University of Canberra

Faculty of Science and Technology

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

**Week 10 Tutorial**

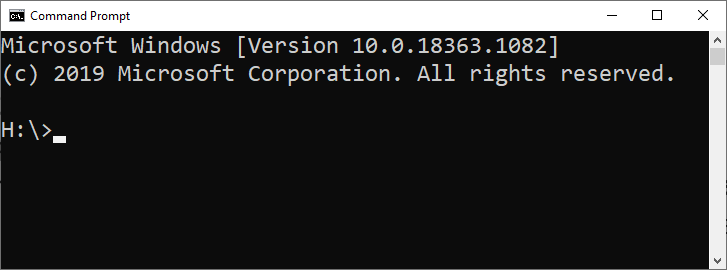
**Python packages: NumPy, Matplotlib and Scikit-learn**

**Objectives**

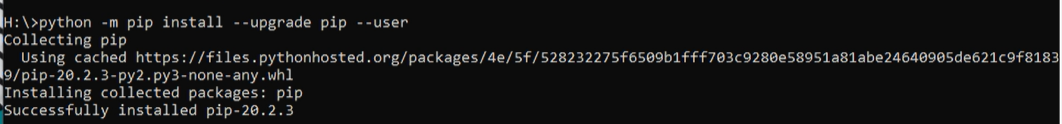
* To install and use Python packages: NumPy, Matplotlib and Scikit-learn.
* To learn NumPy array and Matplotlib for data visualisation.

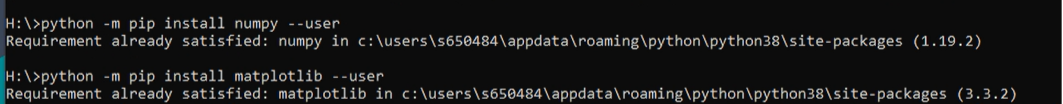
**Python Environments**

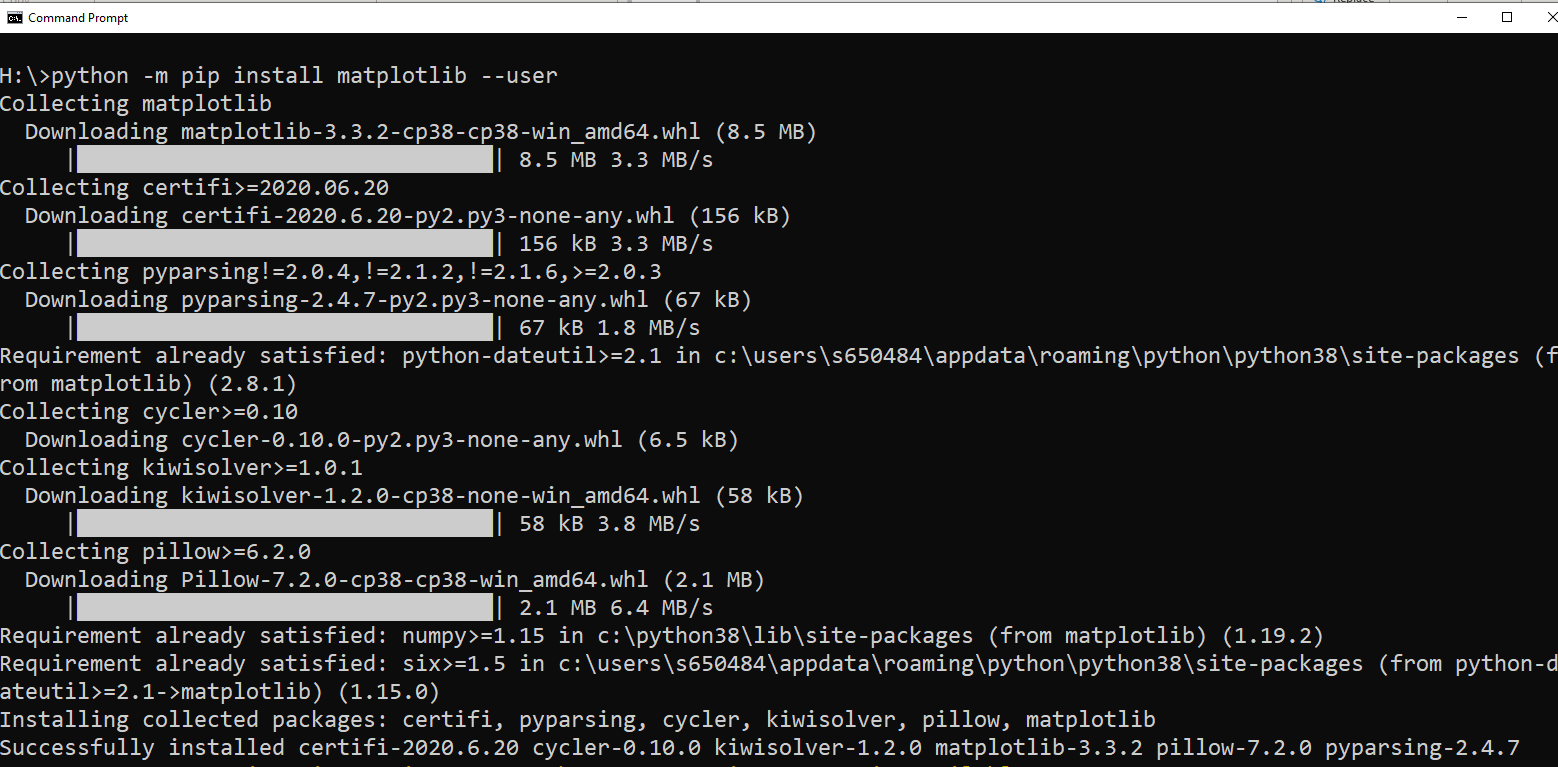
* If your **Python** is not 3.8 or newer, please upgrade to this version.
* Install packages (you can use other methods to install packages):
  + Enter **cmd** to the search box in bottom-left corner in your computer to open Command window

