University of Canberra
Faculty of Science and Technology

Programming for Data Science G (11521)
Week 4 Tutorial
Modules – Input/Output – Exception Handling – Assignment 1

Objectives

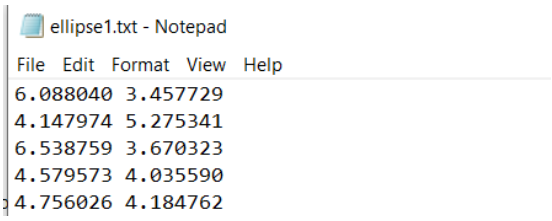
- To create a module and add functions for input/output to this module
- To implement functions for reading data and calculating distances

Use Visual Studio or PyCharm or Spyder (in Anaconda) to Create a New Python Project

- Use your own computer or login to UC Student Virtual Desktop https://frame.nutanix.com/university-of-canberra/ditm/uc-remote-access-student/launchpad/uc-virtual-desktop-student
- Open Visual Studio or PyCharm or Spyder (in Anaconda) to create a new project and name it **Week4Tutorial**.
- Add a new Python file to this project and name it **io_data_module.py**.
- Your project now has 2 Python files: **Week4Tutorial.py** file (for the main program) and **io_data_module.py** (the module file for input/output methods)

Add function to read data from file to **io_data_module.py**

- **Function to read 2-dimensional data from file and save data to tuples**. The dataset is in a text file below. Each line has 2 values which are x & y coordinates of a data sample.



```
6.088040 3.457729
4.147974 5.275341
6.538759 3.670323
4.579573 4.035590
4.756026 4.184762
```

Copy the following function to **io_data_module.py**

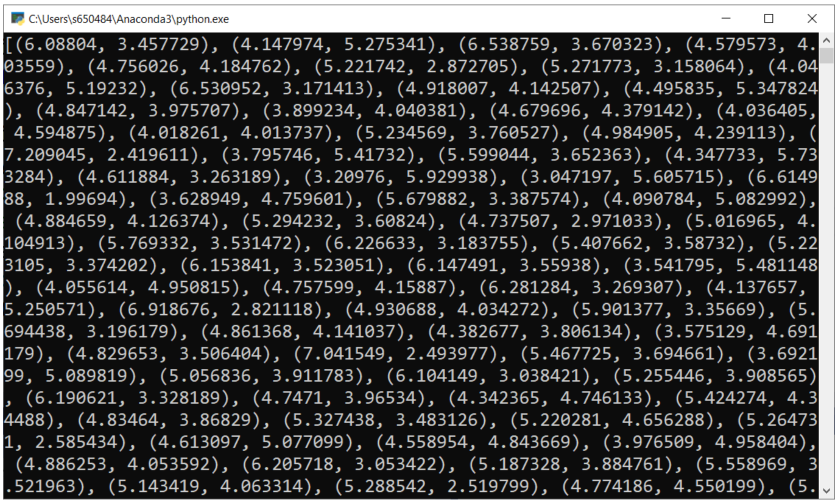
```
#Function to read 2D data from file and save data to a list of tuples
def read_data_file(filename):
    dataset = [] #dataset is a python list
    f = None
    try:
        f = open(filename, 'r')
        while True:
            line = f.readline()
            if len(line) == 0: #end of file
                break
            line = line.replace('\n', '') #remove end of line \n character
            xystring = line.split(' ') #x y coordinates in string format
            #use split function to separate x & y strings then
            #use float function to convert x & y strings to x & y numbers and
            #add them as a tuple (x, y) to dataset that is a list
            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
```

Call the function in Week4Tutorial.py

- Call the function in the program (**Week4Tutorial.py**) as follows

```
import io_data_module as iodata

#Open file and read data
data_list = iodata.read_data_file('ellipse1.txt')
print(data_list)
```

