PROJECT 1:

Basic Plotting in Python using matplot lib.pylot library

Problem Statement: Write a python program that will print out a two-dimensional plot of any two features for all three varieties of the iris flower in the plot area of Spyder using the features of matplotlib.pyplot.

Method of approach: With the given dataset, Irisdata.txt, I began by reading the file each by line and then storing it in a string called 's'. Once the user enters the file name, I've coded it to check for the file's existence, and if it does, it begins the processes described in the body; otherwise, it prompts the user to provide the file name again. After storing the read data in the variable ‘s’, the .split() function is used to break the line into distinct strings, which are then converted to int by mentioning it explicitly. By accessing the last element of the array, the next approach is to declare a temporary variable "temp" to exclude the class type of the iris flower. Once that is done, the data is saved in the previously stated empty matrix, and when it reaches the class type I have created it to categorize into numbers (i.e., if it is setosa, the value is set to 0, if it is versicolor, it is set to 1, and for virginica, it is set to 2). After that, to avoid code recursion, I wrote a function called "graph_plot." This function's aim is to obtain the required inputs from the user before proceeding with the execution. It requests the user's file name as well as the coordinates for plotting the graph for the iris flower. Then, within the for loop, I established an iteration variable ‘i’ to verify the value of the class type that I previously converted to int (i.e., 0, 1, and 2), and then used the plt.scatter() method to plot the coordinates in the graph with the marker type and color specified in the problem statement. Then I used an if..elif..else statement to classify the x and y input values from the user and display them as the axis's names in the graph. Next, using the instructions at https://matplotlib.org/stable/api/_as_gen/matplotlib.lines.Line2D.html, I created a variable "legend props" that specifies the legend's properties, then plotted it using plt.legend(handles = legend props) as per the syntax referred online. Finally, I created a variable called "option" and set its default value to "y." According to the problem statement, when the graph has been plotted, the user should be asked "Would you like to do another plot? (y/n)" and the program stops or the defined function runs again based on the input. To do so, I used an if..else statement within

the while loop, which causes the loop to execute the body according on the condition.

In my opinion, the feature combination of Sepal length vs Sepal width is the best of all the combinations because it provides precise numbers due to the small size of the measurements.

Output and Screenshots:

Fig.1. Screenshot of the spyder console:

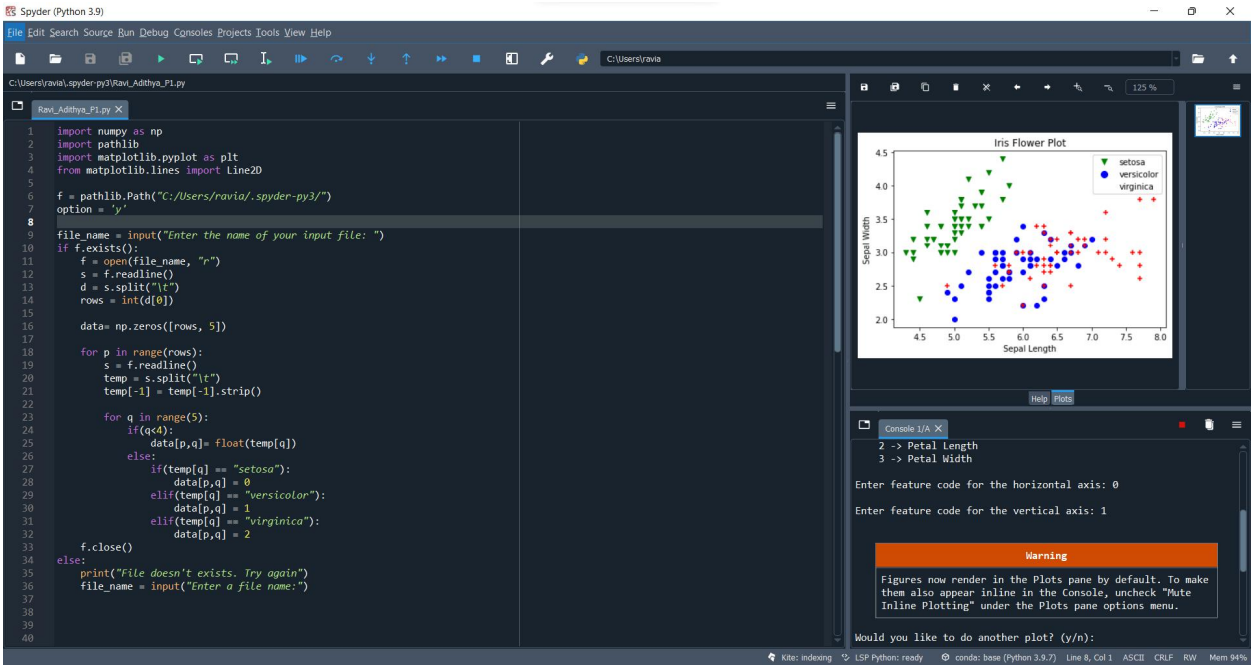
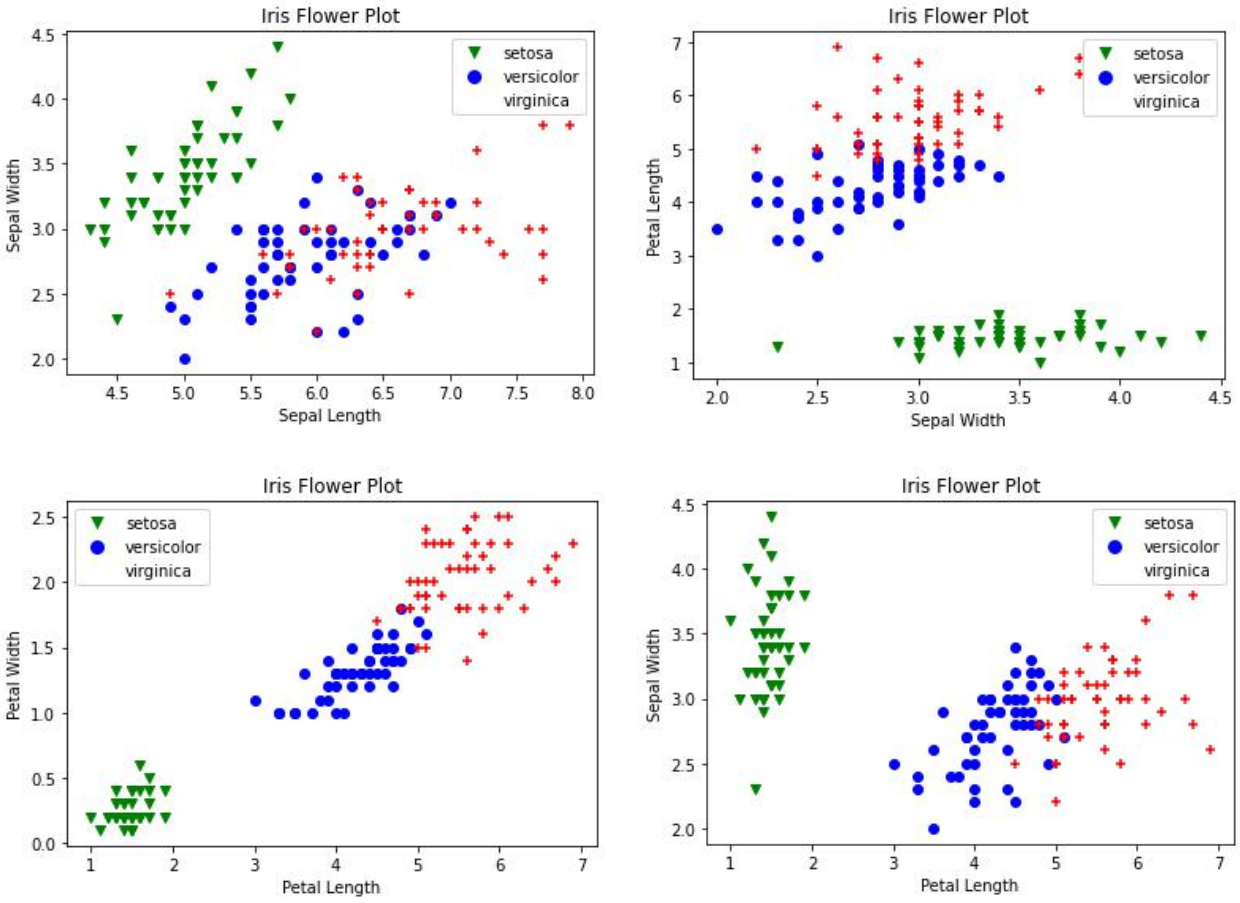


