University of Canberra

Faculty of Science and Technology

**Programming for Data Science G (11521)**

**Week 6 Tutorial**

**Dictionary and Assignment 1**

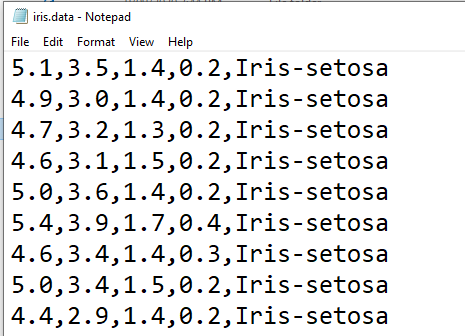
**Tasks**

* To practise examples of Sequence, Set and Dictionary
* To apply List and Tuple to Assignment 1 to display data points and cluster centres.

**Create a new Python project**

* Create **Week6Tutorial** project (Python)
* Add a module file and name it **io\_data\_module.py**
* There are 4 examples for Sequence, 5 for Set and 7 for Dictionary in Week 5 Lecture
* Review those examples before you answer the questions below.

**Question 1:**

* Implement a Python function (**read\_data\_to\_dict**) that opens the **iris.data** file to read all **150 data samples** and **3 class labels**. This function will output a dictionary that contains 3 key-value pairs as follows
  + **Pair 1**: key: Iris-setosa, value: list of 50 tuples, each tuple is a 4D data sample
  + **Pair 2**: key: Iris-versicolor, value: list of 50 tu ples, each tuple is a 4D data sample
  + **Pair 3**: key: Iris-virginica, value: list of 50 tuples, each tuple is a 4D data sample

**Algorithm for read\_data\_to\_dict:**

* Create a dictionary mydict = {}
* Open file, repeat the following until end of file reached:
  + Read a line
  + Split that line using separator (,)
  + Get class label at index -1 on that line, create a list mydict[class\_label] = [] (if this list already exists, do not create it again)
  + Convert other 4 values to float, add them to a list then convert this list to tuple, and add this tuple to mydict[class\_label] using append method
* Return the dictionary

**Algorithm for main program:**

* Call the function and set its return to output\_dict
* Use key to go through the dictionary, all keys are in output\_dict.keys()
* To get value of a key, use output\_dict[key]
* Note: value is a list of 50 tuples, and each tuple is a 4D data sample

**Code for read\_data\_to\_dict:**

def read\_dataset(filename, class\_label\_index, data\_sample\_start\_index, data\_sample\_end\_index):

output\_dict = {}

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

string\_list = line.split(',') #separator is a comma (,)

class\_label = string\_list[class\_label\_index]

if class\_label not in output\_dict:

output\_dict[class\_label] = []

#Convert string to float

data\_sample = [float(x) for x in string\_list[data\_sample\_start\_index:data\_sample\_end\_index+1]]

#Convert list to tuple then add to data\_list

output\_dict[class\_label].append(tuple(data\_sample))

except Exception as ex:

print(ex.args)

finally:

if f:

f.close()

return output\_dict

#end of function

**Code for main program:**

output\_dict = read\_dataset('iris.data', 4, 0, 3)

print(f'Class labels: {output\_dict.keys()}')

for key in output\_dict.keys():

print(f'Class {key}: #data samples = {len(output\_dict[key])}')

for key in output\_dict.keys():

print(f'Class label: {key}')

for index in range(len(output\_dict[key])):

print(f' Data sample {index+1}: {output\_dict[key][index]}')

**Question 2:**

* Implement a Python program in **Week6Tutorial.py** that
  1. (done in **Question 9** in **Week 5 Tutorial**) reads all 150 data samples in **iris.data** and displays them on canvas using the function developed in **Question 8** in **Week 5 Tutorial**, where **x** and **y** are the **first** and **second** values in each data sample, respectively, and
  2. (done in **Question 9** in **Week 5 Tutorial**) displays 3 centre samples centre\_1 = (5.1, 3.0, 1.1, 0.5), centre\_2 = (4.4, 3.2, 2.8, 0.2), and centre\_3 = (5.7, 3.9, 3.9, 0.8) on the same canvas with the **iris.data** data samples as seen below where red dots are iris data samples and black dots are centres.

