University of Canberra
Faculty of Science and Technology

Programming for Data Science G (11521)

Assignment 3 or Take-Home Exam

Available date: **17:00 Friday 03/11/2023 (Week 14)**
Submission date: **16:59 Saturday 04/11/2023 (Week 14)**
Type: Individual assignment
Total mark: 40
Proportion of unit assessment: 40%
Submission:

- Python (.py) or Jupyter Notebook (.ipynb) files **create one file for each Question**
- Compress all files into a single file (.zip) and name as 'your_name_IDno.zip'
- Submit this compressed (.zip) file.

Extension: No extension is possible for this assessment unfortunately, given its nature and the marking constraints. Please apply for Deferred Assignment (only for medical emergencies) before Wednesday 17:00 01/11/2023. The deferred assignment will be available at 17:00 Friday 19/01/2024 and due by 16:59 Saturday 20/01/2024.

Read Carefully: Please note that all files required for this assessment are available from the 'Download files for Assignment 3' link under Assignment 3 in the Modules page. Alternatively, they could be retrieved from your Tutorial folders.

Question 1 [4 marks] (Lectures): Apply **Index and Slice Operation** to list x below to have lists y, z, t and u. Add your code to y, z, t and u (`# missing code`) to implement this task in the program below.

```
x = [[a*b for a in range(2,6)] for b in range(3,9)]
y = # missing code
z = # missing code
t = # missing code
u = # missing code
print('x =', x)
print('y =', y)
print('z =', z)
print('t =', t)
print('u =', u)
```

The complete program without missing code will output the following:

```
x = [[6, 9, 12, 15], [8, 12, 16, 20], [10, 15, 20, 25], [12, 18, 24, 30], [14, 21,
28, 35], [16, 24, 32, 40]]
y = [[8, 12, 16, 20], [10, 15, 20, 25]]
z = [[12, 18, 24, 30]]
t = [[14, 21, 28, 35], [6, 9, 12, 15]]
u = [12, 18]
```

Question 2 [4 marks] (Tutorials): Implement function **load_iris_dataset** below that inputs **iris.data** file and returns **data**, **target** and **target_names** as seen in Week 11 Tutorial. Use the program below to test your function implementation (please do not change the program) and the **iris.data** file in **Week 5 Tutorial**. You may optionally use the *sklearn* package to load the iris dataset within the function.

```
import numpy as np

def load_iris_dataset(filename):

       # implement this function here

# end of function

X, y, class_labels = load_iris_dataset('iris.data')
print('X =', X)
print('y =', y)
print('class labels =', class_labels)
```

The program with correct function implementation will output the following:

```
X = [[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
 [4.8 3.  1.4 0.1]
 [4.3 3.  1.1 0.1]
 [5.8 4.  1.2 0.2]
 [5.7 4.4 1.5 0.4]
 [5.4 3.9 1.3 0.4]
 [5.1 3.5 1.4 0.3]
 [5.7 3.8 1.7 0.3]
 [5.1 3.8 1.5 0.3]
 [5.4 3.4 1.7 0.2]
 [5.1 3.7 1.5 0.4]
 [4.6 3.6 1.  0.2]
 [5.1 3.3 1.7 0.5]

 . . .      . . .          # some lines deleted to reduce length

 [6.3 3.4 5.6 2.4]
 [6.4 3.1 5.5 1.8]
 [6.  3.  4.8 1.8]
 [6.9 3.1 5.4 2.1]
 [6.7 3.1 5.6 2.4]
 [6.9 3.1 5.1 2.3]
 [5.8 2.7 5.1 1.9]
 [6.8 3.2 5.9 2.3]
 [6.7 3.3 5.7 2.5]
y = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

```
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
class labels = ['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

Question 3 [4 marks] (Error): There is a logical error in the function implementation below and the program only outputs the following `[(-0.379504, -0.725536)]` instead of all data in the `ellipse1.txt` file. Use this **`ellipse2a.txt`** file in **Week 4 Tutorial**.

```
def read_ellipse_file(filename):
    dataset = []
    f = None
    try:
        f = open(filename, 'r')
        while True:
            line = f.readline()
            if len(line) == 0: #end of file
                break
            line = line.replace('\n', '')
            xystring = line.split(' ')
        dataset.append((float(xystring[0]), float(xystring[1])))
    except Exception as ex:
        print(ex.args)
    finally:
        if f:
            f.close()
    return dataset
#end of function

data = read_ellipse_file('ellipse1.txt')
print(data)
```

