univariate analysis: Frequency, Mean, Median, Mode,Variance, Standard Deviation, Skewness and Kurtosis.
b. Bivariate analysis: Linear and logistic regression modeling.
c. Multiple Regression analysis.

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

Aim:

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

Algorithm:

Step 1: Start the program

Step 2: Import the required packages

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

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

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

Program:

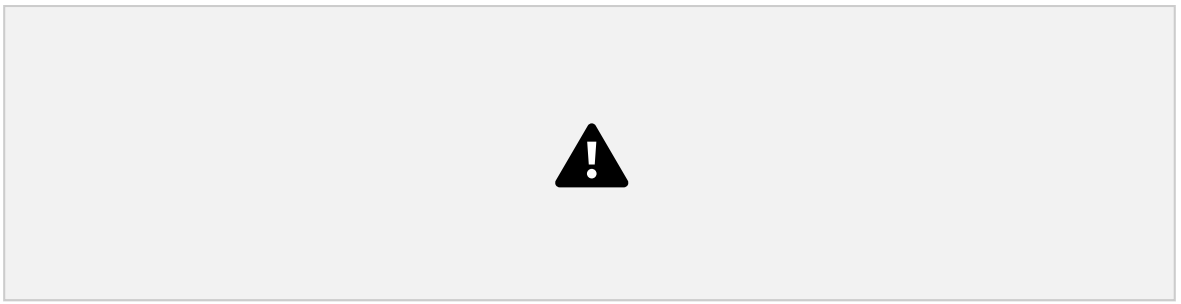
```
import pandas as pd import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn.linear_model import LogisticRegression

from sklearn.externals import joblib

df = pd.read_csv('C:/Users/praveen/Downloads/diabetes.csv')
```

```
count = df['Glucose'].value_counts()
display(count)
```

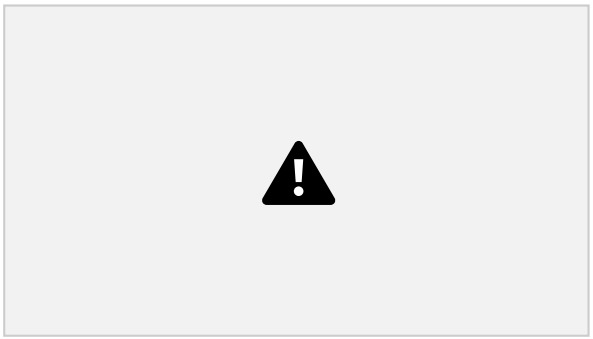
30

```
df.head()
```

```
df.describe()
```



```
df.mean()
```

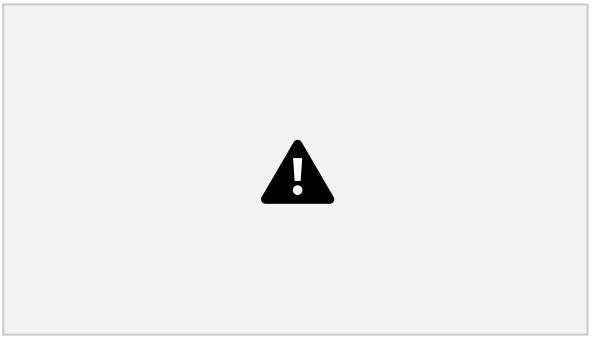


31

```
df.mode()
```

df.var()

df.std()



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

df.kurtosis()