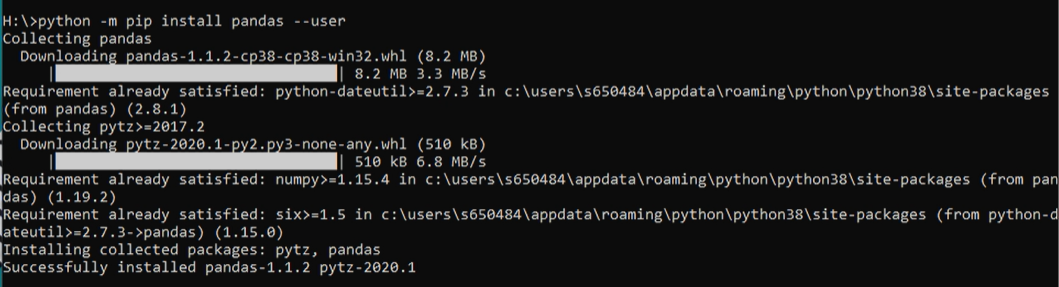
 

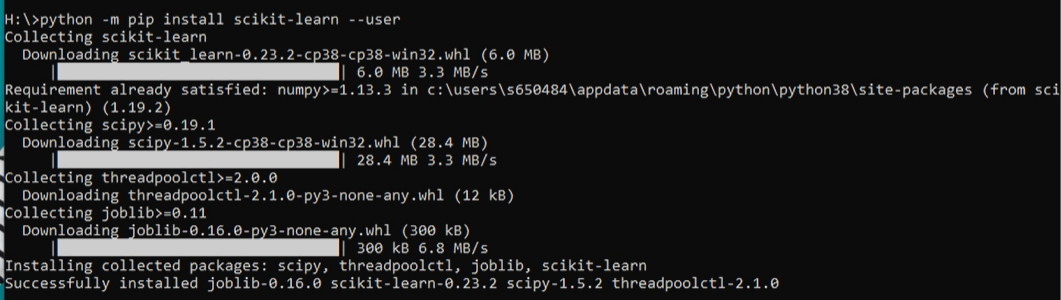
* + Upgrade **pip**: Enter **python -m pip install –upgrade pip --user**
  + Install **Numpy**: Enter **python -m pip install numpy --user**
  + Install **Matplotlib**: Enter **python -m pip install matplotlib --user**
  + Install **Pandas**: Enter **python -m pip install pandas --user**
  + Install **Scikit-learn**: Enter **python -m pip install scikit-learn --user**



