Department of Computer Technology and Information Systems

CTIS264 – Computer Algorithms

Spring 2019 - 2020

Lab Guide 10 - Week 13

Instructor	: Füsun YÜRÜTEN
Assistant	: Leyla SEZER
OBJECTIVE:	MatPlotLib Module

About **Matplotlib** Module**:**

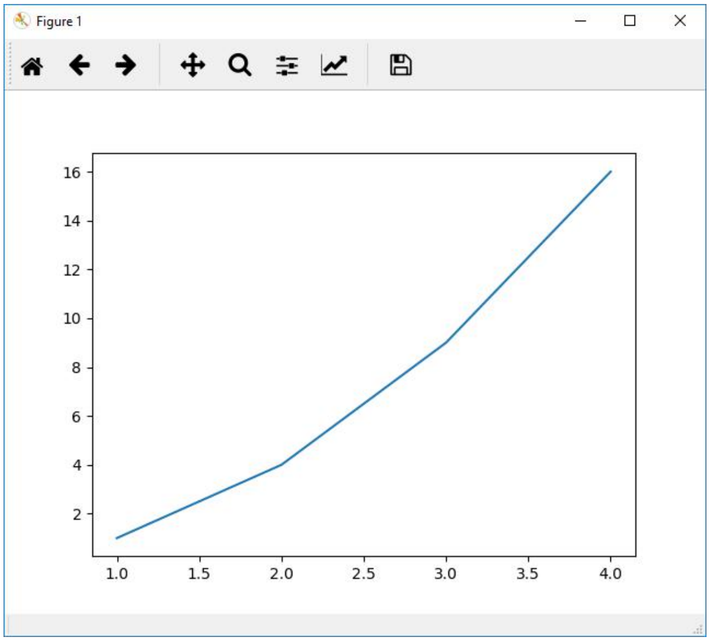
- Matplotlib is a Python plotting library, which is inspired by the MATLAB plotting functionality, so the commands used are similar to those used in MATLAB.
- The pyplot package contains a number of commands used for creating and formatting plots in Python.
- Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels.

Basic Plotting Functions;

- The plot function has different forms, depending on the input arguments.
- If y is a list, plot(y) produces a piecewise graph of the elements of (y) versus the index of the elements of (y).
- Since Python ranges start with 0, the default x list has the same length as y but starts with 0. Hence the x data are [0,1,2,3].
- If you specify two lists as arguments, plot(x,y) produces a graph of y versus x.
- Both lists <u>must</u> have the same number of elements

For example, to plot x versus y, you can issue the command:

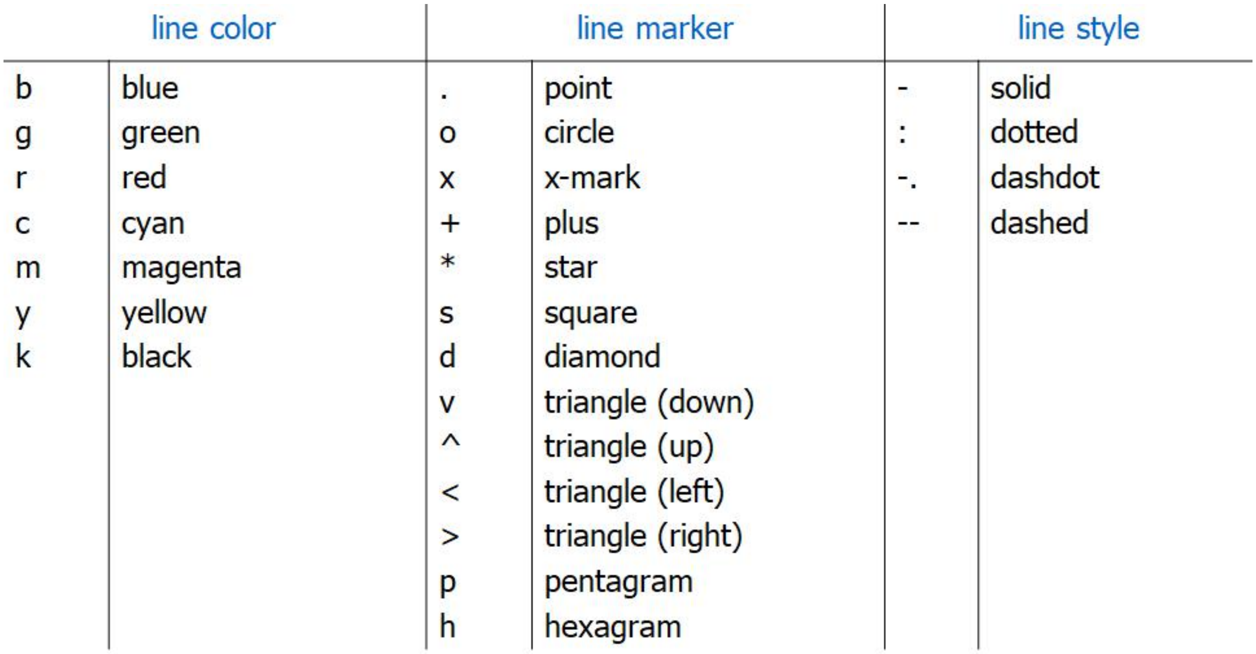
- plot([1, 2, 3, 4], [1, 4, 9, 16])