Pregnancies 0.159220

Glucose 0.640780

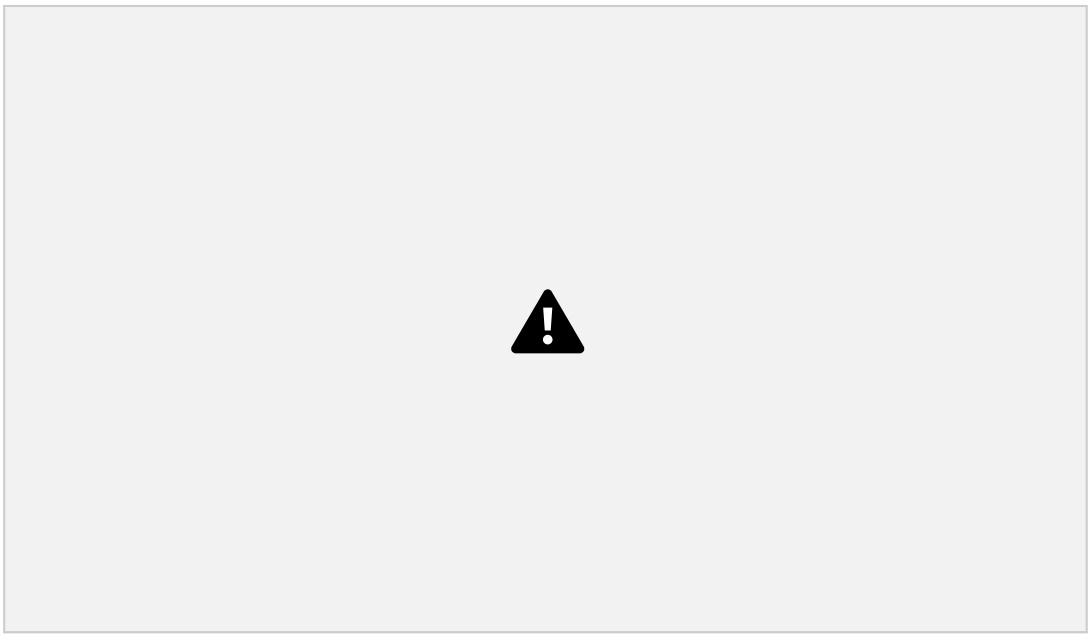
BloodPressure 5.180157

SkinThickness -0.520072

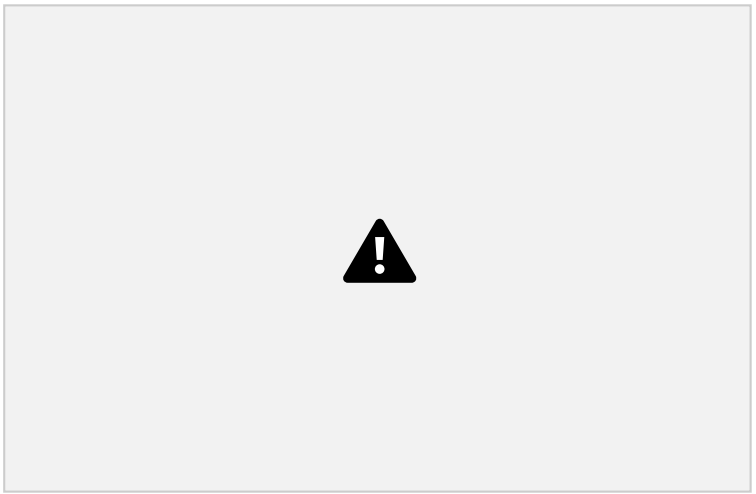
DiabetesPedigreeFunction 5.594954

Age 0.643159
Outcome -1.600930dtype: float64

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

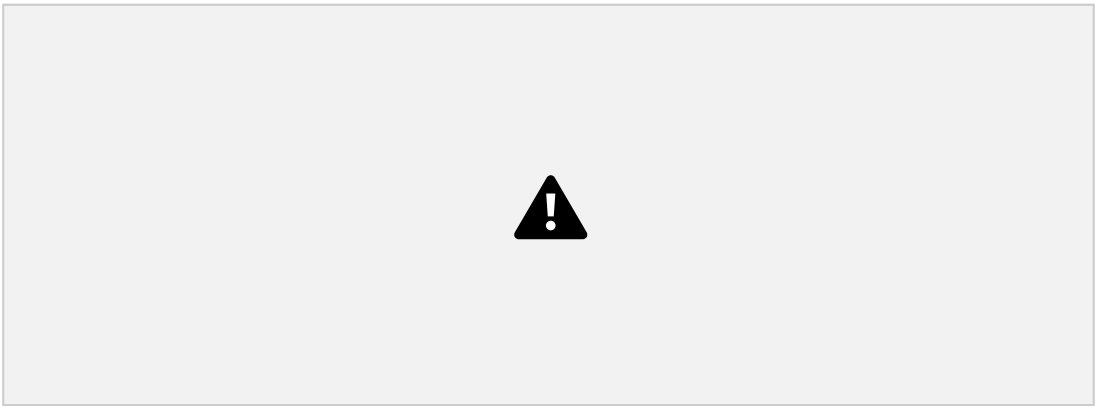


```
sns.countplot('Outco
me', data=df)
plt.show()
```




```
# Computing the %age of diabetic and non-diabetic in the
sample Out0=len([df.Outcome==1])
Out1=len([df.Outcom
e==0])
```

```
Total=Out0+Out1
PC_of_1 = Out1*100/Total
PC_of_0 =
Out0*100/T
otal
PC_of_1,
PC_of_0
(50.0, 50.0)
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()
```



PERFORMANCE	30	
OBSERVATION	30	
RECORD	40	
TOTAL	100	

RESULT:

Thus, various exploratory data analysis has been performed on Pima Indians Diabetes dataset using Python Programming successfully.

Ex-No: 8	Explore and apply various plotting functions to Kaggle real-time data sets