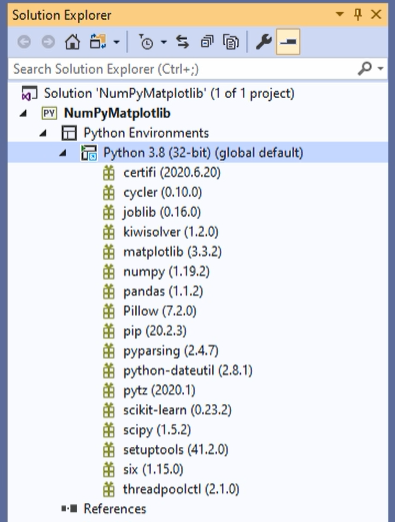








* Open **Visual Studio** (**PyCharm**, **VS Code** or **Spyder**)
* Create a new Python project and name it **NumPyMatplotlib**
* Click **Python Environments** then **Python 3.8** in **Solution Explorer** to see all installed packages



**NumPy Package**

**Example 1:**

* Add the following to the Python file in your project to start using Python

import numpy as np

* Then add the following to convert a list to NumPy array

#Example 1

list0 = [1, 2, 3]

array0 = np.array(list0)

print(array0)

* The output when you run this program is [1 2 3]

**Question 1:**

* Add the following lists to your program

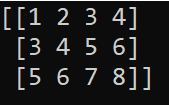
#Question 1

list1 = [1, 2, 3]

list2 = [4, 5, 6]

list3 = [7, 8, 9]

* Then write your code to convert them to array and have the following output



**Example 2:**

* Add the following code to your program to convert a list of lists to an array (matrix) then print the first column in that matrix.

#Example 2

mylist = [[1, 2, 3, 4], [3, 4, 5, 6], [5, 6, 7, 8]]

myarray = np.array(mylist)

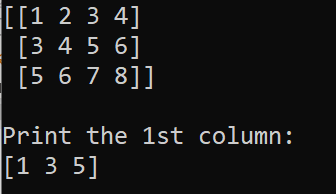
print(myarray)

print()

print('Print the 1st column: ')

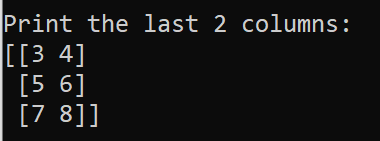
print(myarray[:, 0])

* The output when you run this code is



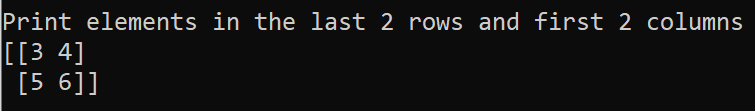
**Question 2:**

* Write your code to print the last 2 columns of the array in Example 2



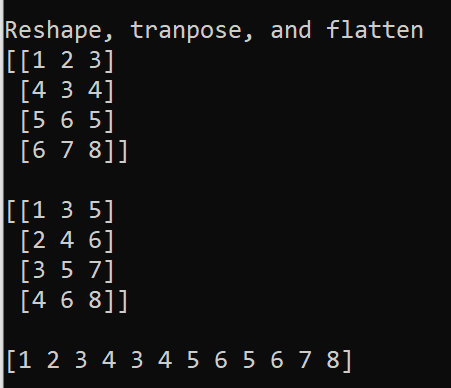
**Question 3:**

* Write your code to print elements in the last 2 rows and first 2 columns of the array in Example 2



**Question 4:**

* Write your code using **reshape**, **transpose**, and **flatten** functions (in Week 9 lecture) on the array in Example 2 to produce the following outputs



**Random functions:**

* You will use this NumPy package to generate random numbers.
* The **np.random.random** function generates random numbers between 0 (inclusive) and 1 (exclusive).
* The following random functions are available in NumPy
  + **rand**(d0, d1, ..., dn) Random values in a given shape.
  + **randn**(d0, d1, ..., dn) Return a sample (or samples) from the “standard normal” distribution.
  + **randint**(low, high=None, size=None, dtype='l') Return random integers from low (inclusive) to high (exclusive). Return random integers from the “discrete uniform” distribution of the specified dtype in the “half-open” interval [low, high). If high is None (the default), then results are from [0, low).
  + **random\_integers**(low, high, size) Random integers of type np.int between low and high, inclusive.
  + **random\_sample**(size) Return random floats in the half-open interval [0.0, 1.0).
  + **random**(size) Return random floats in the half-open interval [0.0, 1.0).
  + **ranf**(size) Return random floats in the half-open interval [0.0, 1.0).

