# PROGRAM . 1

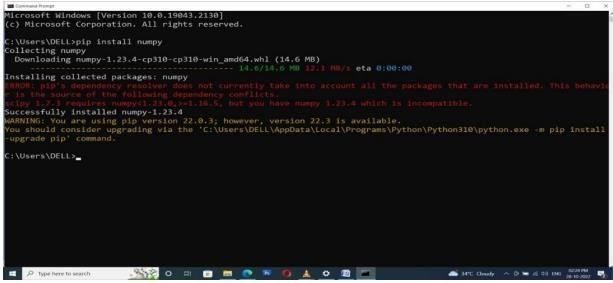
Installing Method

## Numpy

Numpy is a numerical computing package for mathematics, science, and engineering.

Many data science packages use Numpy as a dependency. Command: pip install numpy

## Output:

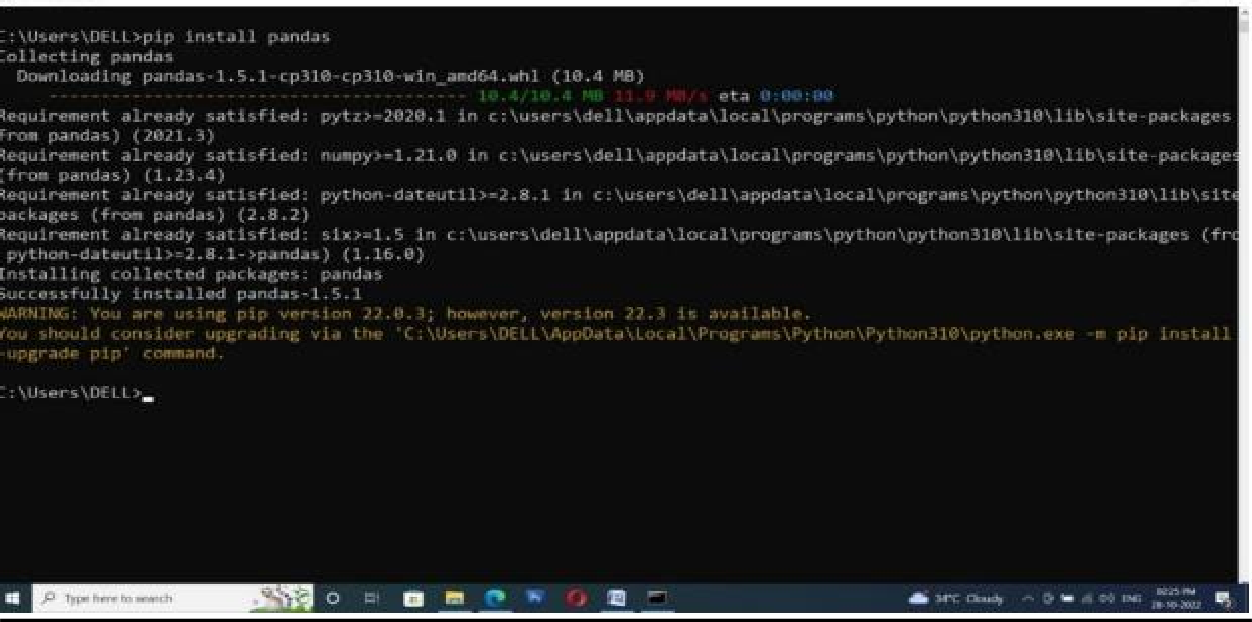


1. **Pandas**

Pandas visualizes and manipulates datatables.There are many functions that allow efficient manipulation for the preliminary steps of data analysis problems.

Command: pip install pandas

## Output:

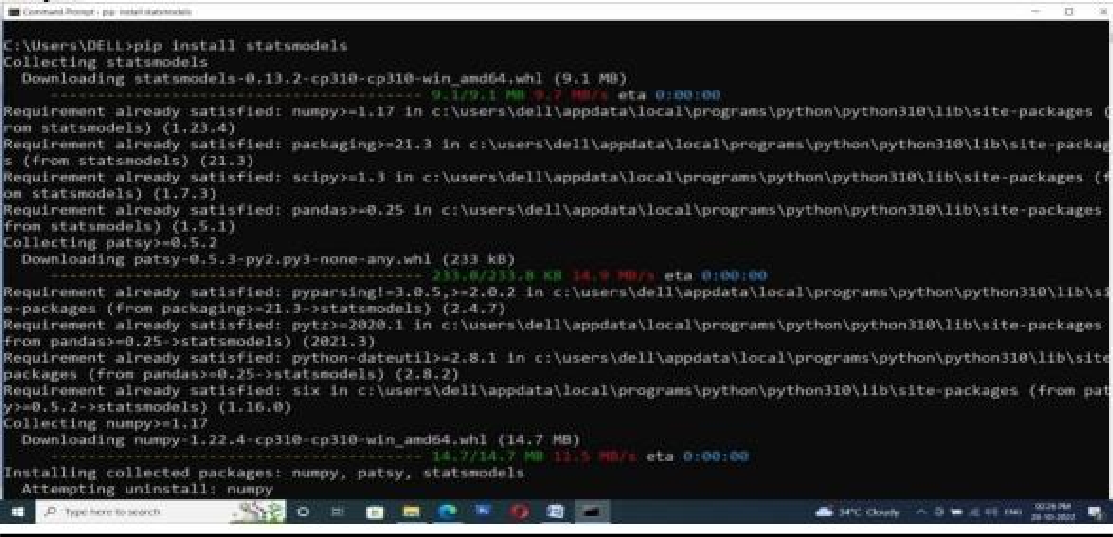


1. **Statsmodel**

Statsmodels is a package for exploring data, estimating statistical models, and performing statistical tests. It include descriptive statistics, statistical tests, plotting functions, and result statistics.

Command: pip install statsmodels

## Output:

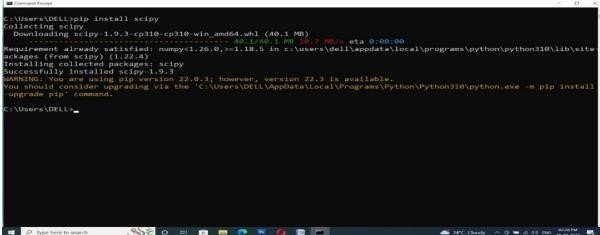


1. **SciPy:**

SciPy is a general-purpose package for mathematics, science, and engineering and extends the base capabilities of NumPy.