Explore and apply various plotting functions to Kaggle real-time datasets a. Normal curves.
b. Density and contour plots.
c. Correlation and scatter plots.
d. Histograms.

Three-dmensional plotting

Aim:

To apply various plotting functions on UCI data set using Python Programming

Algorithm:

Step 1: Start the program

Step 2: Import the required packages

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

UCI data setStep 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 for the following

a. Normal curves

b. Density and contour plots

c. Correlation and scatter plots

d. Histograms

e. Three-dimensional plotting

Step 7: Analyse the sample data and do the required

operationsStep 8: Stop the program

a. Normal curves.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
df=pd.read_csv("C:/Users/praveen/Downloads/dataset_diabetes/diabetic_data.csv")
```

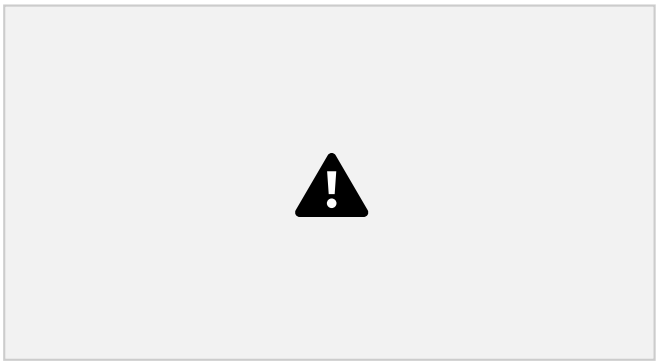