**Example 3:**

* Add the following to call the **random** function

print(np.random.random())

* Run that code to output a random number, for example (you will not get the same number)



* Change random() to random(5) then run. You will have a list of 5 numbers between 0 and 1.

print(np.random.random(5))



* *Remember the function call* ***np.random.random(N)*** *with integer N will output N random numbers between 0 and 1*.

**Example 4:**

* Add the following to call the **rand** function in different shapes

print('rand function')

print('np.random.rand(2, 3)')

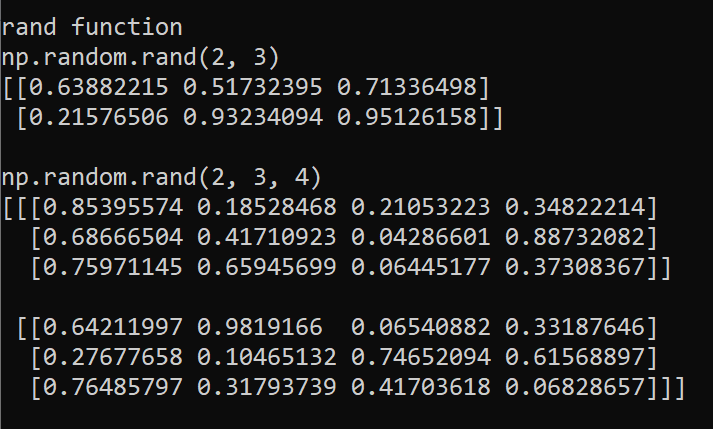
print(np.random.rand(2, 3))

print()

print('np.random.rand(2, 3, 4)')

print(np.random.rand(2, 3, 4))

print()



**Example 5:**

* Add the following to call the **randn** function in different shapes

print('randn function')

print('np.random.randn(2, 3)')

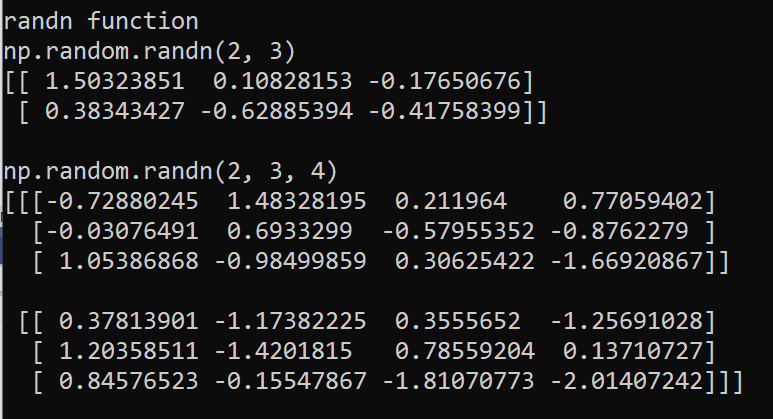
print(np.random.randn(2, 3))

print()

print('np.random.randn(2, 3, 4)')

print(np.random.randn(2, 3, 4))

print()



**Example 6:**

* Add the following code to call the **randint** function with different input parameters

print('randint function')

print('np.random.randint(low=4)')

print(np.random.randint(low=4))

print()

print('np.random.randint(low=4, high=10)')

print(np.random.randint(low=4, high=10))

print()

print('np.random.randint(low=3, high=10, size=5)')

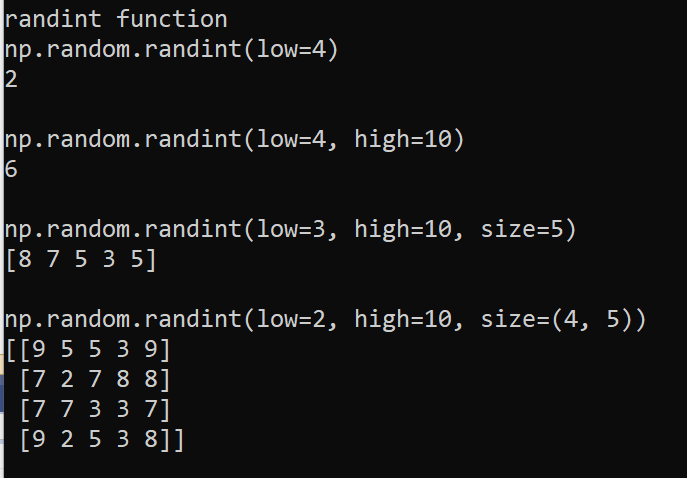
print(np.random.randint(low=3, high=10, size=5))

print()

print('np.random.randint(low=2, high=10, size=(4, 5))')

print(np.random.randint(low=2, high=10, size=(4, 5)))

print()