Command: pip install scipy

## Output:

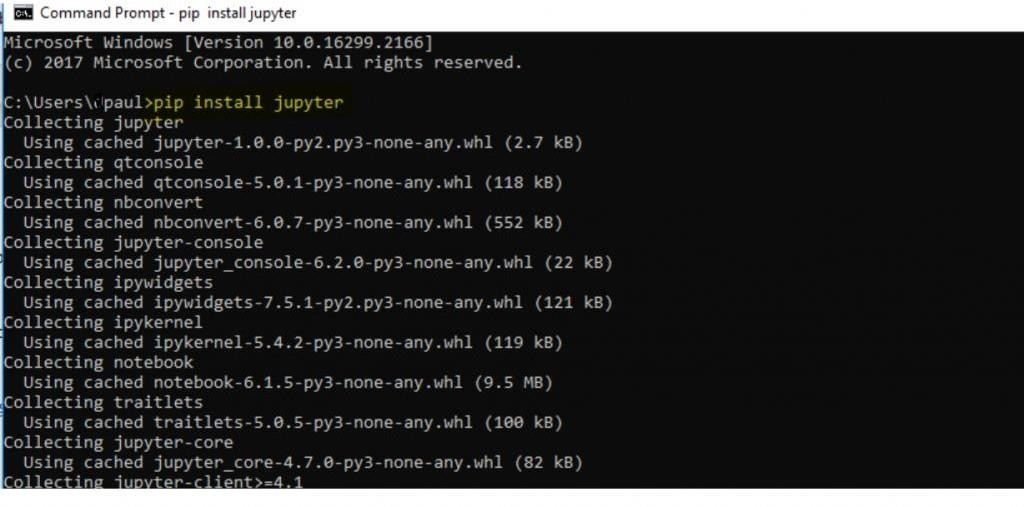


1. **Jupyter**

It is used to create interactive notebook documents that can contain live code, equations, visualizations, media and other computational outputs. Jupyter Notebook is often used by programmers, data scientists and students to document and demonstrate coding workflows or simply experiment with code.

Command: pip install jupyter

**Output:**



**Program – 2**

import numpy as np

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

print (" OUTPUT IS ")

print("ONE DIMENSIONAL ARRAY IS:\n",a)

b=np.array([[10,20],[40,50]])

print (" TWO DIMENSIONAL ARRAY IS:\n",b)

c=np.array([[10,20,30],[40,50,60],[70,80,90]])

print (" THREE DIMENSIONAL ARRAY IS:\n", c)

#array indexing and numpy array atributes

x1 = np.array([45,89,64,33])

x2 = np.array([[1,2,3],[4,5,6],[7,8,9]])

print("Array indexing and numpy array atributes")

print(x1)

print("x2 ndim:",x2.ndim)

print("x2 shape:",x2.shape)

print("x2 size:",x2.size)

#accessing single elements

print("accessing single elements")

print(x1[-2])

print(x2)

x2[1,2] = 60

print(x2)

#array slicing

print("array slicing")

x = np.arange(10)

print(x[5:9])

print(x[:5])

print(x[1::3])

print(x2)

print(x2[0:2,0:2])

#concatenation of arrays

print("concatenation of arrays")

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

q = np.array([8,12,14])

print(np.concatenate([p,q]))

#splitting of array

print("splitting of array")

y = [1,2,3,99,99,3,2,1]

d,e = np.split(x,[2])

print(d,e)

#Functions

print("Function:")

print("add")

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

print(np.add(t,2))

print("Subtract")

print(np.subtract(t,2))

print("negative")

print(np.negative(t))

print("multiply")

print(np.multiply(t,2))

print("divide")

print(np.divide(t,5))

print("floor divide")

print(np.floor\_divide(t,2))

print("power")

print(np.power(t,2))

print("modulus")

print(np.mod(t,2))

#absloute

print("Absolute")

print(np.absolute([-1,-2,0,1,2]))

#Trigonometric

print("trignometric")

theta = [0,1.57,3.14]

print(np.sin(theta))

print(np.cos(theta))

print(np.tan(theta))

#exponent

print("Exponent")

print(np.exp(x))

print(np.exp2(x))

print(np.power(x,3))

print("Log")

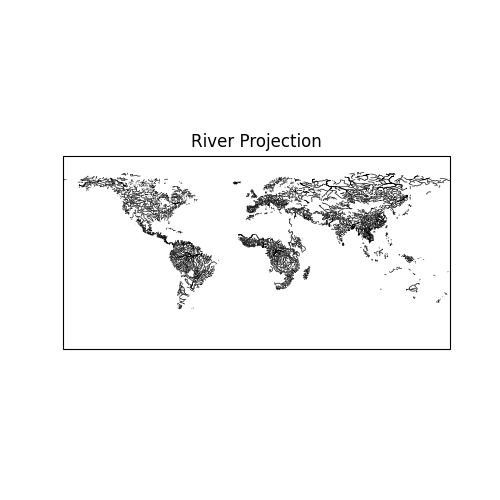
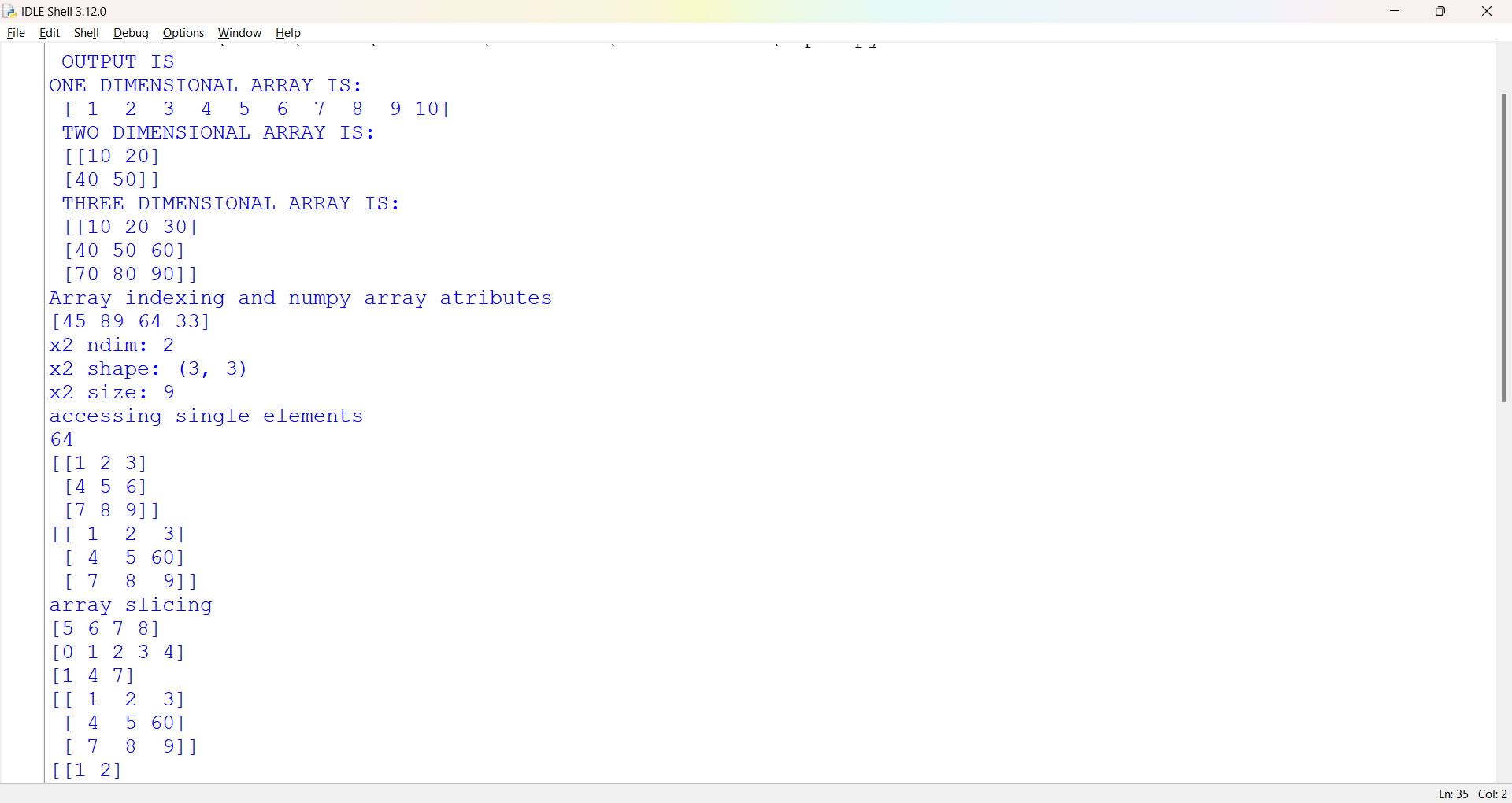
o = [1,10,100]