```
df.head()
mean
=df['time_in_hospital'].mean()
std =df['time_in_hospital'].std()
x_axis = np.arange(1, 10, 0.01)
plt.plot(x_axis,
```

norm.pdf(x_axis, mean, std))

plt.show()

Output



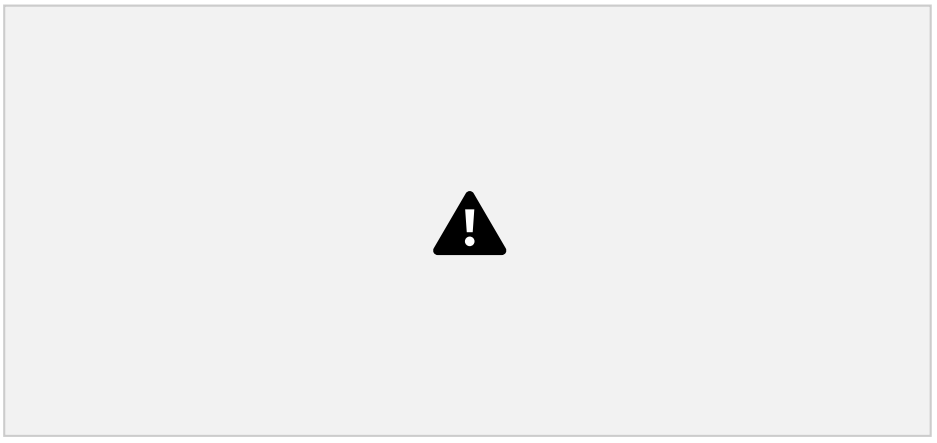
(b) Density and contourplots

Program

df.time_in_hospital.plot.density(color='green')

plt.title('Density plot for time_in_hospital')

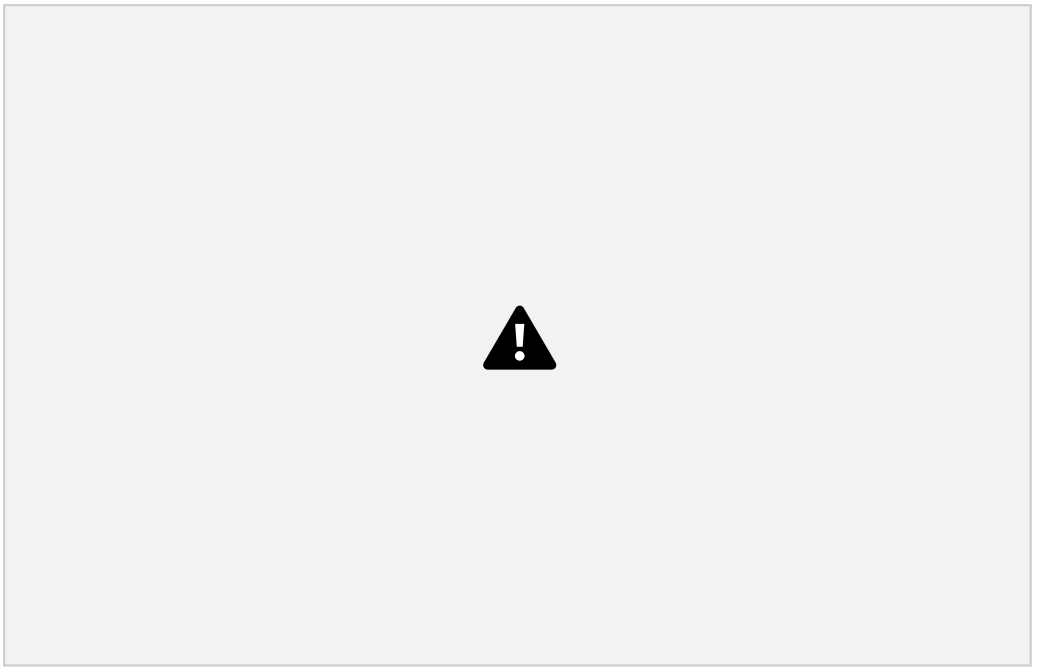