Correct the function implementation and run the program to output all data from the file as follows

```
[(0.810831, -0.552096), (1.357364, 1.165427), (0.464588, 0.34218), (-0.643902, 0.164353),
(0.836779, 0.896883), (-1.486807, -0.659875), (-1.08032, -0.691931), (-0.454588, -1.313122),
(0.015564, 0.301778), (0.505334, -0.808687), (-0.04276, 1.120155), (-1.208152, 0.058249),
(0.703988, -0.228097), (-1.15505, -0.714617), (-0.534014, 0.581944), (-0.030791, -0.977369),
(0.582063, -0.364418), (0.150766, -0.747611), (0.026504, -0.455569), (-0.987247, -0.706803),
(1.179535, 0.456121), (0.319984, 0.95496), (0.997544, -0.316347), (-0.986591, -0.959999),
(0.235613, 0.309634), (-0.127184, 0.493345), (-0.299913, 0.514379), (1.410037, 0.234304),
(-1.229346, -0.264383), (0.444969, -0.83582), (0.367703, -0.115663), (-1.167328, -1.253192),
(-0.355128, -0.492419), (-1.077822, -0.624575), (1.482912, 1.224543), (-0.061826, 0.854676),
(0.448663, 1.119711), (0.131415, -0.391768), (0.86191, 1.28364), (1.610711, 0.863136), (-
1.219452, -1.230034), (0.950742, -0.447317), (-0.136005, -1.021787), (0.332291, -0.114963),
(-0.914996, -0.583172), (0.559103, 0.845035), (-0.334597, 0.788516), (-0.26004, 0.451973),
(-1.560841, -0.849909), (0.240598, -0.019207), (0.485926, -0.435991), (0.162651, 0.065829),
(0.908328, -0.259693), (0.847576, -0.534039),
. . .       . . .           # some lines deleted to reduce length
(4.581788, 4.297241), (3.138971, 1.630446), (3.232468, 2.695755), (3.905107, -0.094816),
(4.547954, 0.319106), (3.647935, 2.562436), (4.277992, 1.135122), (3.842152, 0.784564),
(4.595315, 4.068041), (4.272179, 4.184751), (4.243864, 1.589254), (3.541893, 2.264996),
(4.058019, 2.347678), (4.319215, -0.246127), (3.786731, 2.405603), (3.765264, 1.354399),
(4.227114, 3.969368), (3.856169, 4.865862), (3.631329, 1.835823), (4.200977, 3.235876),
(3.676197, 3.970413), (3.612092, 0.48177), (3.47764, 0.167865), (3.352367, 2.238113),
(4.148772, 2.156695), (4.02844, 3.749225), (4.511957, 2.506159), (4.437639, 3.251211),
(3.462291, 3.258124), (4.670854, 4.064812), (4.227865, 2.782661), (4.314377, 2.156743),
(4.242528, 0.039906), (4.371407, 2.713846), (4.86595, 0.722895), (4.2555, 0.686982),
(3.555857, -0.046087), (3.976513, 3.751556), (3.744741, 0.380391), (4.717089, 2.967637),
(3.381081, 3.375803), (3.964701, 2.051801), (4.275585, 2.496787), (4.086547, 2.350102),
(4.195333, 3.627176), (3.972308, 1.631397), (3.446353, 1.739243), (3.507772, -0.259507),
```