**Example 7:**

* Add the following code to call the **random\_integers , random\_sample** , **random** , and **ranf** functions

print('np.random.random\_integers(low=2, high=10, size=(4, 5))')

print(np.random.random\_integers(low=2, high=10, size=(4, 5)))

print()

print('np.random.random\_sample(size=(4, 5))')

print(np.random.random\_sample(size=(4, 5)))

print()

print('np.random.random(size=(4, 5))')

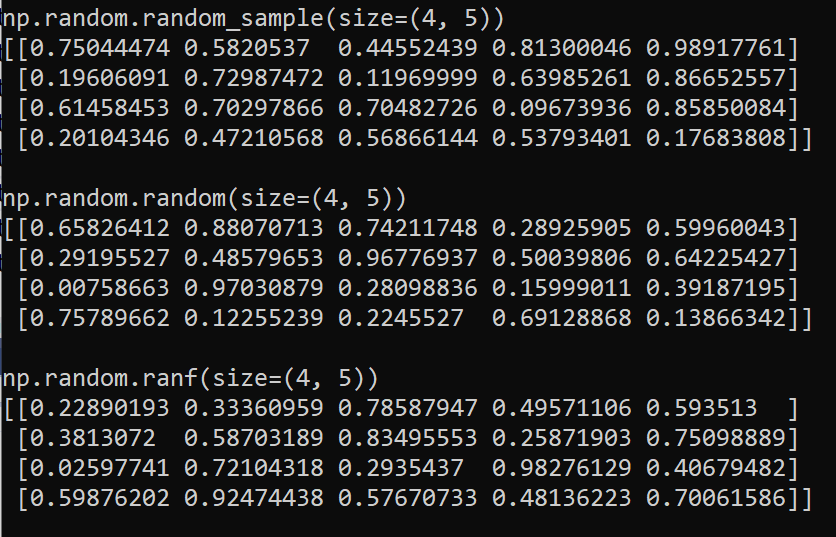
print(np.random.random(size=(4, 5)))

print()

print('np.random.ranf(size=(4, 5))')

print(np.random.ranf(size=(4, 5)))

print()



**Matplotlib Package**

* The **plot** function (<https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.plot.html>):
  + **plot(x, y)** # plot x and y using default line style and colour
  + **plot(x, y, 'bo')** # plot x and y using blue circle markers
  + **plot(y)** # plot y using x as index array 0..N-1
  + **plot(y, 'r+')** # ditto, but with red plusses

**Example 8**: Basic plot with x and y values are from a *polar* coordinate function.

* Add the following to your program

import numpy as np

import matplotlib.pyplot as plt

x = []

y = []

#Generate 200 values between 0 and 2pi

values = np.linspace(0, 2\*np.pi, 100)

for a in values:

r = np.cos(20\*a)

x.append(r \* np.cos(a))

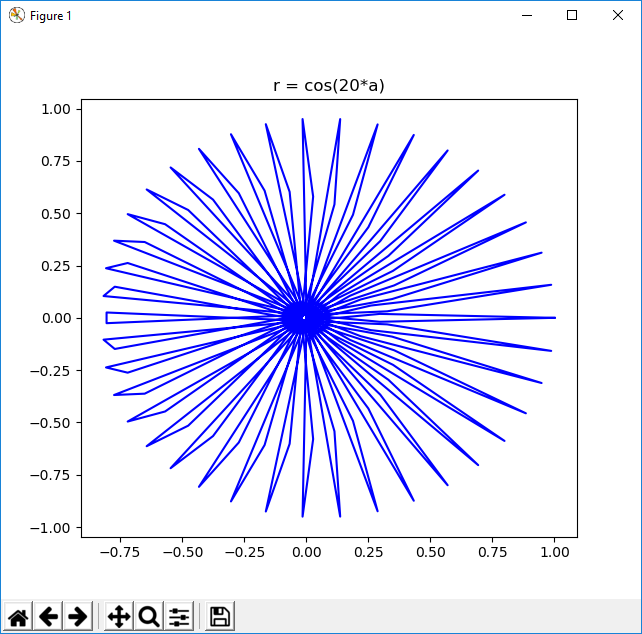
y.append(r \* np.sin(a))

plt.plot(x, y, c='b')

plt.title('r = cos(20\*a)')

plt.show()