plt.show()



df.num_lab_procedures.plot.density(color='green')

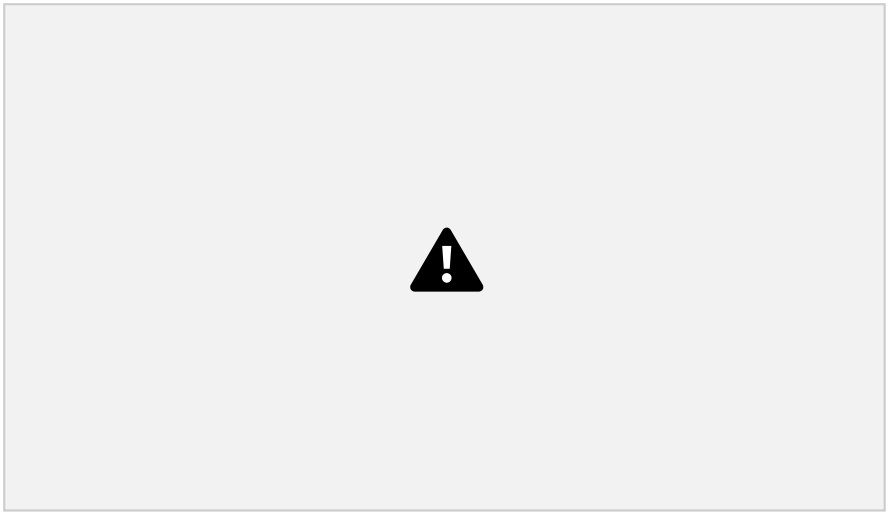
plt.title('Density Plot for num_lab_procedures')

plt.show()



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

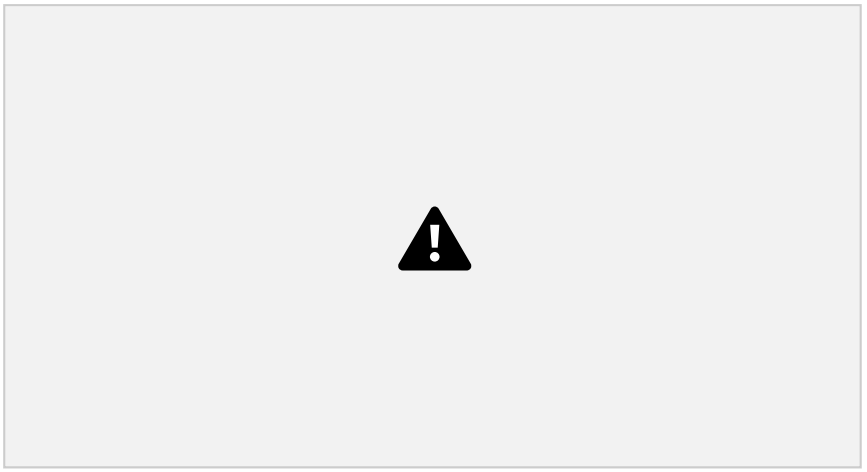
Output



Program

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

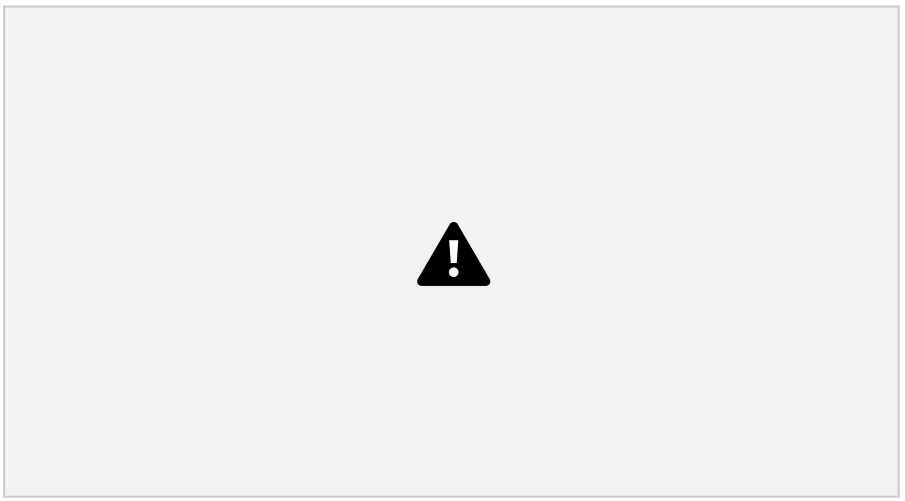
Output



Program:

```
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

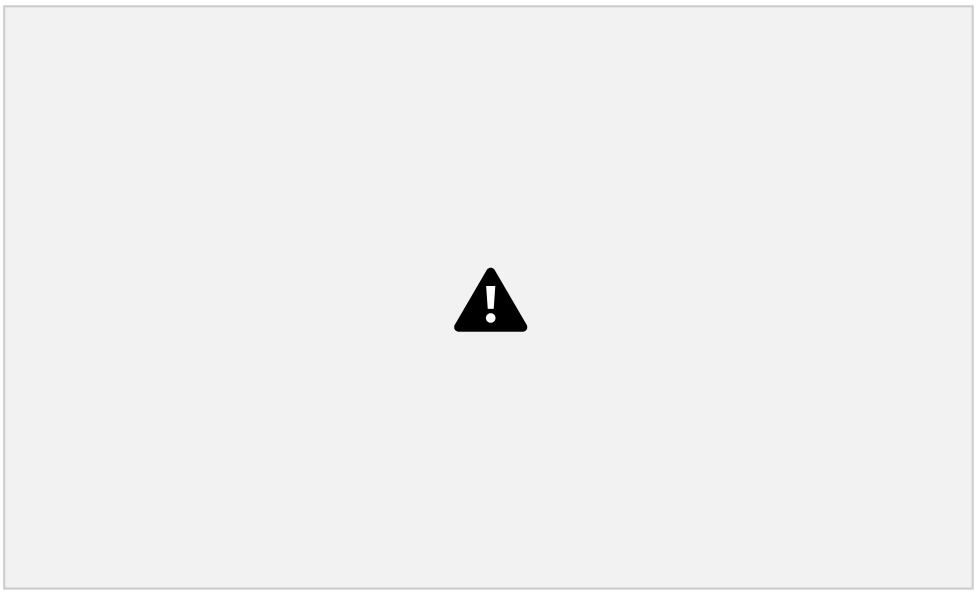


(c) Correlation and scatter plots

Program

```
mp.figure(figsize=(20,10))
dataplot = sb.heatmap(data.corr(), cmap="YlGnBu", annot=True)
```