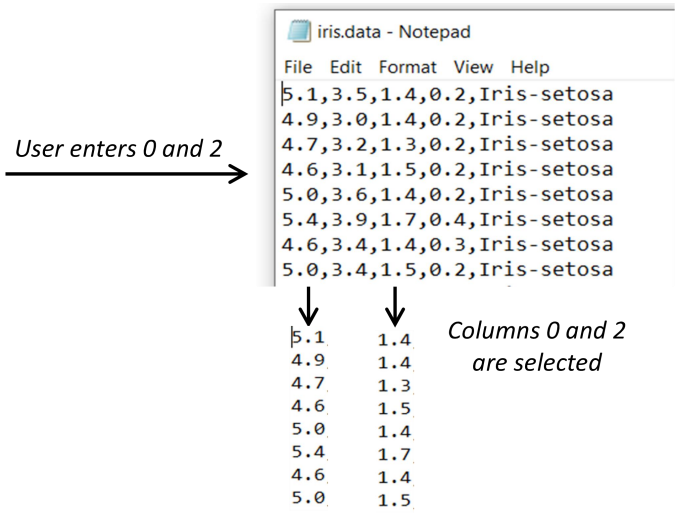
```
(3.752135, 1.186792), (4.342305, 4.512587), (3.904573, 3.183321), (3.474702, 2.146167),
(4.807326, 0.550306), (4.697809, 3.732143), (4.764769, 3.746963), (3.837609, 1.819013),
(3.327122, 0.877491), (4.179445, 0.509374), (3.758163, 4.535824), (3.420195, 1.667822),
(3.358179, 1.207931), (4.333138, 1.123023), (4.500865, 2.093882), (4.174701, 4.928042),
(3.140143, 2.522568), (4.281658, 3.494965), (3.551645, 1.458362), (4.178576, 4.075363),
(4.237745, 3.754904)]
```

Question 4 [8 marks] (NumPy): Below is the program you used in **Example 14** in **Week 10 Tutorial**. In that example, the first two columns of data in Iris dataset were selected for plotting:

```
X = iris.data[:, :2]
```

Your tasks for this question are as follows.

- Remove `X = iris.data[:, :2], plt.xlabel('Sepal length')` and `plt.ylabel('Sepal width')` from the program.
- Add `import numpy as np`
- Add your code to ask the user to input 2 numbers between 0 and 3 inclusive.
- ***Use the two input numbers as column indices*** and built-in functions in **NumPy** package to implement the program ***to select 2 columns of data*** from the four columns in the **Iris** dataset and ***update `xlabel` and `ylabel` accordingly***.
- Please **do not** use **Pandas** or other packages. Only **Numpy** package is imported.



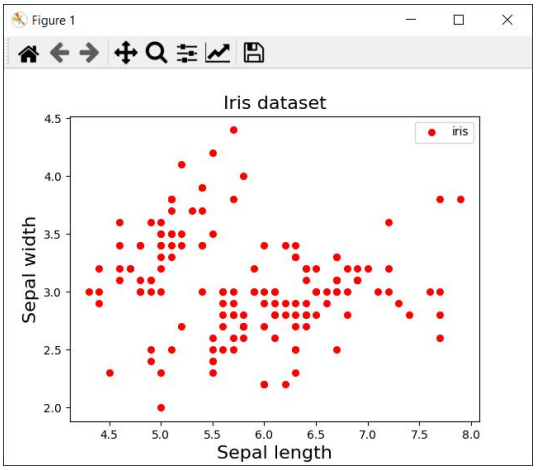
All data from the two selected columns are added to X

```
import matplotlib.pyplot as plt
from sklearn import datasets

iris = datasets.load_iris() # returns np-array (150 rows and 4 columns)

# Each row is a 4D sample
# Column: Sepal Length, Sepal Width, Petal Length and Petal Width.

X = iris.data[:, :2]  #take the first two columns.
y = iris.target

# Target names: Setosa, Versicolour, and Virginica
labels = iris.target_names

# Plot the data
plt.scatter(X[:, 0], X[:, 1], label='iris', c='r', marker='o', s=30)
```

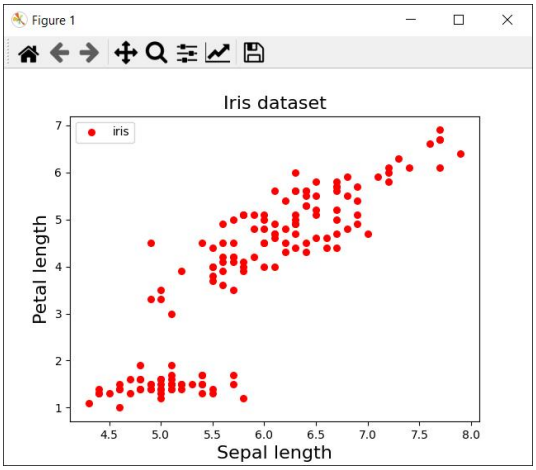
```
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')

plt.title('Iris dataset')
plt.legend()
plt.show()
```

