Program 1: Demonstrate various descriptive analytics on IRIS dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
edu = pd.read_csv('/content/drive/MyDrive/DataScienceResources/IRIS_DATASET.csv')
edu.head()
edu.describe()
edu['5.1']
#filtering of data
x = edu[edu['5.1']>6.5].tail()
xyz = pd.DataFrame(x, columns = ['5.1'])
print(xyz)
edu[edu['5.1']>4]
edu
#sorting 1.4 in descending order
edu.sort_values(by='1.4', ascending=False, inplace = True)
edu.head()
group = edu[['5.1', '1.4']].groupby('5.1').mean()
group.head()
#rearranging the data
filtered = edu[edu['1.4'] > 6.2]
pivedu = pd.pivot_table(filtered, index = ['3.5'], values = ['5.1'],
columns=['1.4'])
pivedu.head()
#ranking the data
rankedu = pivedu.rank(axis = 0)
rankedu.head()
#plotting total sum
pivedu.sum(axis = 1).sort_values(ascending = False).plot(kind = 'bar', alpha = 0.9,
title = 'Total Sum')
plt.show()
#slicing operation
edu[140:145]
#null values
x = edu[edu['5.1'].isnull()].head()
xyz = pd.DataFrame(x, columns = ['5.1'])
print(xyz)
#sqrt function
edu['5.1'].apply(np.sqrt).head()
#lambda function
edu['5.1'].apply(lambda x: x**2).head()
```

Program 2: Demonstrate Reading and writing of data from Text Files

```python
#program to show various ways to read and write data in a file
file1 = open("myfile.txt", 'w')
L = ["This is Delhi \n", "This is Paris \n", "This is London \n"]
file1.write("Hello \n")
file1.writelines(L)
file1.close()

file1 = open("myfile.txt", 'r+')
print("Output of Read function is ")
print(file1.read())
print()
file1.seek(0)

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

file1.seek(0)
print("Output of Read(9) function is ")
print(file1.read(9)) #First 9 characters of the file
print()
file1.seek(0)

print("Output of Readline(9) function is ")
print(file1.readline(9)) #First 9 characters of the first line of the file
print()
file1.seek(0)

print("Output of Readlines function is ")
print(file1.readlines())
file1.close()
#python program to illustrate append vs write mode
file1 = open("myfile.txt", 'w')
L = ["This is Delhi \n", "This is Paris \n", "This is London \n"]
file1.writelines(L)
file1.close()

file1 = open("myfile.txt", 'a')
file1.write("Today \n")
file1.close()

file1 = open("myfile.txt", 'r')
print("Output of Readlines after appending")
print(file1.readlines())
print()
file1.close()

file1 = open("myfile.txt", 'w')
file1.write("Tomorrow \n")
file1.close()

file1 = open("myfile.txt", 'r')
print("Output of Readlines after writing")
print(file1.readlines())
```

Program 3: Demonstrate reading data from the web (Web scraping)

```
#python program to read webfiles
import requests
from bs4 import BeautifulSoup

# URL of the webpage to read
url = "https://www.lipsum.com/"

# Send a GET request to the URL
response = requests.get(url)

# Check if the request was successful
if response.status_code == 200:
    # Parse the webpage content
    soup = BeautifulSoup(response.text, "html.parser")

    # Extract and print the textual content (strips out HTML tags)
    print(soup.get_text())
else:
    print(f"Failed to retrieve the page. Status code: {response.status_code}")
```

Program 6: Normal Probability Distribution on student scores.

```
#probability density
import numpy as np

def normal_dist(x, mean, sd):
    prob_density = (1 / (np.sqrt(2 * np.pi * sd**2))) * np.exp(-0.5 * ((x - mean) /
sd)**2)
    return prob_density
mean = 0
sd = 1
x = 1
result = normal_dist(x, mean, sd)
print(result)


#normal distribution
import numpy as np
import matplotlib.pyplot as plt

mean = 100
std_dev = 15
size = 100000
values = np.random.normal(loc=mean, scale=std_dev, size=size)
plt.hist(values, 100)
plt.axvline(values.mean(), color='red', linestyle='dashed', linewidth=2)
plt.show()


#nomral probability distribution
import scipy.stats as stats
import numpy as np
mean = 78
std_dev = 25