print(np.log(o))

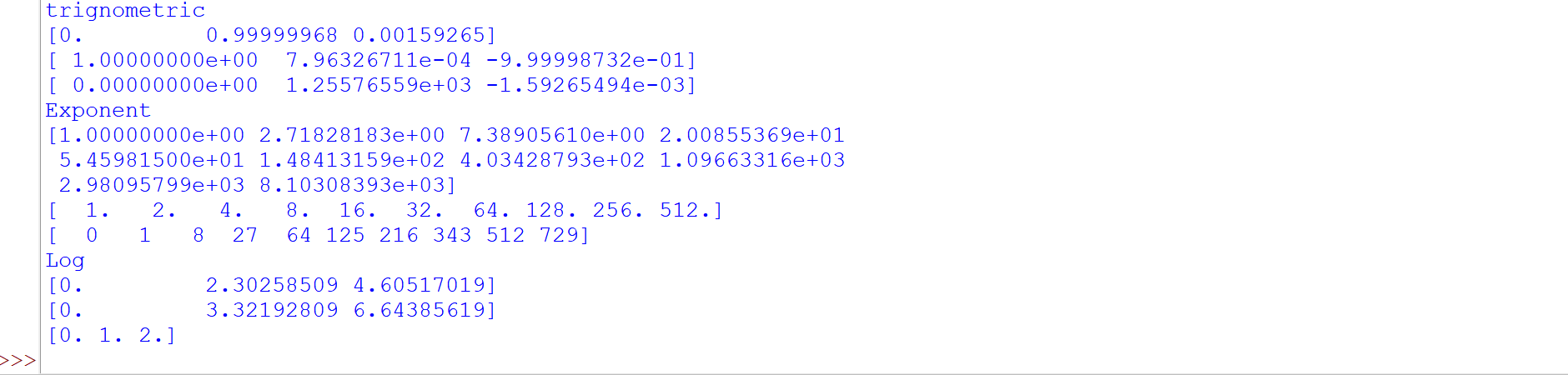
print(np.log2(o))

print(np.log10(o))

**Output**

****

****

****

**Program – 3**

import pandas as pd

data = pd.Series([1,6,11,777])

data[0]

data[1:4]

data = pd.Series([10,20,30,40],index = ['a','b','c','d'])

print(data)

print(data.values)

print(data.index)

students\_dict = {'Ramu':991,'Shyam':992,'Arun':993}

students = pd.Series(students\_dict)

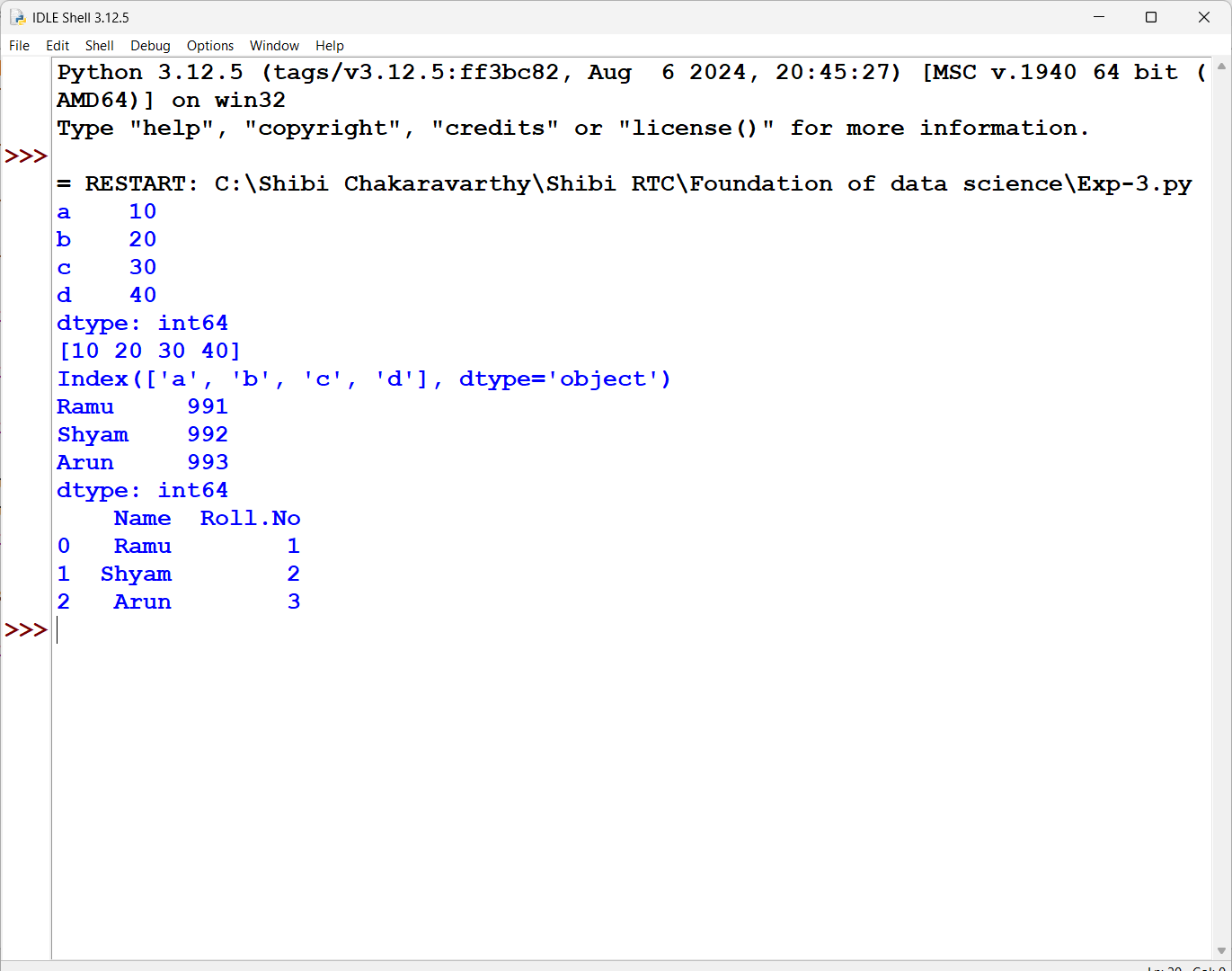
print(students)