* Run your program to have the following output



**Example 9**: Apply a loop to Example 8 to have multiple outputs

* Enter the following to your program

#generate 0, 50, 100, 150, 200, 250

for n in np.arange(0, 300, 50):

x = []

y = []

#Generate 200 values between 0 and 2pi

values = np.linspace(0, 2\*np.pi, 100)

for a in values:

r = 1 + np.cos(n\*a)

x.append(r \* np.cos(a))

y.append(r \* np.sin(a))

plt.plot(x, y, c='b')

plt.title('r = cos(n\*a)')

plt.show()

* Run your program to have 5 outputs (close the first plot to display the second and so on)

**Example 10**: The Spirograph

* Enter the following to your program

#Run 10 times

for m in range(10):

#Spirograph

R = 10

k = np.random.random()

l = np.random.random()

x = []

y = []

values = np.linspace(0, 200, 400)

for t in values:

xx = R \* ((1 - k) \* np.cos(t) + l \* k \* np.cos((1-k)\*t/k))

x.append(xx)

yy = R \* ((1 - k) \* np.sin(t) - l \* k \* np.sin((1-k)\*t/k))

y.append(yy)

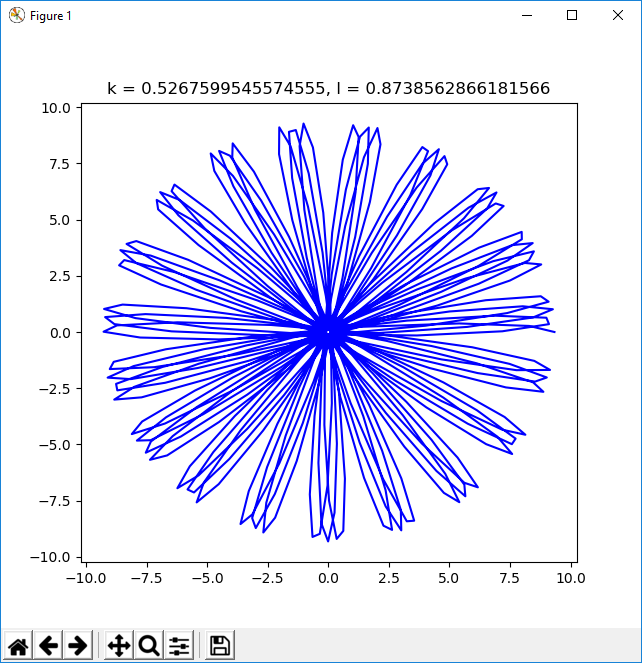
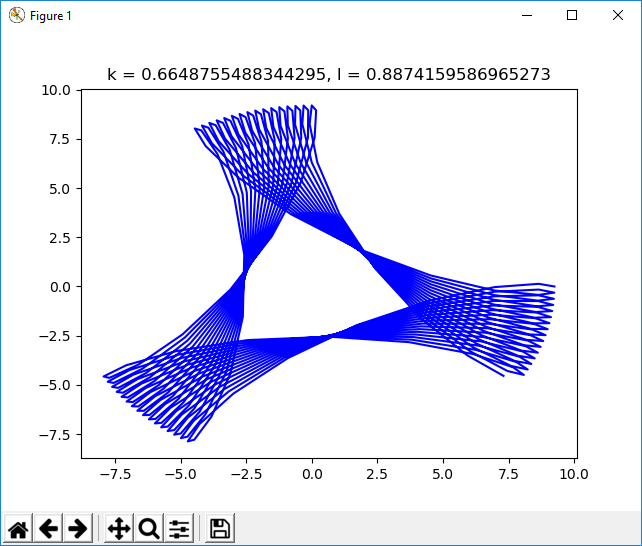
#end spirograph

plt.plot(x, y, c='b')

plt.title(f'k = {k}, l = {l}')

plt.show()

* Run your program to have 10 outputs (close the first plot to display the second and so on). Below are 2 outputs.