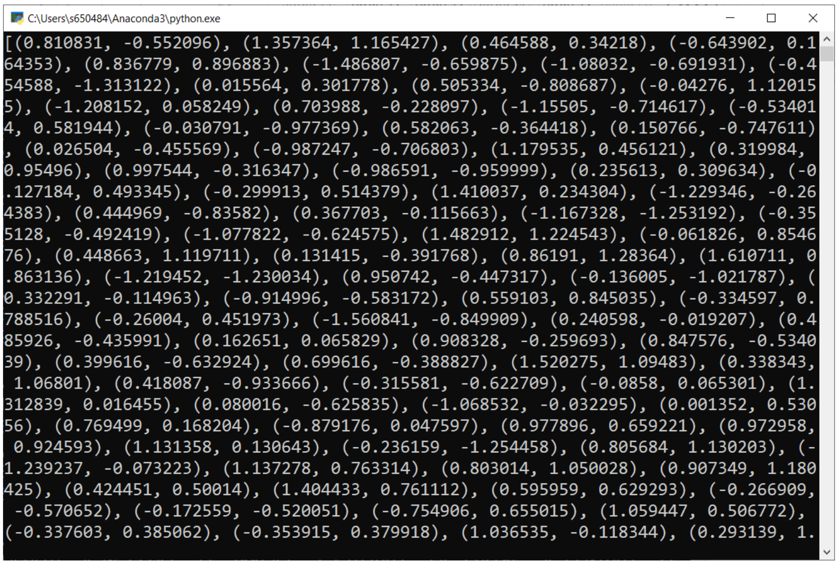
- Download **ellipse1.txt** and **ellipse2.txt** files from Canvas and place them in the same folder with .py files. Run the program. The data for **ellipse1.txt** is of the following form `[(x1, y1), (x2, y2),. . , (xn, yn)]`



C:\Users\s650484\Anaconda3\python.exe

```
[(6.08804, 3.457729), (4.147974, 5.275341), (6.538759, 3.670323), (4.579573, 4.
03559), (4.756026, 4.184762), (5.221742, 2.872705), (5.271773, 3.158064), (4.04
6376, 5.19232), (6.530952, 3.171413), (4.918007, 4.142507), (4.495835, 5.347824
), (4.847142, 3.975707), (3.899234, 4.040381), (4.679696, 4.379142), (4.036405,
 4.594875), (4.018261, 4.013737), (5.234569, 3.760527), (4.984905, 4.239113), (
7.209045, 2.419611), (3.795746, 5.41732), (5.599044, 3.652363), (4.347733, 5.73
3284), (4.611884, 3.263189), (3.20976, 5.929938), (3.047197, 5.605715), (6.6149
88, 1.99694), (3.628949, 4.759601), (5.679882, 3.387574), (4.090784, 5.082992),
 (4.884659, 4.126374), (5.294232, 3.60824), (4.737507, 2.971033), (5.016965, 4.
104913), (5.769332, 3.531472), (6.226633, 3.183755), (5.407662, 3.58732), (5.22
3105, 3.374202), (6.153841, 3.523051), (6.147491, 3.55938), (3.541795, 5.481148
), (4.055614, 4.950815), (4.757599, 4.15887), (6.281284, 3.269307), (4.137657,
5.250571), (6.918676, 2.821118), (4.930688, 4.034272), (5.901377, 3.35669), (5.
694438, 3.196179), (4.861368, 4.141037), (4.382677, 3.806134), (3.575129, 4.691
179), (4.829653, 3.506404), (7.041549, 2.493977), (5.467725, 3.694661), (3.6921
99, 5.089819), (5.056836, 3.911783), (6.104149, 3.038421), (5.255446, 3.908565)
, (6.190621, 3.328189), (4.7471, 3.96534), (4.342365, 4.746133), (5.424274, 4.3
4488), (4.83464, 3.86829), (5.327438, 3.483126), (5.220281, 4.656288), (5.26473
1, 2.585434), (4.613097, 5.077099), (4.558954, 4.843669), (3.976509, 4.958404),
 (4.886253, 4.053592), (6.205718, 3.053422), (5.187328, 3.884761), (5.558969, 3
.521963), (5.143419, 4.063314), (5.288542, 2.519799), (4.774186, 4.550199), (5.
```

- Change **ellipse1.txt** to **ellipse2.txt** and run your project again. The data has the same format