list1 = [['Ramu',1],['Shyam',2],['Arun',3]]

df = pd.DataFrame(list1,columns = ['Name','Roll.No'])

print(df)

**Output**

****

**Program – 4**

import pandas as pd

import matplotlib.pyplot as pl

df=pd.read\_csv("Iris\_Data1.csv")

print(df)

print()

print("Descriptive analysis on Iris Data: ")

print("Info:\n")

print(df.info)

print("Describe:\n")

print(df.describe)

print("Head:\n")

print(df.head)

print("Species count:\n")

print(df['CLASS'].value\_counts())

print("IsNull:\n")

print(df.isnull)

print("Max:\n")

print(df.max)

print("Shape:\n")

print(df.shape)

print("Size:\n")

print(df.size)

#plotting graph

x = df['ID'].head(10)

y = df["SL"].head(10)

print(pl.title("Bar Graph - ID vs Sepal Length"))

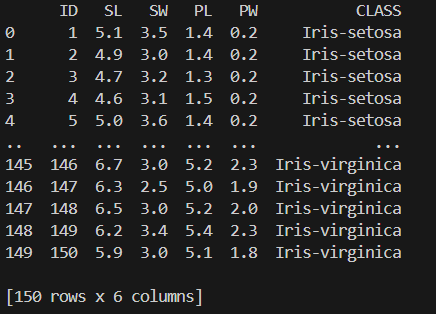
print(pl.plot(x,y,marker = "\*",linestyle = "dashed"))

pl.xlabel("ID")

pl.ylabel("Sepal length")

pl.show()

**Output:**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

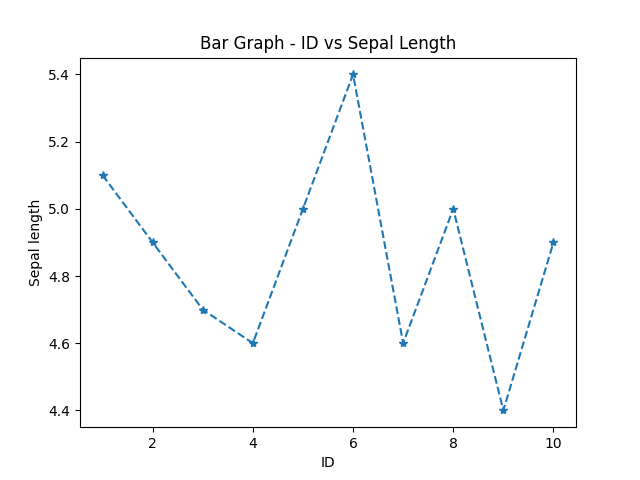
**A screenshot of a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generatedA screen shot of a computer

Description automatically generatedA screenshot of a computer program

Description automatically generated**



**Program – 5**

**5.1**

**Perquisite - pip install scikit-learn**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LogisticRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

import statsmodels.api as sm

df=pd.read\_csv("diabetes.csv")

print(df)

print("\nFrequency of Pregnancies:\n")

print(df["Pregnancies"].value\_counts())

print("\nFrequency of Glucose:\n")

print(df["Glucose"].value\_counts())

print("\nFrequency of BloodPressure:\n")

print(df["BloodPressure"].value\_counts())

print("\nFrequency of SkinThickness:\n")

print(df["SkinThickness"].value\_counts())

print("\nFrequency of Insulin:\n")

print(df["Insulin"].value\_counts())

print("\nFrequency of BMI:\n")

print(df["BMI"].value\_counts())

print("\nFrequency of DiabetesPedigreeFunction:\n")

print(df["DiabetesPedigreeFunction"].value\_counts())

print("\nFrequency of Age:\n")

print(df["Age"].value\_counts())

print("\nFrequency of Outcome:\n")

print(df["Outcome"].value\_counts())

print("Mean, Median, Mode, Standard deviation, skewness and Kurtosis\n")

print("Mean of Pregnancies:",df["Pregnancies"].mean())

print("Median of Pregnancies:",df['Pregnancies'].median())

print("Mode of Pregnancies:",df["Pregnancies"].mode())

print("Standard Deviation of Pregnancies:",df["Pregnancies"].std())

print("Skewness of Pregnancies:",df["Pregnancies"].skew())

print("Kurtosis of Pregnancies:",df["Pregnancies"].kurt())

print("\nBivariate Analysis : linear and logistic regression modelling:\n")

x = df['Age']

y = df['BMI']

n = np.size(x)

x\_mean = np.mean(x)

y\_mean = np.mean(y)

x\_mean,y\_mean

Sxy = np.sum(x\*y)- n\*x\_mean\*y\_mean

Sxx = np.sum(x\*x)-n\*x\_mean\*x\_mean

b1 = Sxy/Sxx

b0 = y\_mean-b1\*x\_mean

print('slope b1 is', b1)

print('intercept b0 is', b0)

y\_pred = b1 \* x + b0

print("Linear Regression :",y\_pred.mean())

#LOGISTIC REgression

X = df[['Age', 'Pregnancies']]

y = df['Outcome']

# Splitting the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Creating a logistic regression model

model = LogisticRegression()

model.fit(X\_train\_scaled, y\_train)

# Making predictions on the test data

y\_pred = model.predict(X\_test\_scaled)

# Evaluating the model

accuracy = model.score(X\_test\_scaled, y\_test)

print("\nLogistic Regression:\n")

print(f"Accuracy: {accuracy:.2f}")

print("\nMultiple Regression:\n")

model1 = sm.OLS.from\_formula(' Pregnancies ~ Age+ Outcome ', df).fit()

print(model1.summary())

**5.2**

df1=pd.read\_csv("diabetes\_data\_upload.csv")

print(df1)

print()