# a. Percentage of Students who got less than 60 marks
percentage_less_than_60 = stats.norm.cdf(60, loc=mean, scale=std_dev) * 100
print(f"Percentage of students who got less than 60 marks:
{percentage_less_than_60:.2f}%")
# b. Percentage of Students who have scored More than 70
percentage_more_than_70 = (1 - stats.norm.cdf(70, loc=mean, scale=std_dev)) * 100
print(f"Percentage of students who have scored More than 70:
{percentage_more_than_70:.2f}%")

# c. Percentage of Students who have scored More than 75 and less than 85
percentage_between_75_and_85 = (stats.norm.cdf(85, loc=mean, scale=std_dev) -
stats.norm.cdf(75, loc=mean, scale=std_dev)) * 100
print(f"Percentage of students who have scored More than 75 and less than 85:
{percentage_between_75_and_85:.2f}%")
```

Program 7: Linear Regression Line has equation Y = a + bX... Write a python program to describe Linear Regression.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.api as sm
#generate the data. Calculate Xmean, Ymean, Sxx, Sxy to find the value of slope and
intercept of regression line.
x = np.array([1,2,3,4,5])
y = np.array([7,14,15,18,19])
n = np.size(x)
xmean = np.mean(x)
ymean = np.mean(y)
print(f'xmean = {xmean}  ymean = {ymean}')
Sxy = np.sum(y*x) - n*xmean*ymean
Sxx = np.sum(x*x) - n*xmean*xmean
print(f'Sxy = {Sxy}  Sxx = {Sxx}')
b1 = Sxy/Sxx
b0 = ymean - b1*xmean
print(f'Slope b1 = {b1}  Intercept b0 = {b0}')
plt.scatter(x,y)
plt.xlabel('Independent Variable x')
plt.ylabel('Dependent Variable y')
#plot the given data points and fit the regression line
ypred = b1*x + b0
plt.scatter(x,y,color='blue')
plt.plot(x,ypred,color='red')
plt.show()
#analyze the performance of the model by calculating the mean squared error and R^2
error = y - ypred
se = np.sum(error**2)
print("Squared Error = ", se)
mse = se/n
print("Mean Squared Error = ", mse)
rmse = np.sqrt(mse)
print("Root Mean Squared Error = ", rmse)
SSt = np.sum((y - ymean)**2)
print("Total Sum of Squares = ", SSt)
r2 = 1 - (se/SSt)
print("R^2 = ", r2)
#use scikit library to confirm the above steps
x = x.reshape(-1,1)
regressionModel = LinearRegression()
#fit the data
regressionModel.fit(x,y)
print(f'Slope b1 = {regressionModel.coef_[0]}  Intercept b0 =
{regressionModel.intercept_}')
#predict
ypred = regressionModel.predict(x)
#model evaluation
mse = mean_squared_error(y,ypred)
print("Mean Squared Error = ", mse)
rmse = np.sqrt(mse)
print("Root Mean Squared Error = ", rmse)
r2 = r2_score(y,ypred)
print("R^2 = ", r2)
```

Program 4: Build a model that tells the prices of homes with 3300 sq. ft and 5000 sq ft using Linear Regression.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import linear_model
df = pd.read_csv('/content/drive/MyDrive/DataScienceResources/house_prices.csv')
plt.xlabel("Area(sq.ft.)")
plt.ylabel("Price($)")
plt.scatter(df.Area, df.Price)
reg = linear_model.LinearRegression()
reg.fit(df[['Area']], df.Price)
reg.predict([[3300]])
plt.xlabel("Area(sq.ft.)")
plt.ylabel("Price($)")
plt.scatter(df.Area, df.Price)
plt.plot(df.Area, reg.predict(df[['Area']]), color='blue')
```

Program 5: Plot Mean and Standard Deviation in Pandas

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Make the dataframe for evaluation on Errorbars
df = pd.DataFrame({
    'insert': [0.0, 0.1, 0.3, 0.5, 1.0],
    'mean': [0.009905, 0.45019, 0.376818, 0.801856, 0.643859],
    'quality': ['good', 'good', 'poor', 'good', 'poor'],
    'std': [0.003662, 0.281895, 0.306806, 0.243288, 0.322378]})

print(df)
fig, ax = plt.subplots()
for key, group in df.groupby('quality'):
    group.plot('insert', 'mean', yerr='std', label=key, ax=ax)
plt.show()
qual = df.groupby("quality").agg({'mean': 'mean', 'std': 'mean'})

# Plot using horizontal bar plot with error bars for std deviation
qual['mean'].plot(kind="barh", xerr=qual['std'], color='green', title="Mean and
Standard Deviation by Quality", legend=False)
plt.xlabel("Mean")
plt.ylabel("Quality")
plt.show()
```