**Algorithm for Steps 1, 2**: in Answers to Week 5 Tutorial questions

**Python code for Steps 1, 2**:

import io\_data\_module as iodata

import tkinter as tk

#Create canvas

top = tkinter.Tk()

canvas = tkinter.Canvas(top, bg="white", height=600, width=800)

#Display data

s = 150 #scale factor

r = 4 #radius

tx = -500 #x translation

ty = -200 #y translation

xi = 0

yi = 1

drawing\_list = data\_list.copy()

for sample in drawing\_list:

x = sample[xi]

y = sample[yi]

x = x\*s + tx #some values are negative so +150 is to make them positive

y = y\*s + ty

canvas.create\_oval(x-r, y-r, x+r, y+r, outline = "red", fill="red")

drawing\_list = centre\_list.copy()

for sample in drawing\_list:

x = sample[xi]

y = sample[yi]

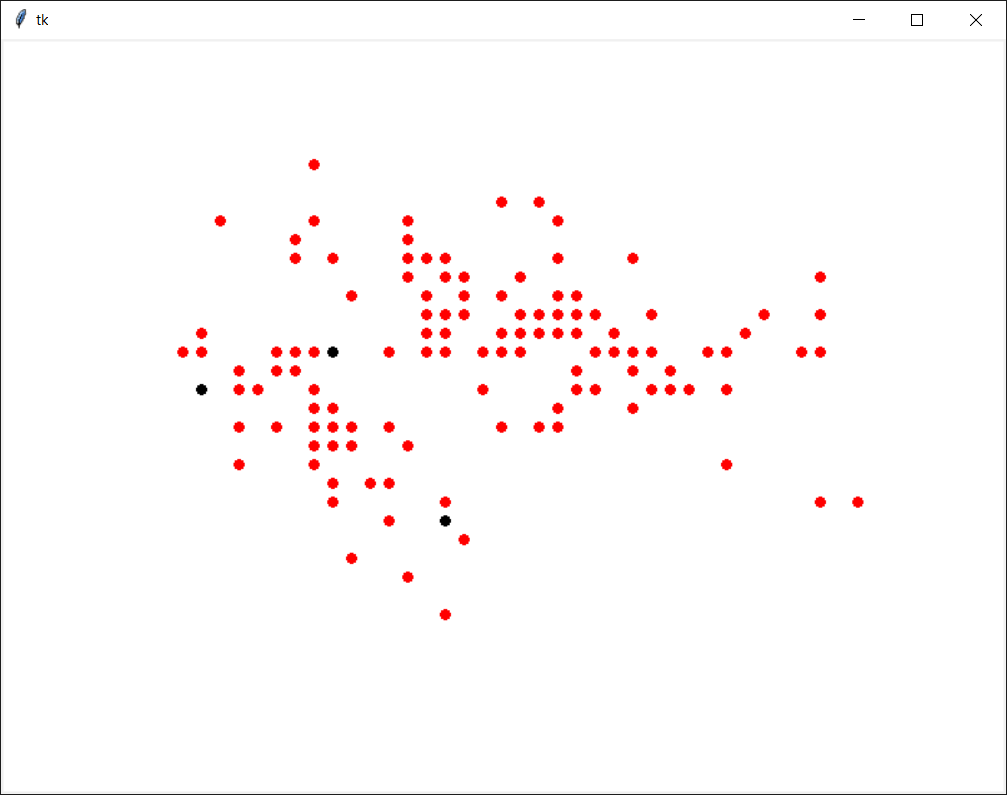
x = x\*s + tx

y = y\*s + ty

canvas.create\_oval(x-r, y-r, x+r, y+r, outline = "black", fill="black")

canvas.pack()

top.mainloop()