**Example 11**: Multiple plots

* Enter the following to your program

# generate 0, 2, 4, 6, ..., 48

t = np.arange(0, 50, 2)

x1 = t

y1 = t\*\*0.5

s1 = 'rs' #red square

x2 = t

y2 = t

s2 = 'b--' #blue dash

x3 = t

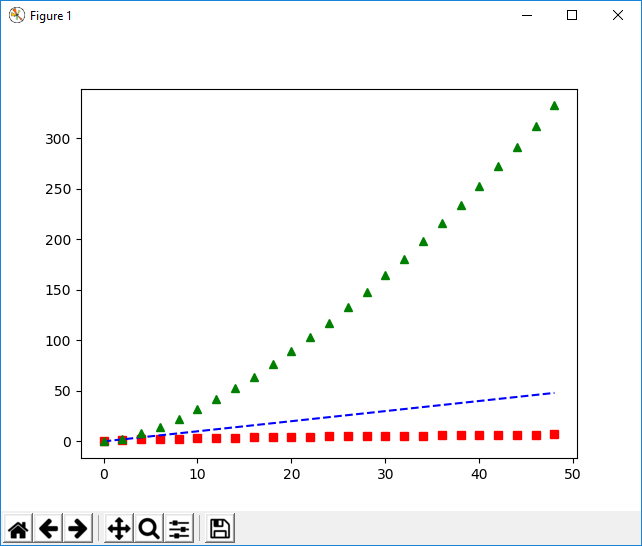
y3 = t\*\*1.5

s3 = 'g^' #green triangle

plt.plot(x1, y1, s1, x2, y2, s2, x3, y3, s3)

plt.show()

* Run your program to have the following output



**Example 12**: Use Scatter function. The input data is a dictionary with 4 keys (a, b, c, d) and values are 1-dimensional np.arrays.

* Enter the following to your program

dict = {'a': np.arange(50), #array of 50 integers 0 1 2 3 4 ...

'b': np.random.randn(50), #array of 50 random values

'c': np.random.random(50)} # array of 50 random numbers between 0 and 1

dict['d'] = dict['a'] + 10 \* np.random.randn(50)

dict['d'] = np.abs(dict['d']) \* 50

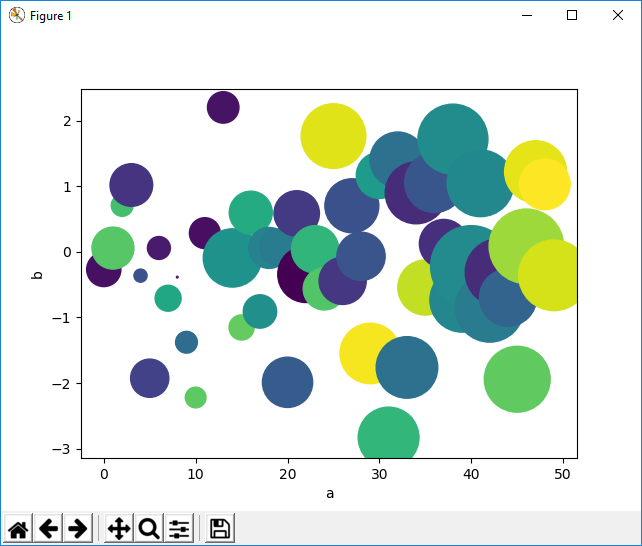
plt.scatter('a', 'b', c='c', s='d', data=dict)

plt.xlabel('a')

plt.ylabel('b')

plt.show()

* Run your program to have the following output



**Example 13**: Plotting with categorical variables.

* Enter the following to your program

items = ['A1', 'A2', 'FE']

marks = [15, 10, 60]

plt.figure(figsize=(10, 4))

plt.subplot(131) #1 row, 3 columns, 1st col

plt.bar(items, marks)

plt.subplot(132) #2nd col

plt.scatter(items, marks)