Creating plots with pyplot;

- For every x, y pair of arguments, there is an optional third argument which is the format string that indicates the color and line type of the plot.
- The letters and symbols of the format string are from MATLAB, and you concatenate a color string with a line style string.
- The default format string is 'b-', which is a solid blue line. For example, to plot the above with red circles, you would issue the command:

  plot([1,2,3,4], [1,4,9,16], 'ro')



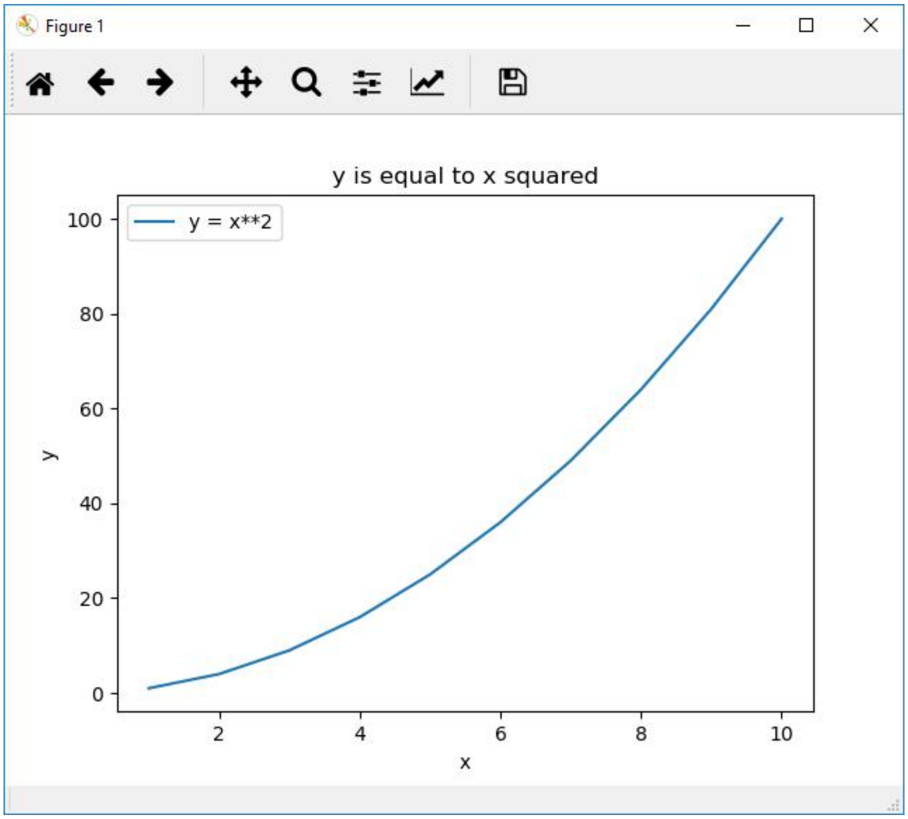
line color		line marker		line style	
b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed
m	magenta	*	star		
y	yellow	s	square		
k	black	d	diamond		
		v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagram		
		h	hexagram		

Formatting Plots with pyplot;

- The result of a plot function call is not a finished product, since there are no titles, axis labels, or grid lines on the plot.
- The details can be added with the functions:
  - title,
  - xlabel,
  - ylabel,
  - grid
  - legend
  - axis

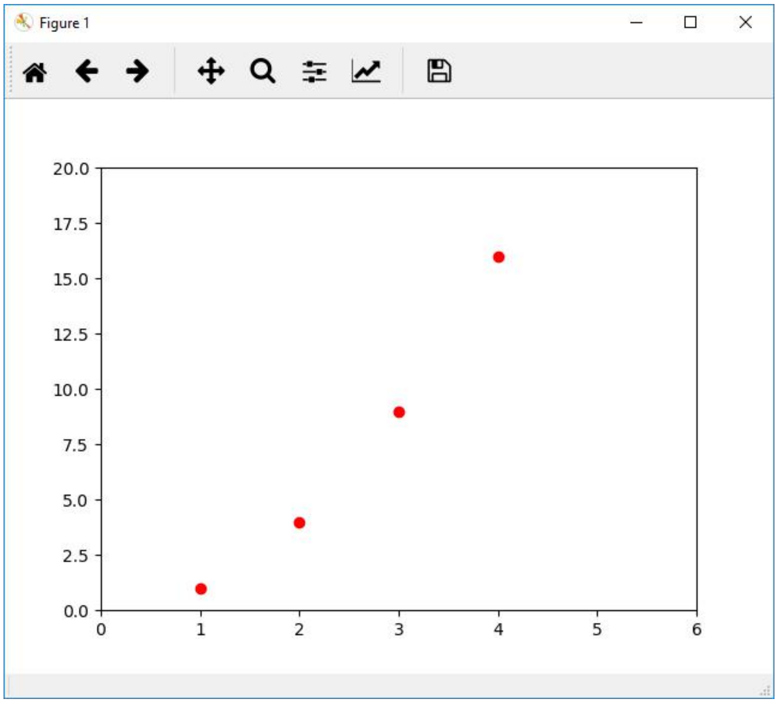
Formatting with pyplot – numpy arrays;

```
import matplotlib.pyplot as plt
import numpy as np
plt.clf()
x = np.arange(1,11)
y = x**2
plt.plot(x,y)
plt.xlabel('x')
plt.ylabel('y')
plt.title('y is equal to x squared')
plt.legend(['y = x**2'])
plt.show()
```