Expected Output: Let **a** and **b** be two input numbers used as indices of two columns selected from the Iris dataset. Below are plots with different input values of **a** and **b**: (x label and y label are changed accordingly)

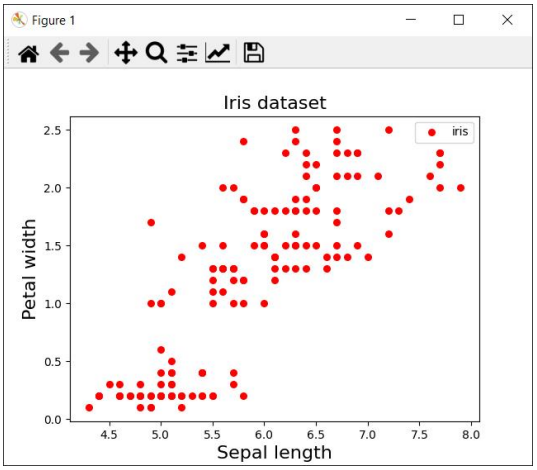
a = 0, b = 1



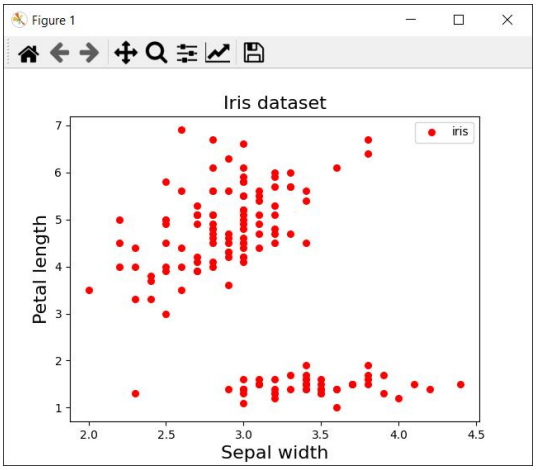
a = 0, b = 2



a = 0, b = 3

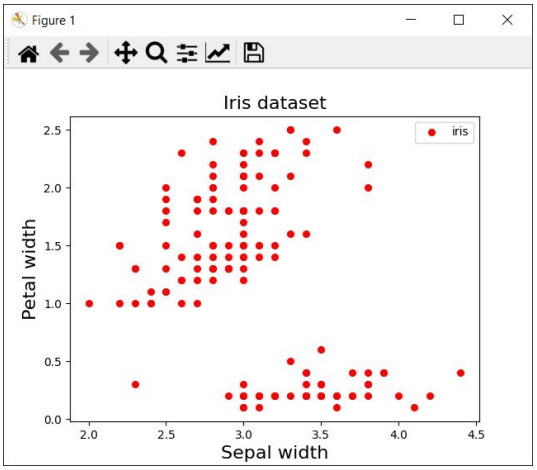


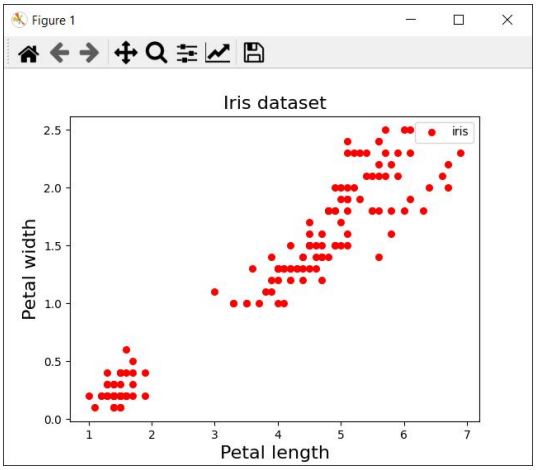
a = 1, b = 2



a = 1, b = 3

a = 2, b = 3





Question 5 [8 marks] (Pandas): Implement a program that uses **Pandas** package to input data from **mycsv.csv** and output data to **myexcel.xls** file (2 sheets). Download these files from Assignment 3 module from the 'Modules' page in Canvas site.