Fig.2. Screenshots of Different plots:



Source Code:

```
import numpy as np
import pathlib
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
```

```
f = pathlib.Path("C:/Users/ravia/.spyder-py3/")
option = 'y'

file_name = input("Enter the name of your input file: ")
if f.exists():
    f = open(file_name, "r")
    s = f.readline()
    d = s.split("\t")
    rows = int(d[0])

    data= np.zeros([rows, 5])

    for p in range(rows):
        s = f.readline()
        temp = s.split("\t")
        temp[-1] = temp[-1].strip()

        for q in range(5):
            if(q<4):
                data[p,q]= float(temp[q])
            else:
                if(temp[q] == "setosa"):
                    data[p,q] = 0
                elif(temp[q] == "versicolor"):
                    data[p,q] = 1
                elif(temp[q] == "virginica"):
                    data[p,q] = 2
```

```
    f.close()
else:
    print("File doesn't exists. Try again")
    file_name = input("Enter a file name:")




def graph_plot():
    print("You can do a plot of any two features of the Iris Data set\nThe feature codes are:\n\t0 -> Sepal Length\n\t1 -> Sepal Width\n\t2 -> Petal Length\n\t3 -> Petal Width")


    x = int(input("Enter feature code for the horizontal axis: "))
    y = int(input("Enter feature code for the vertical axis: "))
    for i in range(rows):


        if data[i,4] == 0:
            plt.scatter(data[i,x], data[i,y], c = "green", marker = "v", label ="setosa")
        elif data[i,4] == 1:
            plt.scatter(data[i,x], data[i,y], c = "blue", marker = "o", label ="versicolor")
        else:
            plt.scatter(data[i,x], data[i,y], color = "red", marker = "+", label ="virginica")


    if(x == 0):
        x = "Sepal Length"
    elif(x == 1):
```

```
    x = "Sepal Width"
elif(x == 2):
    x = "Petal Length"
else:
    x = "Petal Width"
if(y == 0):
    y = "Sepal Length"
elif(y == 1):
    y = "Sepal Width"
elif(y == 2):
    y = "Petal Length"
else:
    y = "Petal Width"


plt.title("Iris Flower Plot")
plt.xlabel(x)
plt.ylabel(y)


legend_props = [Line2D([0], [0], marker='v', color='white',
markerfacecolor='green', label='setosa', markersize=10),
        Line2D([0], [0], marker='o', color='white', markerfacecolor='blue',
label='versicolor', markersize=10),
        Line2D([0], [0], marker="+", color ='white', markerfacecolor='red',
label='virginica', markersize=10)]
plt.legend(handles = legend_props)
plt.show()
```

```
graph_plot()
while option == 'y':
    option = input("Would you like to do another plot? (y/n): ")
    if option == 'y':
        graph_plot()
    elif option == 'n':
        break;
```