#Univariate analysis

#---> frequency of age

print("\nFrequency of Age:")

print(df1["Age"].value\_counts())

#---> frequency of gender

print("\nFrequency of Gender:")

print(df1["Gender"].value\_counts())

#---> frequencey of polyuria

print("\nFrequency of Polyuria:")

print(df1["Polyuria"].value\_counts())

#---> frequency of delayed healing

print("\nFrequency of delayed healing:")

print(df1["delayed healing"].value\_counts)

#--->frequency of class

print("\nFrequency of class:")

print(df1["class"].value\_counts())

#---> Mean,median,mode,standard deviation,skewness,kurtosis

print("\nMean,median,mode,standard deviation,skewness,kurtosis:")

print("Mean of Age:",df1["Age"].mean())

print("Median of Age:",df1['Age'].median())

print("Mode of Age:",df1["Age"].mode())

print("Standard Deviation of Age:",df1["Age"].std())

print("Skewness of Age:",df1["Age"].skew())

print("Kurtosis of Age:",df1["Age"].kurt())

#---> Birvariate analysis:Linear and Logistic regression modelling

print("\nLinear Regression:")

x = df1['Age']

y = df1['Polyuria']

n = np.size(x)

x\_mean = np.mean(x)

y\_mean = np.mean(y)

x\_mean,y\_mean

Sxy = np.sum(x\*y)- n\*x\_mean\*y\_mean

Sxx = np.sum(x\*x)-n\*x\_mean\*x\_mean

b1 = Sxy/Sxx

b0 = y\_mean-b1\*x\_mean

print('slope b1 is:', b1)

print('intercept b0 is:', b0)

y\_pred = b1 \* x + b0

print("Linear Regression :",y\_pred.mean())

print("\nLogistic regression:")

X = df1[['Age', 'weakness']]

y = df1['Polyuria']

# Splitting the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Creating a logistic regression model

model = LogisticRegression()

model.fit(X\_train\_scaled, y\_train)

# Making predictions on the test data

y\_pred = model.predict(X\_test\_scaled)

# Evaluating the model

accuracy = model.score(X\_test\_scaled, y\_test)

print(f"Accuracy: {accuracy:.2f}")

print("\nMultiple Regression:")

model2 = sm.OLS.from\_formula('Age ~ Polyuria + weakness + Polydipsia', df1).fit()

print(model2.summary())

#---> Comparing all

print("\nPima Dataset:")

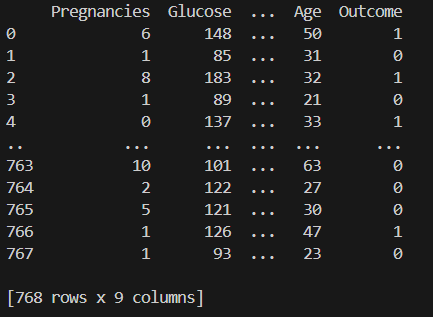
print(model1.params)

print("\nUCI Dataset:")

print(model2.params)

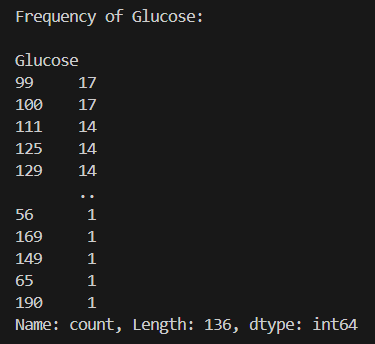
**Output:**

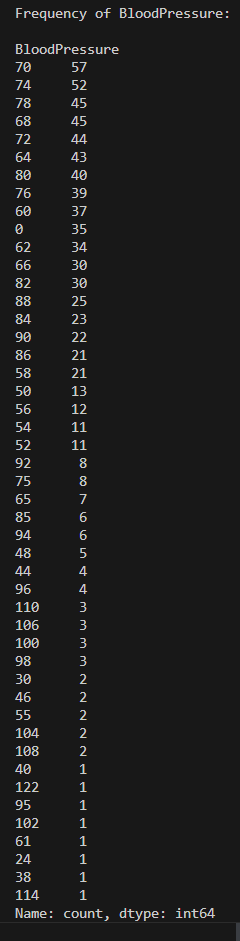
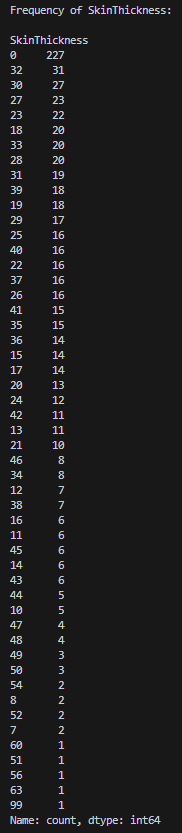
**5.1**

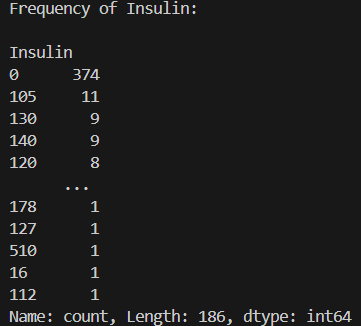
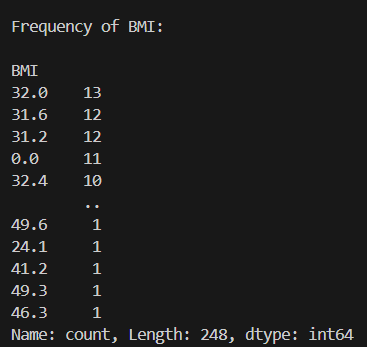
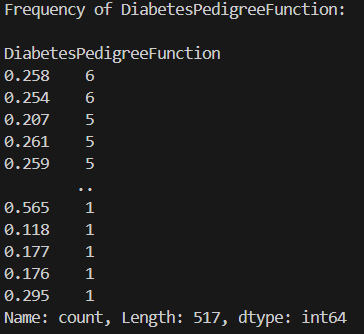