The data to be input from **mycsv.csv** are from the following selected columns and rows:

- **Columns:** only 4 columns with the following headers **`SIS User ID`, `Submit Assignment 1 (57584)`, `Submit Assignment 2 (57585)`,** and **`Submit Assignment 3 (57473)`**
- **Rows**: only rows that have ID value in column **`SIS User ID`**. Those ID values start with either letter **u** (e.g., u123456) or letter **s** (e.g., s123484). Other ID values are ignored.

You will write code to change column headers as follows:

- Change **SIS User ID** to **StudentID**
- Change **Submit Assignment 1 (57584)** to **Ass1**
- Change **Submit Assignment 2 (57585)** to **Ass2**, and
- Change **Submit Assignment 3 (57473)** to **FE**.

You will also write code to separate data in to 2 groups (group 1: ID starting with u and group 2: ID starting with s). **Hint:** Remove rows that do not contain id's starting with 's' or 'u' and perform row selection via string match.

Arrange the **'s' group data** in the **descending order of A2**, and 'u' group data in the **ascending order of FE.**

The data to be output to sheets named `UID' and `SID' respectively in **myexcel.xls** are the sorted 's' and 'u' group data

- **Sheet UID**: columns: **StudentID**, **Ass1**, **Ass2**, and **FE**, and rows: only rows that have ID value starting with letter **u**.
- **Sheet SID**: columns: **StudentID**, **Ass1**, **Ass2**, and **FE**, and rows: only rows that have ID value starting with letter **s**.

Note: If you get an error saying 'xlwt module not found', you will need to run 'pip install xlwt' in your Conda virtual environment and add 'import xlwt' at the beginning of your python code.

Expected Output in File myexcel.xls:

Sheet SID contents:

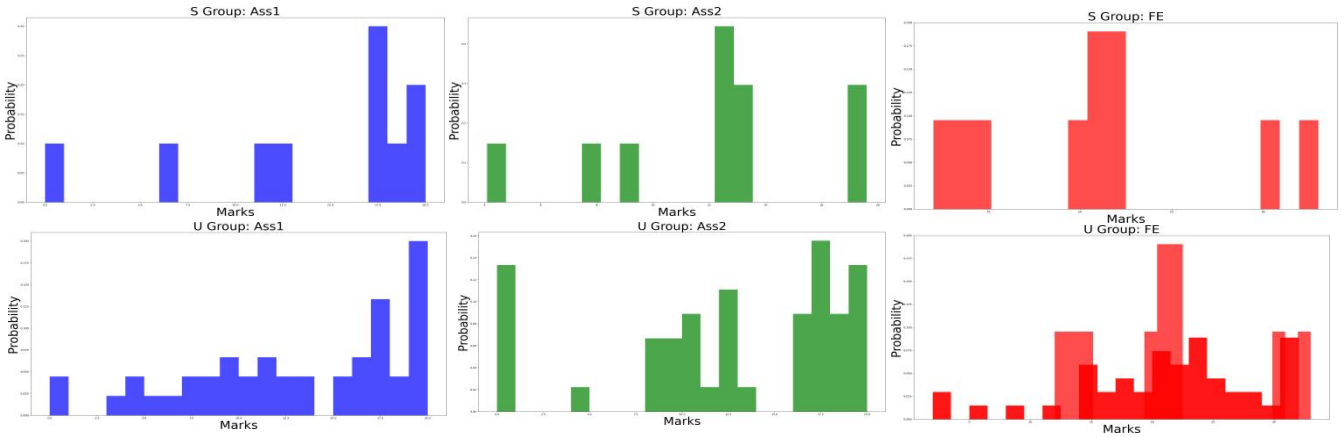
StudentID	Ass1	Ass2	FE
s123492	17.5	17.6	14
s123463	20	17.5	22
s123473	17.5	13.3	22
s123484	18.8	13.3	12
s123494	17.75	12.75	33
s123465	0	12.65	15
s123497	11	12.65	21
s123457	6.8	9.4	30
s123472	12.75	8.09	20
s123509	19.4	4.1	21