* 1. determines **list\_1** that contains data samples whose nearest centre is **centre\_1**
  2. determines **list\_2** that contains data samples whose nearest centre is **centre\_2**
  3. determines **list\_3** that contains data samples whose nearest centre is **centre\_3**

**Algorithm for steps 3, 4, 5:**

* Read data list
* Add 3 centres (centre\_1, centre\_2 and centre\_3) to a centre list.
* Create 3 empty lists **list\_1**, **list\_2** and **list\_3**
* For each sample in data list
  + Calculate distance between sample and all centres in centre list
  + Find nearest centre
  + If nearest centre is centre\_1, add this sample to list\_1

Else if nearest centre is centre\_2, add this sample to list\_2

Else add this sample to list\_3

**Python code for steps 3, 4, 5:**

import io\_data\_module as iodata

import tkinter as tk

centre\_1 = (5.1, 3.0, 1.1, 0.5)

centre\_2 = (4.4, 3.2, 2.8, 0.2)

centre\_3 = (5.7, 3.9, 3.9, 0.8)

centre\_list = [centre\_1, centre\_2, centre\_3]

data\_list = iodata.**read\_multi\_dim\_data\_file**('datasets/iris.data')

print(data\_list)

list\_1 = []

list\_2 = []

list\_3 = []

for sample in data\_list:

nearest\_centre = iodata.**find\_nearest\_centre**(sample, centre\_list)

if nearest\_centre == centre\_1:

list\_1.append(sample)

elif nearest\_centre == centre\_2:

list\_2.append(sample)

else:

list\_3.append(sample)

* 1. **draws lines from data samples to their nearest centre (Fig. 2 below)**

import io\_data\_module as iodata

import tkinter as tk

#Create canvas

top = tkinter.Tk()

canvas = tkinter.Canvas(top, bg="white", height=600, width=800)

#Display data

s = 150 #scale factor

r = 4 #radius

tx = -500 #x translation

ty = -200 #y translation

xi = 0

yi = 1

drawing\_list = data\_list.copy()

for sample in drawing\_list:

x = sample[xi]

y = sample[yi]

x = x\*s + tx #some values are negative so +150 is to make them positive

y = y\*s + ty

canvas.create\_oval(x-r, y-r, x+r, y+r, outline = "red", fill="red")

drawing\_list = centre\_list.copy()

all\_sample\_lists = [list\_1, list\_2, list\_3]

for k in range(len(drawing\_list)):

(x1, y1) = (drawing\_list[k][xi], drawing\_list[k][yi])

x1 = x1\*s + tx

y1 = y1\*s + ty

for i in range(len(all\_sample\_lists[k])):

(x2, y2) = (all\_sample\_lists[k][i][xi], all\_sample\_lists[k][i][yi])