Output

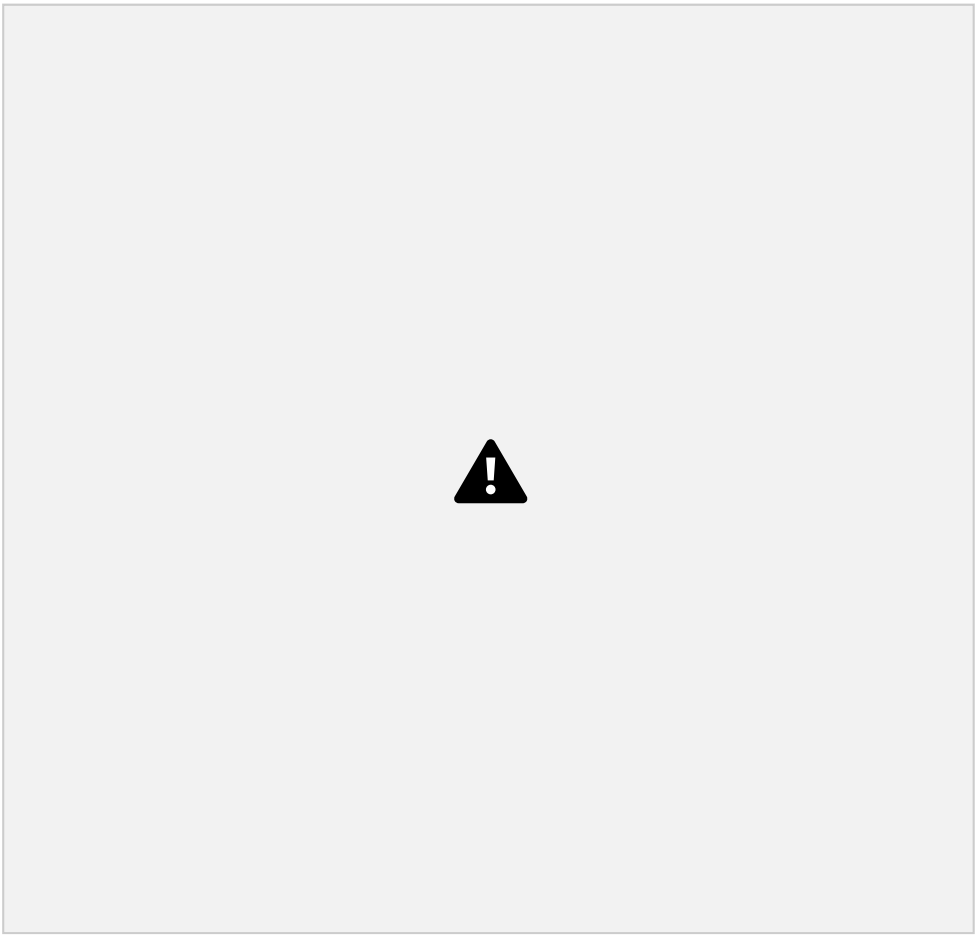


(d) Histograms

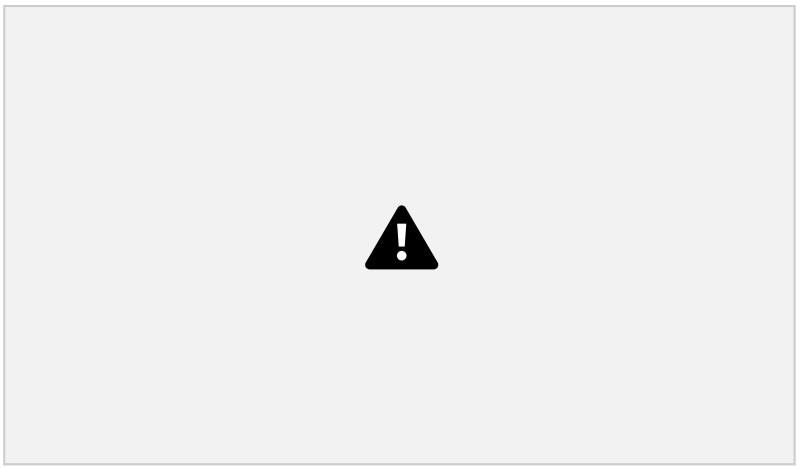
Program

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


Output



```
# plotting histogram for carat using distplot()
sb.distplot(a=df.num_lab_procedures, kde=False)

# visualizing plot using matplotlib.pyplot library

plt.show()
```



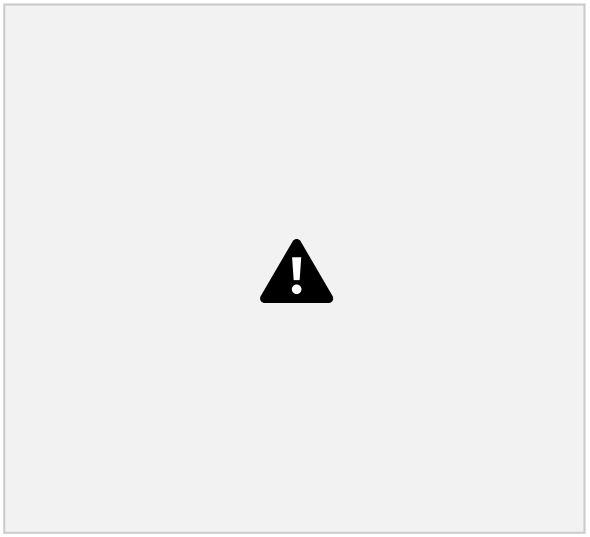
Output

(e) Three-dimensional plotting

Program

```
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



PERFORMANCE	30	
OBSERVATION	30	
RECORD	40	
TOTAL	100	

RESULT:

Thus, apply various plotting functions on UCI data set using Python