Sheet UID contents:

StudentID	Ass1	Ass2	FE
u123482	9.7	10.15	2
u123483	15.15	8.09	3
u123480	16.5	9.4	6
u123481	19.35	20	8
u123479	20	0	11
u123495	17.5	17.5	14
u123461	5.5	13.3	14
u123499	19.05	4.1	15
u123503	19.75	19.35	15
u123462	20	12.75	16
u123468	20	0	16
u123486	16.9	17.5	17
u123469	10.15	9.4	18
u123507	19.75	12.65	18
u123508	15.45	18.1	19
u123489	18.75	18.8	19
u123485	11.5	12.75	21
u123505	17	17.5	21
u123456	10	0	21
u123460	4.5	8.09	21
u123467	9.4	0	21
u123470	8.09	20	22
u123490	7.95	11.5	22
u123500	6.25	16.4	22
u123496	9.1	17.3	22
u123466	0	18.1	23
u123491	3	16.9	23
u123498	11.15	18.1	23
u123474	17.3	0	24
u123506	19.3	17.3	24
u123459	0	10.15	24
u123458	13.35	20	25

u123475	12.65	0	25
u123501	17.65	20	25
u123510	12.9	16.4	26
u123488	8.3	12.65	26
u123504	17.4	12.75	28
u123487	17.6	17.3	28
u123493	7.3	8.3	30
u123471	13.3	10.15	31
u123464	11.75	17.3	31
u123477	4.1	20	31
u123478	16.4	10.15	32
u123502	20	16.5	32
u123476	18.1	9.4	32

Question 6 [4 marks] (Matplotlib): Implement a program that uses **Matplotlib** and **Pandas** packages to read data from 2 sheets in **myexcel.xls** and plots **histograms** as per the outputs below for the distributions of **Ass1**, **Ass2** and **FE**, in sheets **UID** and **SID**. Expected outputs with number of bins = 20 and face colours of blue, green, and red respectively for Ass1, Ass2 and FE.



Question 7 [4 marks] (Scikit-learn): Use the program in **Week 11 Tutorial** for Scikit-learn. Implement function **`get_confusion_matrix`** below that inputs **`y_pred`**, **`y_test`** and **class labels**, and outputs the confusion matrix. The function should work with any number of class labels. Note the numbers in **`y_pred`** and **`y_test`** are class label indices.

Please use **Pandas** package to implement the function and **do not** use other packages.

```
import pandas as pd
##############################################
def get_confusion_matrix(true_list, predicted_list, class_labels):

    # Add your code here. Only Pandas package is imported

    return #confusion matrix
##############################################
```

Use the program in Week 11 Tutorial that uses Scikit-learn (Page 6) with to test this function. Remove the following code from that program

```
#Plot confusion matrix
```

```
metrics.plot_confusion_matrix(classifier, X_test, y_test,
display_labels=class_names)
plt.show()
```

and add the following code

```
labels = ['Setosa', 'Versicolour', 'Virginica']
confusion_matrix = get_confusion_matrix(y_pred, y_test, labels)
print(confusion_matrix)
```

then add the `get_confusion_matrix` function you implemented. The program with correct function implementation will output the following:

```
                Setosa  Versicolour  Virginica
Setosa              13            0          0
Versicolour          0           15          0
Virginica            0            1          9
```

Hint: Pandas functions exist that will allow you to compute the confusion matrix

Question 8 [4 marks] (Tkinter): Implement a graphical user interface (GUI) program using **Tkinter** with the following tasks:

- The GUI window only has a frame (left) and a canvas (right).
- The frame contains a dropdown list or a group of 3 radio buttons.
- User can select a data file from a dropdown list or from a group of 3 radio buttons displayed on the frame.
- After a data file is selected, the program will immediately display all data samples in that data file on the canvas. No button is required.
- The data files are **blue_2d.txt**, **red_2d.txt** and **unknown_2d.txt** used in **Assignment 1**. Any colour and shape can be used to display data samples. It is acceptable if your program can work with 2 dimensional data only.

--- END ---