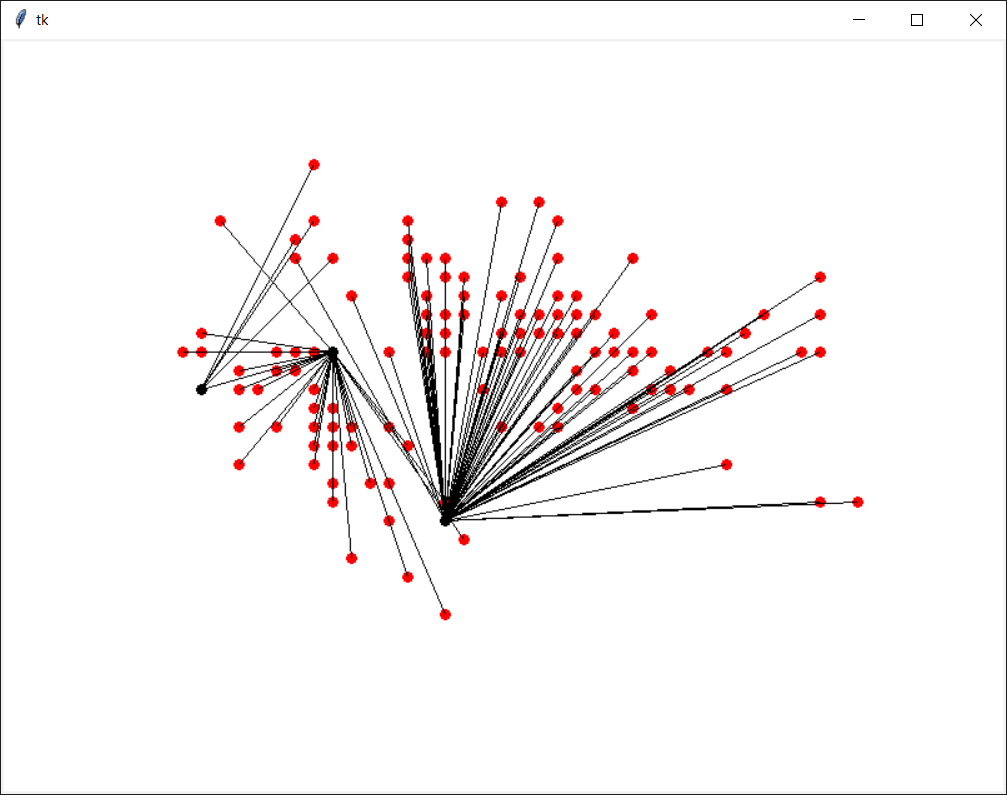
x2 = x2\*s + tx

y2 = y2\*s + ty

canvas.create\_line(x1, y1, x2, y2, fill = "black")

canvas.pack()

top.mainloop()



* 1. calculates **new\_centre\_1** that is average of data samples in **list\_1**
  2. calculates **new\_centre\_2** that is average of data samples in **list\_2**
  3. calculates **new\_centre\_3** that is average of data samples in **list\_3**

**Algorithm for Steps 7, 8, 9:**

* Input: 3 lists (**list\_1**, **list\_2** and **list\_3**) from Steps 3, 4, 5
* Create **new\_centre\_1**
* For index from 0 to 4
  + Calculate sum of sample[index] for all samples in list\_1
  + **new\_centre\_1**[index] = sum / number of samples in list\_1
* Create **new\_centre\_2**
* For index from 0 to 4
  + Calculate sum of sample[index] for all samples in list\_2
  + **new\_centre\_2**[index] = sum / number of samples in list\_2
* Create **new\_centre\_3**
* For index from 0 to 4
  + Calculate sum of sample[index] for all samples in list\_3
  + **new\_centre\_3**[index] = sum / number of samples in list\_3

**Python code for Steps 7, 8, 9:**

new\_centre\_1 = [0,0,0,0]

for index in range(4): #range(len(list\_1[0])):

sum = 0

for sample in list\_1:

sum += sample[index]

new\_centre\_1[index] = sum / len(list\_1)

print(new\_centre\_1)

new\_centre\_2 = [0,0,0,0]

for index in range(4): #range(len(list\_1[0])):

sum = 0

for sample in list\_2:

sum += sample[index]

new\_centre\_2[index] = sum / len(list\_2)

print(new\_centre\_2)

new\_centre\_3 = [0,0,0,0]

for index in range(4): #range(len(list\_1[0])):

sum = 0

for sample in list\_3:

sum += sample[index]

new\_centre\_3[index] = sum / len(list\_3)

print(new\_centre\_3)

* 1. **draws lines from data samples to their nearest new centre (Fig. 3 below)**

#Create canvas

top = tkinter.Tk()

canvas = tkinter.Canvas(top, bg="white", height=600, width=800)