A screenshot of a computer screen

Description automatically generated

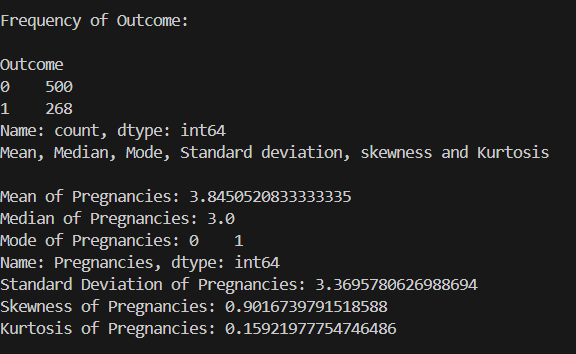


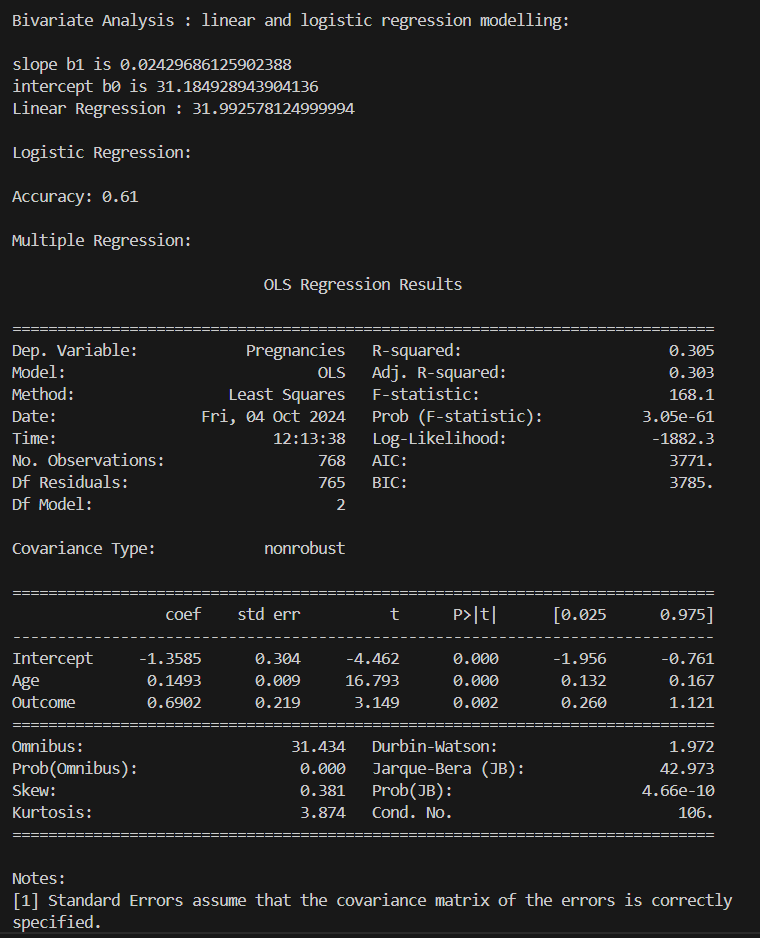
 

A screenshot of a computer

Description automatically generated

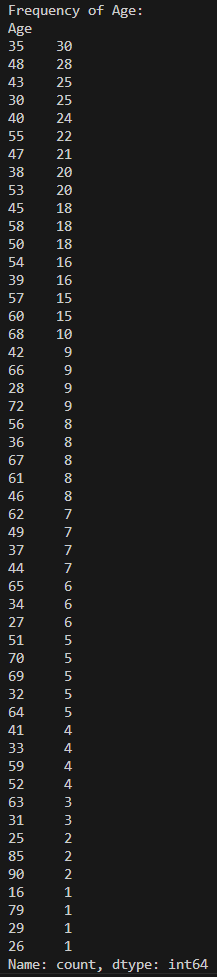


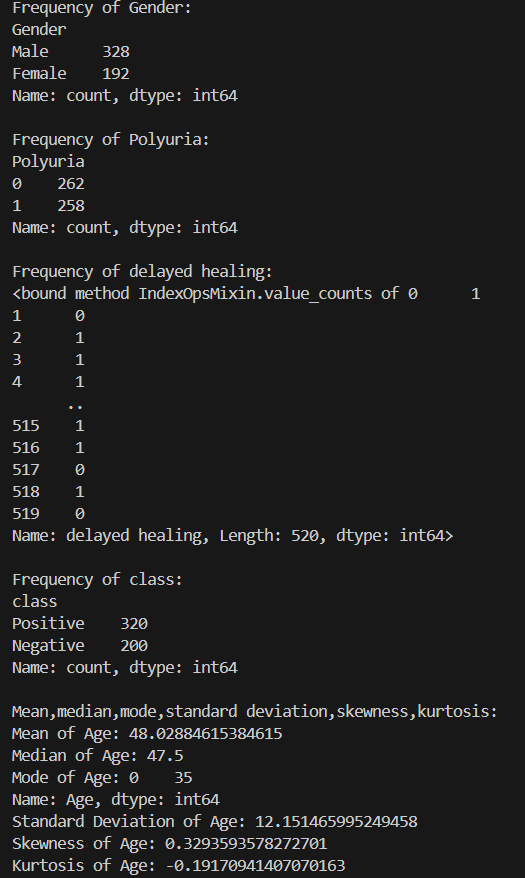


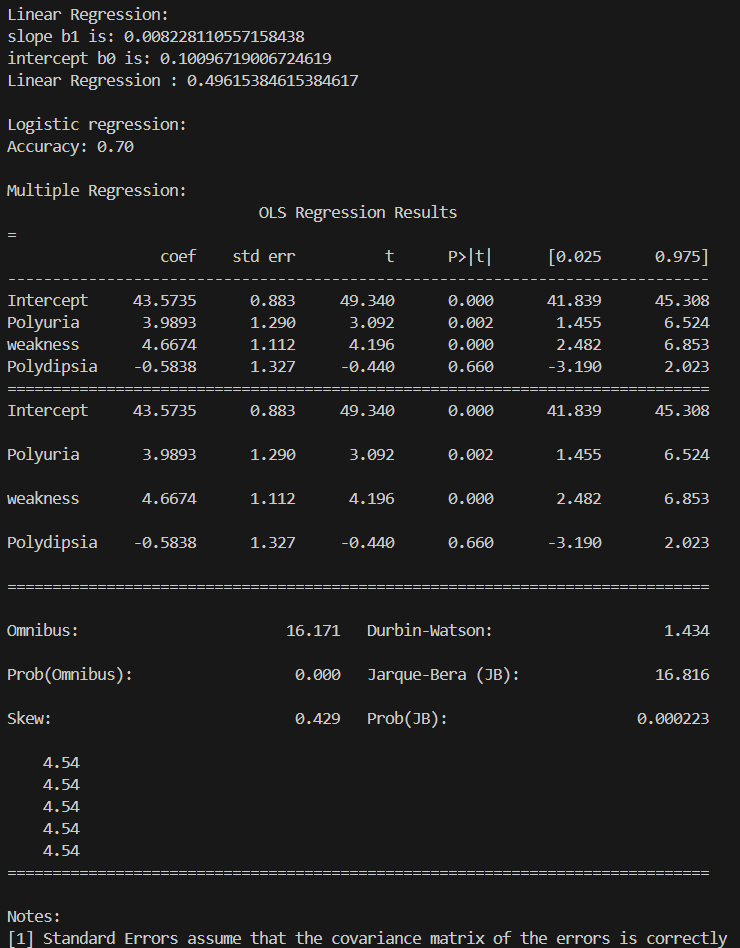
**5.2**

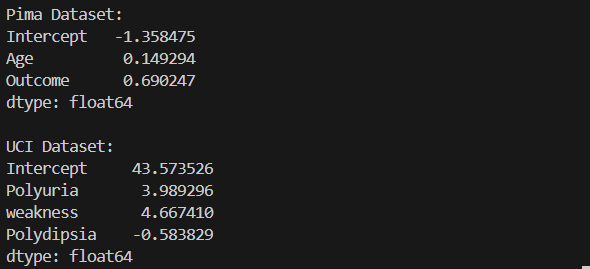
A screenshot of a computer screen

Description automatically generated









**Program – 6**

import matplotlib

import pandas as pd

import numpy as np

from scipy.stats import norm

import seaborn as sns

import matplotlib.pyplot as pl

df=pd.read\_csv("diabetes\_data\_upload.csv")

print(df)