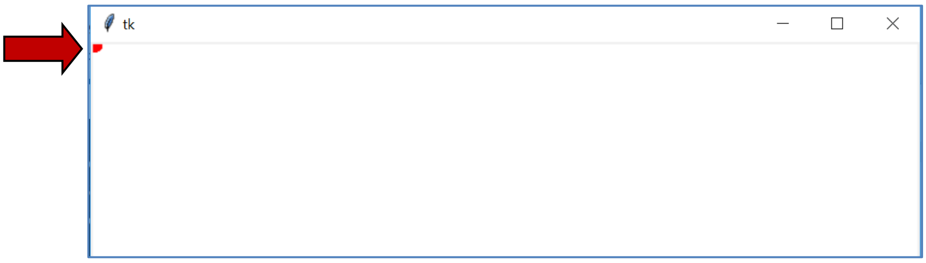
C:\Users\s650484\Anaconda3\python.exe

```
[(0.810831, -0.552096), (1.357364, 1.165427), (0.464588, 0.34218), (-0.643902, 0.1
64353), (0.836779, 0.896883), (-1.486807, -0.659875), (-1.08032, -0.691931), (-0.4
54588, -1.313122), (0.015564, 0.301778), (0.505334, -0.808687), (-0.04276, 1.12015
5), (-1.208152, 0.058249), (0.703988, -0.228097), (-1.15505, -0.714617), (-0.53401
4, 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, -0.26
4383), (0.444969, -0.83582), (0.367703, -0.115663), (-1.167328, -1.253192), (-0.35
5128, -0.492419), (-1.077822, -0.624575), (1.482912, 1.224543), (-0.061826, 0.8546
76), (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), (0.4
85926, -0.435991), (0.162651, 0.065829), (0.908328, -0.259693), (0.847576, -0.5340
39), (0.399616, -0.632924), (0.699616, -0.388827), (1.520275, 1.09483), (0.338343,
 1.06801), (0.418087, -0.933666), (-0.315581, -0.622709), (-0.0858, 0.065301), (1.
312839, 0.016455), (0.080016, -0.625835), (-1.068532, -0.032295), (0.001352, 0.530
56), (0.769499, 0.168204), (-0.879176, 0.047597), (0.977896, 0.659221), (0.972958,
 0.924593), (1.131358, 0.130643), (-0.236159, -1.254458), (0.805684, 1.130203), (-
1.239237, -0.073223), (1.137278, 0.763314), (0.803014, 1.050028), (0.907349, 1.180
425), (0.424451, 0.50014), (1.404433, 0.761112), (0.595959, 0.629293), (-0.266909,
 -0.570652), (-0.172559, -0.520051), (-0.754906, 0.655015), (1.059447, 0.506772),
(-0.337603, 0.385062), (-0.353915, 0.379918), (1.036535, -0.118344), (0.293139, 1.
```

- The following code is to display the data on Canvas as a list of tuples (data samples) `[(x1, y1), (x2, y2),. . , (xn, yn)]`. Add the green highlighted code to the program in **Week4Tutorial.py** as seen below

```
import io_data_module as iodata
import tkinter

#Open file and read data
data_list = iodata.read_data_file('ellipse1.txt')
print(data_list)

#Create canvas
top = tkinter.Tk()
C = tkinter.Canvas(top, bg="white", height=700, width=700)

#Display data
for x, y in data_list:
    C.create_oval(x-2, y-2, x+2, y+2, outline = "red", fill="red")

C.pack()
top.mainloop()
```

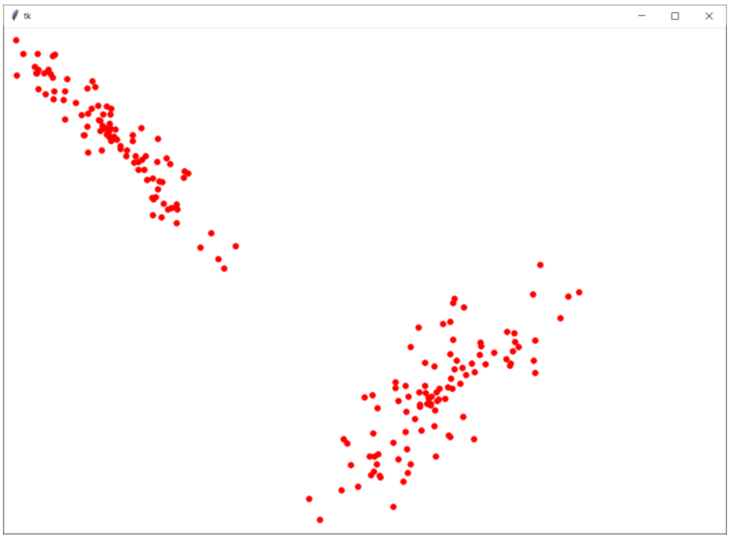
- Run your project and you will see all the data samples displayed in the top left corner on Canvas as below. The problem is the values of x & y in the data list are very small.