#Display data

s = 150 #scale factor

r = 4 #radius

tx = -500 #x translation

ty = -200 #y translation

xi = 0

yi = 1

drawing\_list = data\_list.copy()

for sample in drawing\_list:

x = sample[xi]

y = sample[yi]

x = x\*s + tx #some values are negative so +150 is to make them positive

y = y\*s + ty

canvas.create\_oval(x-r, y-r, x+r, y+r, outline = "red", fill="red")

drawing\_list = new\_centre\_list.copy()

all\_sample\_lists = [list\_1, list\_2, list\_3]

for k in range(len(drawing\_list)):

(x1, y1) = (drawing\_list[k][xi], drawing\_list[k][yi])

x1 = x1\*s + tx

y1 = y1\*s + ty

for i in range(len(all\_sample\_lists[k])):

(x2, y2) = (all\_sample\_lists[k][i][xi], all\_sample\_lists[k][i][yi])

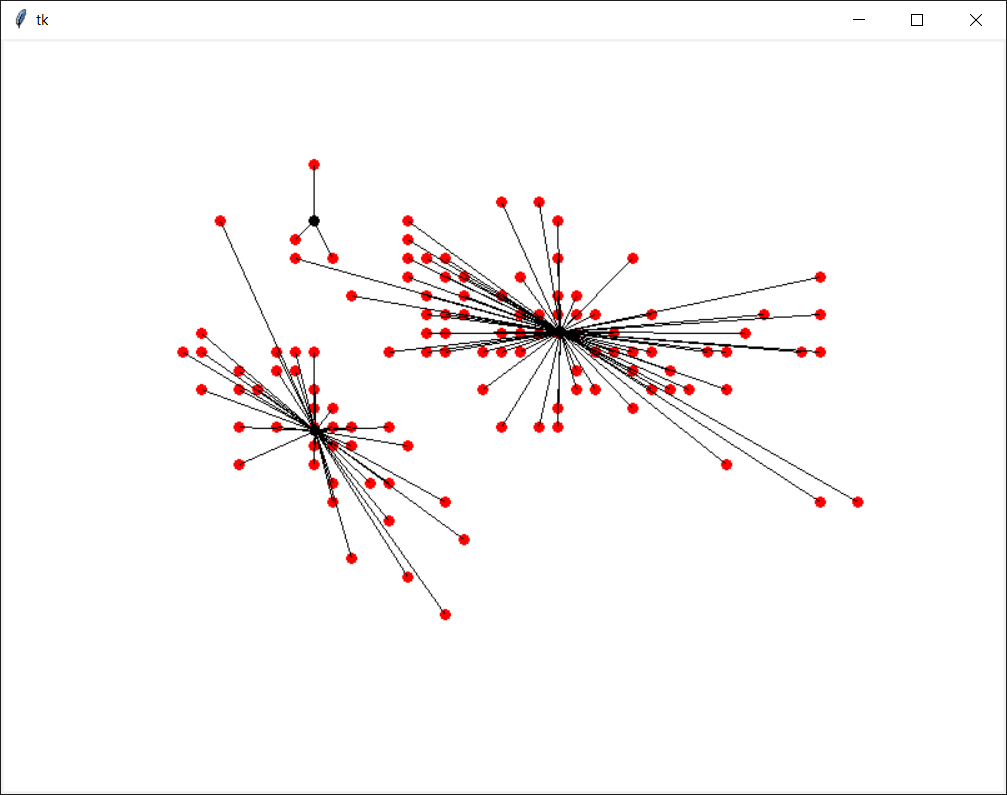
x2 = x2\*s + tx

y2 = y2\*s + ty

canvas.create\_line(x1, y1, x2, y2, fill = "black")

canvas.pack()

top.mainloop()



**Question 3:**

* Use the same program in Question 2 and change dataset from **iris.data** to **ellipse1.txt** and change 3 centres to 2 centres **centre\_1 = (2.036779, 2.896883)** and **centre\_2 = (2.836779, 3.896883)** and run the program again (skip step 5 for list\_3 and step 9 for new\_centre\_3). Below are the outputs.