#normal curve plot

data=df['Polyuria']

mean=data.mean()

std\_dev=data.std()

random=np.random.normal(mean,std\_dev,len(data))

xmin,xmax=pl.xlim()

x=np.linspace(xmin,xmax,100)

p=norm.pdf(x,mean,std\_dev)

pl.title("Normal Curve")

pl.plot(x,p,"g",linewidth=2,label="Normal")

pl.show()

#Density and Contour plots

sns.kdeplot(data=df, x='Age', y='weakness', fill=True)

pl.title('Density and Contour Plot')

pl.xlabel('x')

pl.ylabel('y')

pl.show()

#correlation and scatter plot

x = df['Age']

y = df['Obesity']

sns.scatterplot(x)

corr = np.corrcoef(x, y)[0, 1]

pl.title('Scatter Plot with Correlation Coefficient')

pl.xlabel('Age')

pl.ylabel('Obesity')

pl.text(0.5, 0.5, 'Correlation Coefficient: {0:.2f}'.format(corr))

pl.show()

#Histogram plot

numbers = df['Age']

pl.hist(numbers, bins=10)

pl.title("Histogram")

pl.xlabel("Interval")

pl.ylabel("Age")

pl.show()

#3d plot 1

fig = pl.figure()

ax = pl.axes(projection='3d')

zline = np.linspace(0, 5, 20)

xline = df['Age'].head(20)

yline = df['Itching'].head(20)

ax.scatter3D(xline, yline, zline, 'greenmaps')

pl.show()

#3d plot 2

fig = pl.figure()

ax = pl.axes(projection='3d')

zline = np.linspace(0, 5, 20)

xline = df['Age'].head(20)

yline = df['Itching'].head(20)

ax.plot3D(xline, yline, zline)

pl.show()

#3d plot 3

def f(x, y):

    return np.sin(np.sqrt(x \*\* 2 + y \*\* 2))

x =df['Age'].head(10)

y = df['Obesity'].head(10)

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

Z = f(X, Y)

fig = pl.figure()

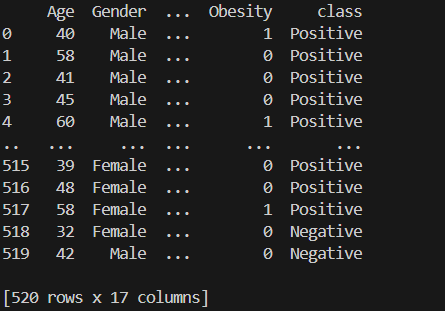
ax = pl.axes(projection ='3d')

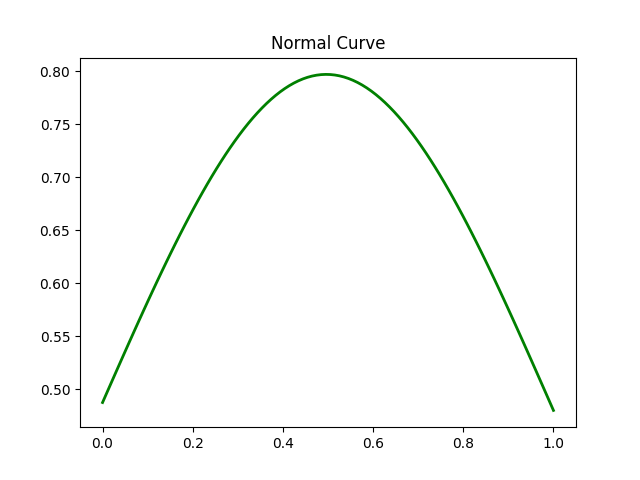
ax.plot\_wireframe(X, Y, Z, color ='green')

ax.set\_title('WireFrame for UCI Dataset')

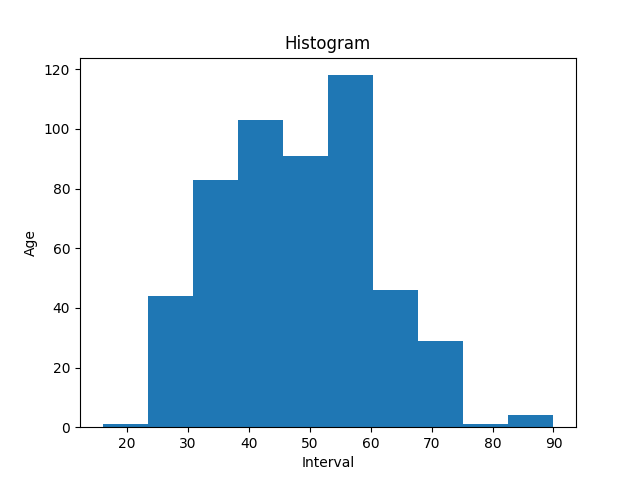
pl.show()