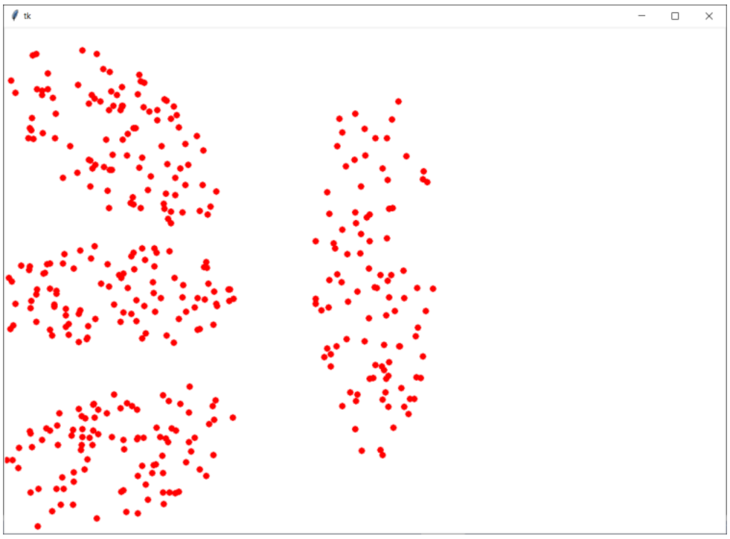
- To see the data samples clearly, we need to scale these x & y values. Add the yellow highlighted code to the program as follows

```
import io_data_module as iodata
import tkinter as tk

#Open file and read data
data_list = iodata.read_data_file('ellipse1.txt')
#print(data_list)

#Create canvas
top = tk.Tk()
C = tk.Canvas(top, bg="white", height=700, width=1000)

#Display data
s = 90 #scale factor
r = 4 #radius
for x, y in data_list:
    x = x*s + 150 #some values are negative so +150 is to make them positive
    y = y*s + 150
    C.create_oval(x-r, y-r, x+r, y+r, outline = "red", fill="red")

C.pack()
top.mainloop()
```

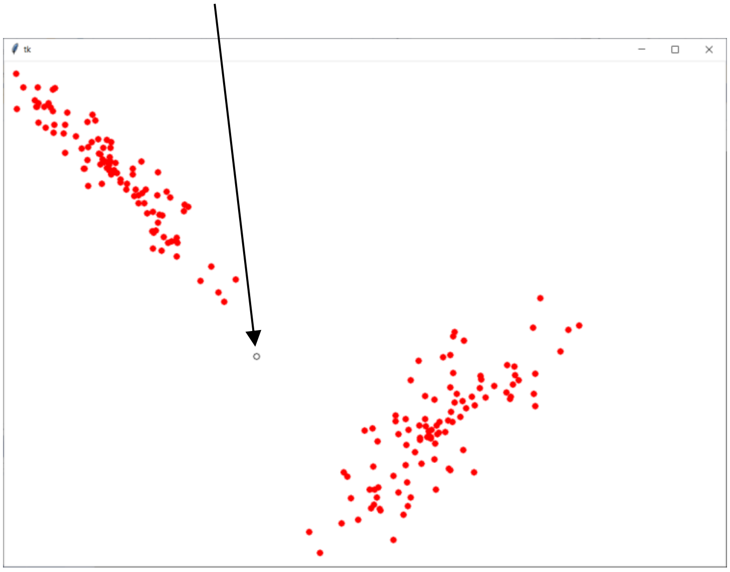
- Run the program again and we can see data samples as follows.



- Change the filename to ellipse2.txt and run the program.



Question 1: Modify your program (**Week4Tutorial.py**) to display the following data sample `unknown_sample = (2.236779, 2.896883)` with the **ellipse1** dataset as follows



Question 2: Write a function named **find_nearest_neighbour** that takes **unknown_sample** and **data_list** as its input parameters and returns the *nearest* data sample of **unknown_sample.** Place this function in **io_data_module.**

```
#define function
def find_nearest_neighbour(unknown_sample, data_list):

    #write your code here

    return nearest_sample
#end function
```

Question 3: Modify your program (**Week4Tutorial.py**) to call the **find_nearest_neighbour** function to get the nearest sample of the **unknown_sample**. Change colour of this nearest sample to black and draw a black line between this nearest sample and the unknown sample as follows