import io\_data\_module as iodata

import tkinter

#Open file and read data

data\_list = iodata.read\_multi\_dim\_data\_file('ellipse1.txt')

print(len(data\_list))

#For ellipse1

centre\_1 = (2.036779, 2.896883)

centre\_2 = (2.836779, 3.896883)

centre\_list = [centre\_1, centre\_2]

#Create canvas

top = tkinter.Tk()

canvas = tkinter.Canvas(top, bg="white", height=600, width=800)

#Display data

s = 70 #scale factor

r = 4 #radius

tx = 150 #x translation

ty = 150 #y translation

xi = 0

yi = 1

drawing\_list = data\_list.copy()

for sample in drawing\_list:

x = sample[xi]

y = sample[yi]

x = x\*s + tx #some values are negative so +150 is to make them positive

y = y\*s + ty

canvas.create\_oval(x-r, y-r, x+r, y+r, outline = "red", fill="red")

drawing\_list = centre\_list.copy()

for sample in drawing\_list:

x = sample[xi]

y = sample[yi]

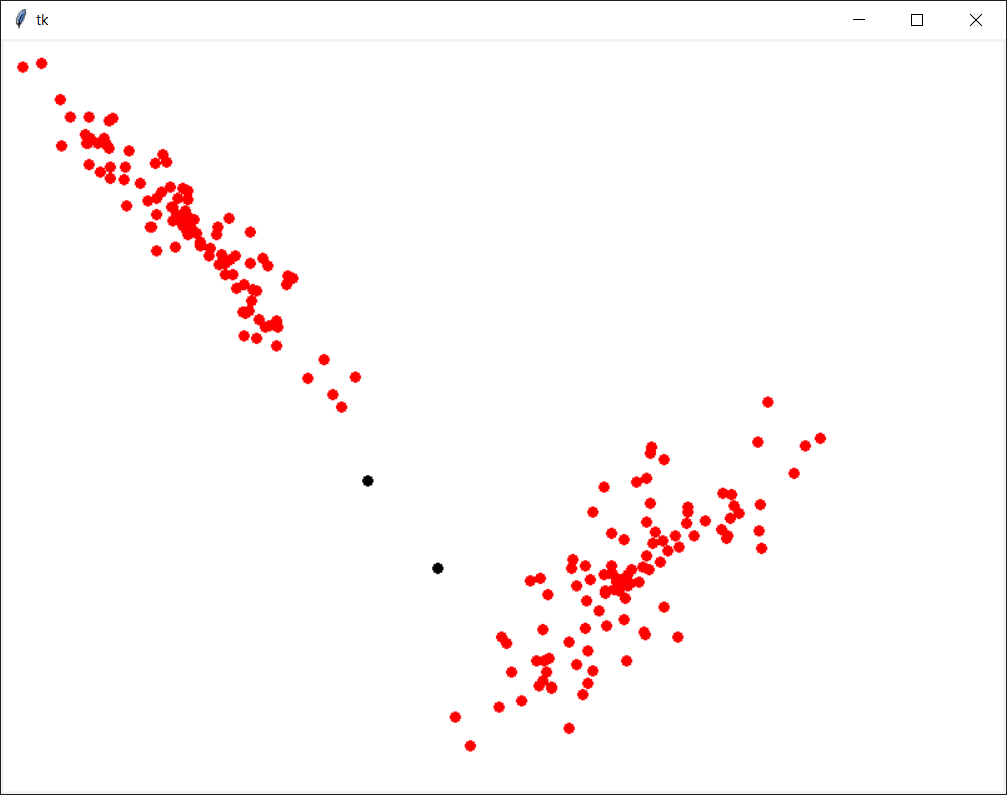
x = x\*s + tx #some values are negative so +150 is to make them positive

y = y\*s + ty

canvas.create\_oval(x-r, y-r, x+r, y+r, outline = "black", fill="black")

canvas.pack()

top.mainloop()



import io\_data\_module as iodata

import tkinter

#Open file and read data

data\_list = iodata.read\_multi\_dim\_data\_file('ellipse1.txt')

print(len(data\_list))