**Output**

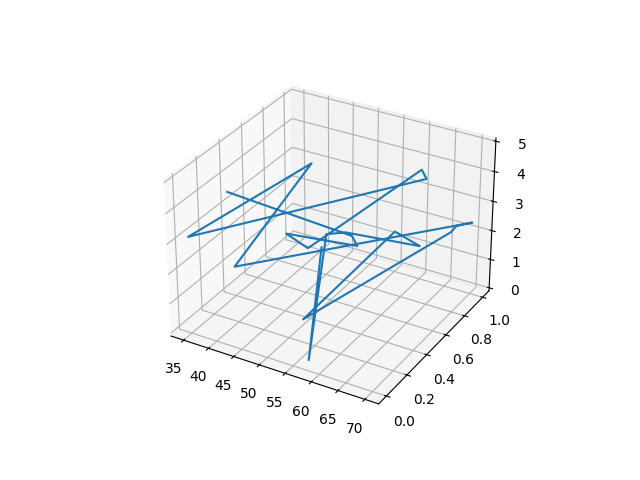
****

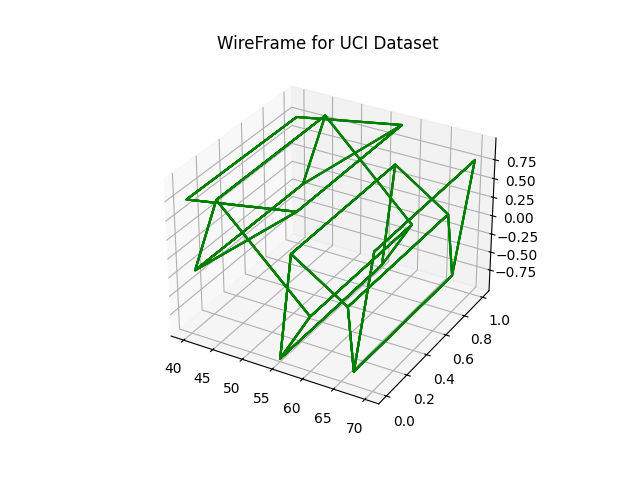
**A blue ovals with white text

Description automatically generatedA diagram of a scatter plot

Description automatically generated**

**A graph of a graph with blue dots

Description automatically generated with medium confidence**

****

**Program – 7**

import matplotlib.pyplot as plt

import mathplot

import cartopy.crs as ccrs

from cartopy.io import shapereader

from mpl\_toolkits.basemap import Basemap

#orthographic projection

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

projection=ccrs.Orthographic(central\_longitude=50,central\_latitude=0)

m=Basemap()

ax=plt.axes(projection=projection)

ax.stock\_img()

ax.set\_title('Orthographic Projection')

plt.show()

#robinson projection

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

projection=ccrs.Robinson()

m=Basemap()

ax=plt.axes(projection=projection)

ax.stock\_img()

ax.coastlines()

ax. set\_title('Robinson Projection')

plt.show()

#River projection

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

projection=ccrs.Orthographic(central\_longitude=77, central\_latitude=27)

m=Basemap()

ax=plt.axes(projection=projection)

m.drawrivers()

ax.coastlines()

ax. set\_title('River Projection')

plt.show()

#Blue Marble projection

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

projection=ccrs.Orthographic(central\_longitude=77,central\_latitude=27)

m=Basemap()

ax=plt.axes(projection=projection)

m.bluemarble()

ax.set\_title('Bluemarble Projection')

plt.show()

#Shaded relief projection

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

projection=ccrs. Orthographic(central\_longitude=77, central\_latitude=27)

m=Basemap()

ax=plt.axes(projection=projection)

m. shadedrelief()

ax.set\_title('Shaded Relief Projection')

plt.show()

#Countries boundary

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

projection=ccrs.Orthographic(central\_longitude=77,central\_latitude=27)

m=Basemap()

ax=plt.axes(projection=projection)

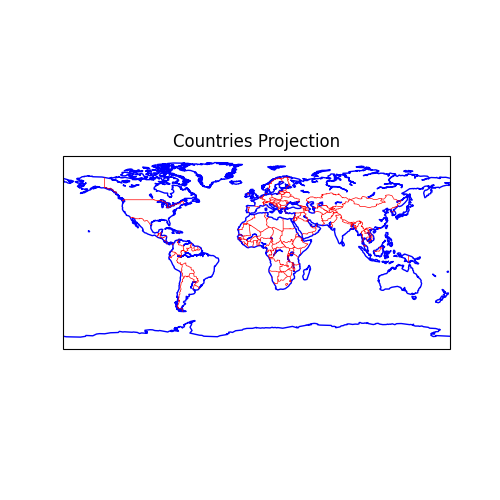
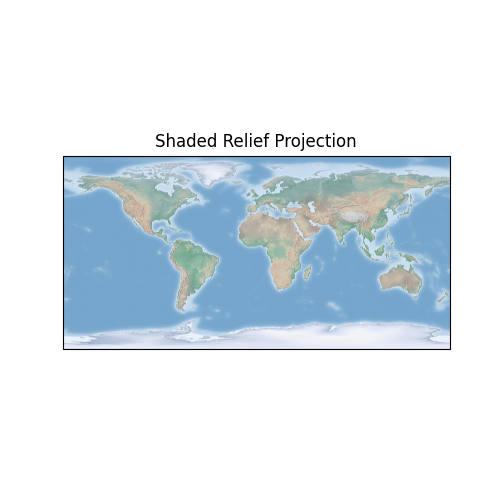
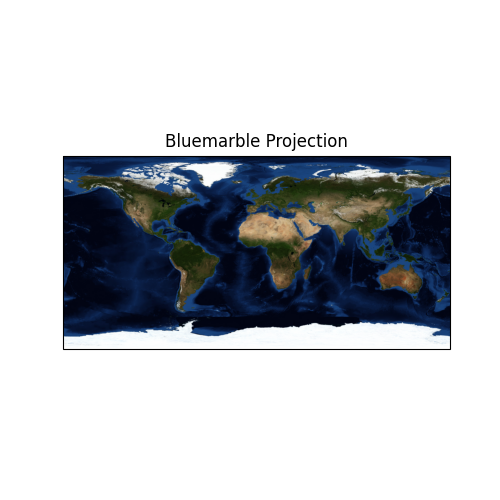
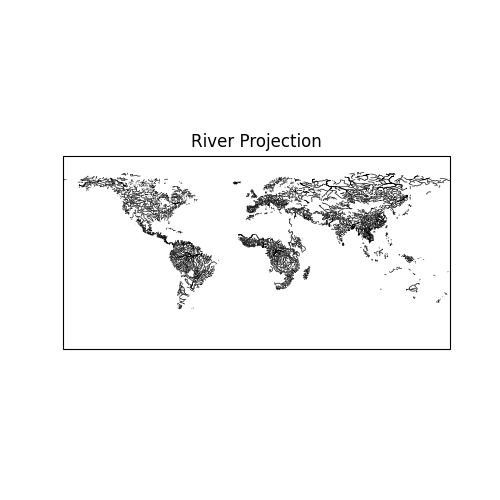
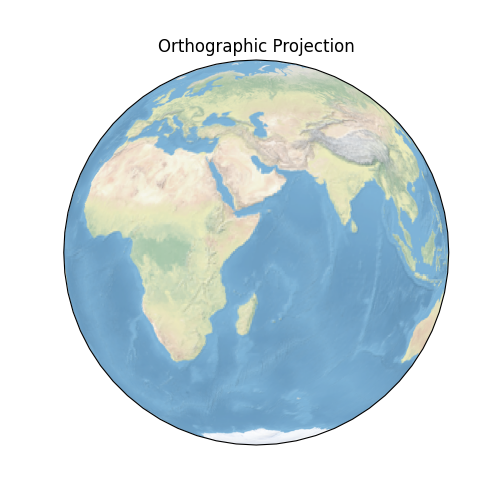
m.drawcoastlines(color='blue')

m.drawcountries(color="red")

ax. set\_title('Countries Projection')

plt.show()

**Outputs**

****