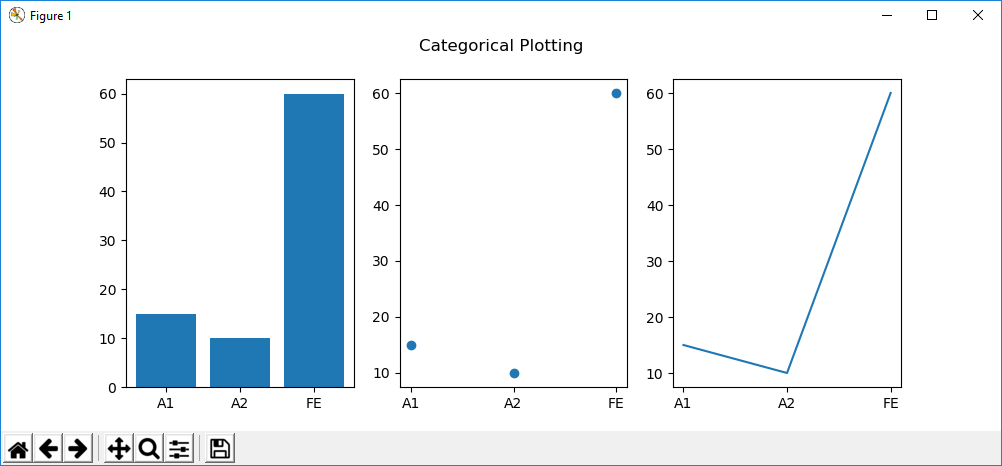
plt.subplot(133) #3rd col

plt.plot(items, marks)

plt.suptitle('Categorical Plotting')

plt.show()

* Run your program to have the following output



**Example 14**: Iris dataset. The program below loads Iris dataset. The rows are 4-dimensional samples and the columns are features: Sepal Length, Sepal Width, Petal Length and Petal Width.

* Enter the following to your program

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')

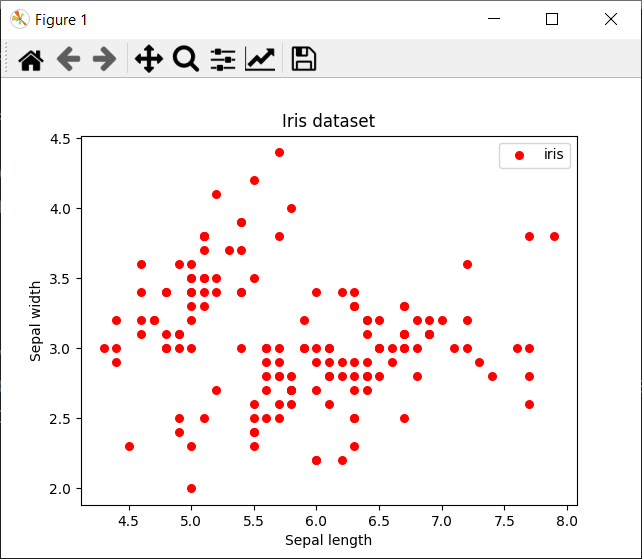
plt.ylabel('Sepal width')

plt.title('Iris dataset')

plt.legend()

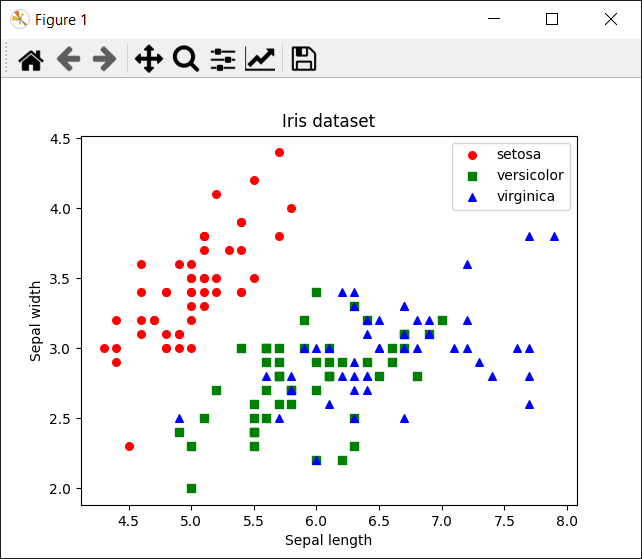
plt.show()

* Run your program to have the following output



**Question 5**: Iris dataset. Modify the program in **Example 14** to load data samples in different colours (**red**, **green**, and **blue**) and different markers (**circle**, **square**, and **triangle**) for 3 data classes (**setosa**, **versicolour**, and **virginica**). The first 50 samples are setosa, the next 50 samples are versicolour, and the last 50 samples are virginica.

.



**Total mark for assessment: 3%. Complete and submit all questions: 2%, and lab attendance: 1%. Submit after 23:59 (midnight): -0.5% and -0.5% for each day after.**

.