#For ellipse1

centre\_1 = (2.036779, 2.896883)

centre\_2 = (2.836779, 3.896883)

centre\_list = [centre\_1, centre\_2]

#Week 6 Tutorial

list\_1 = []

list\_2 = []

for sample in data\_list:

nearest\_centre = iodata.find\_nearest\_centre(sample, centre\_list)

if nearest\_centre == centre\_1:

list\_1.append(sample)

elif nearest\_centre == centre\_2:

list\_2.append(sample)

all\_sample\_lists = [list\_1, list\_2]

new\_centre\_1 = [0,0]

for index in range(2): #range(len(list\_1[0])):

sum = 0

for sample in list\_1:

sum += sample[index]

new\_centre\_1[index] = sum / len(list\_1)

print(new\_centre\_1)

new\_centre\_2 = [0,0]

for index in range(2): #range(len(list\_1[0])):

sum = 0

for sample in list\_2:

sum += sample[index]

new\_centre\_2[index] = sum / len(list\_2)

print(new\_centre\_2)

new\_centre\_list = [new\_centre\_1, new\_centre\_2]

#Create canvas

top = tkinter.Tk()

canvas = tkinter.Canvas(top, bg="white", height=600, width=800)

#Display data

s = 70 #scale factor

r = 4 #radius

tx = 150 #x translation

ty = 150 #y translation

xi = 0

yi = 1

drawing\_list = data\_list.copy()

for sample in drawing\_list:

x = sample[xi]

y = sample[yi]

x = x\*s + tx #some values are negative so +150 is to make them positive

y = y\*s + ty

canvas.create\_oval(x-r, y-r, x+r, y+r, outline = "red", fill="red")

#Week 6 Tutorial

drawing\_list = centre\_list.copy()

for k in range(len(drawing\_list)):

(x1, y1) = (drawing\_list[k][xi], drawing\_list[k][yi])

x1 = x1\*s + tx

y1 = y1\*s + ty

for i in range(len(all\_sample\_lists[k])):

(x2, y2) = (all\_sample\_lists[k][i][xi], all\_sample\_lists[k][i][yi])

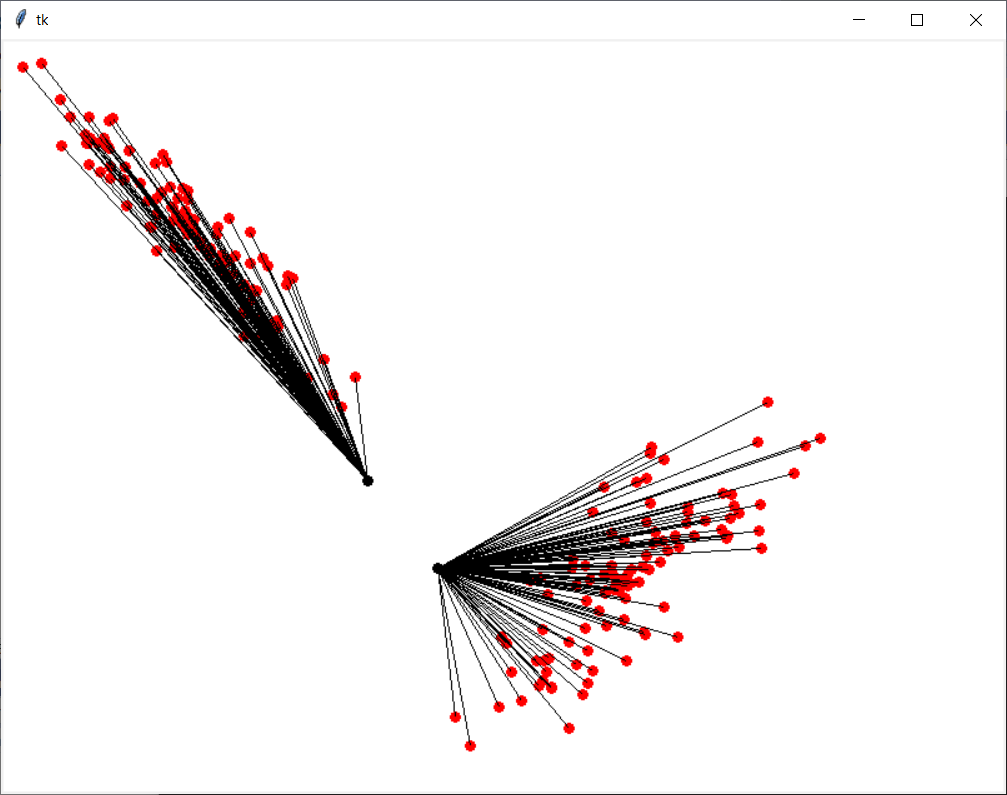
x2 = x2\*s + tx

y2 = y2\*s + ty

canvas.create\_line(x1, y1, x2, y2, fill = "black")

canvas.pack()

top.mainloop()



import io\_data\_module as iodata

import tkinter

#Open file and read data

data\_list = iodata.read\_multi\_dim\_data\_file('ellipse1.txt')

print(len(data\_list))

#For ellipse1

centre\_1 = (2.036779, 2.896883)

centre\_2 = (2.836779, 3.896883)

centre\_list = [centre\_1, centre\_2]

#Week 6 Tutorial

list\_1 = []

list\_2 = []

for sample in data\_list:

nearest\_centre = iodata.find\_nearest\_centre(sample, centre\_list)

if nearest\_centre == centre\_1:

list\_1.append(sample)

elif nearest\_centre == centre\_2:

list\_2.append(sample)

all\_sample\_lists = [list\_1, list\_2]

new\_centre\_1 = [0,0]

for index in range(2): #range(len(list\_1[0])):

sum = 0

for sample in list\_1:

sum += sample[index]

new\_centre\_1[index] = sum / len(list\_1)

print(new\_centre\_1)

new\_centre\_2 = [0,0]

for index in range(2): #range(len(list\_1[0])):

sum = 0

for sample in list\_2:

sum += sample[index]

new\_centre\_2[index] = sum / len(list\_2)

print(new\_centre\_2)

new\_centre\_list = [new\_centre\_1, new\_centre\_2]

#Create canvas

top = tkinter.Tk()

canvas = tkinter.Canvas(top, bg="white", height=600, width=800)

#Display data

s = 70 #scale factor

r = 4 #radius

tx = 150 #x translation

ty = 150 #y translation

xi = 0

yi = 1

drawing\_list = data\_list.copy()

for sample in drawing\_list:

x = sample[xi]

y = sample[yi]

x = x\*s + tx #some values are negative so +150 is to make them positive

y = y\*s + ty

canvas.create\_oval(x-r, y-r, x+r, y+r, outline = "red", fill="red")

#Week 6 Tutorial

drawing\_list = new\_centre\_list.copy()

for k in range(len(drawing\_list)):

(x1, y1) = (drawing\_list[k][xi], drawing\_list[k][yi])

x1 = x1\*s + tx

y1 = y1\*s + ty

for i in range(len(all\_sample\_lists[k])):

(x2, y2) = (all\_sample\_lists[k][i][xi], all\_sample\_lists[k][i][yi])

x2 = x2\*s + tx

y2 = y2\*s + ty

canvas.create\_line(x1, y1, x2, y2, fill = "black")

canvas.pack()

top.mainloop()