Creating plots with pyplot – axis;

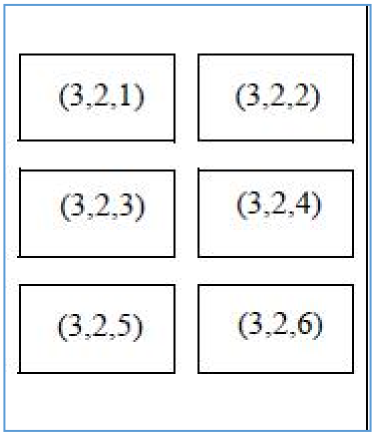
```
import matplotlib.pyplot as plt
#clears the current figure window
plt.clf()

#creates the plot with x, y values red #circle markers
plt.plot([1,2,3,4], [1,4,9,16], 'ro')

#changes  the  limits  of  the  x  and  y
#axis
#[xmin, xmax, ymin, ymax]

plt.axis([0, 6, 0, 20])
plt.show()
```



Displaying Multiple Plots in One Figure – subplot();

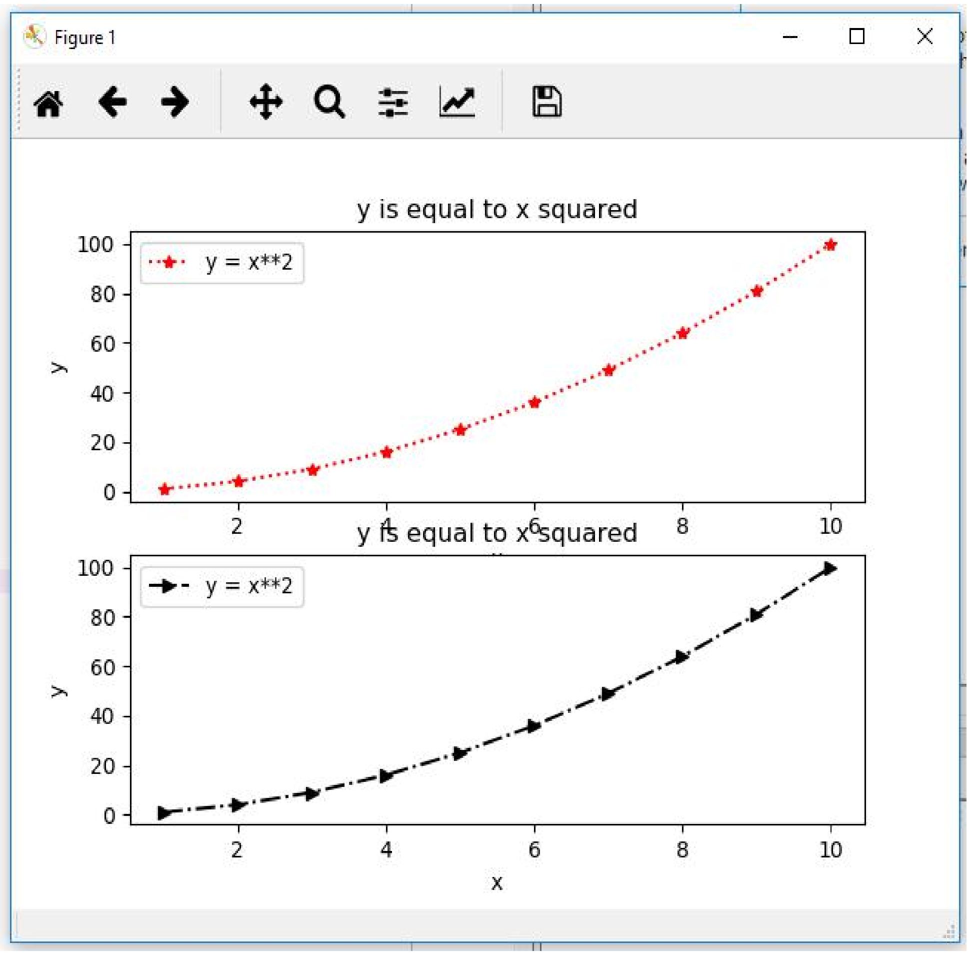
```
subplot(m,n,p)
```

- This splits the figure window into an m-by-n matrix of small subplots and selects the pth subplot for the current plot.
- For example, the command subplot(3,2,1) creates six areas arranged in three rows and two columns as shown, and makes the upper left subplot current.



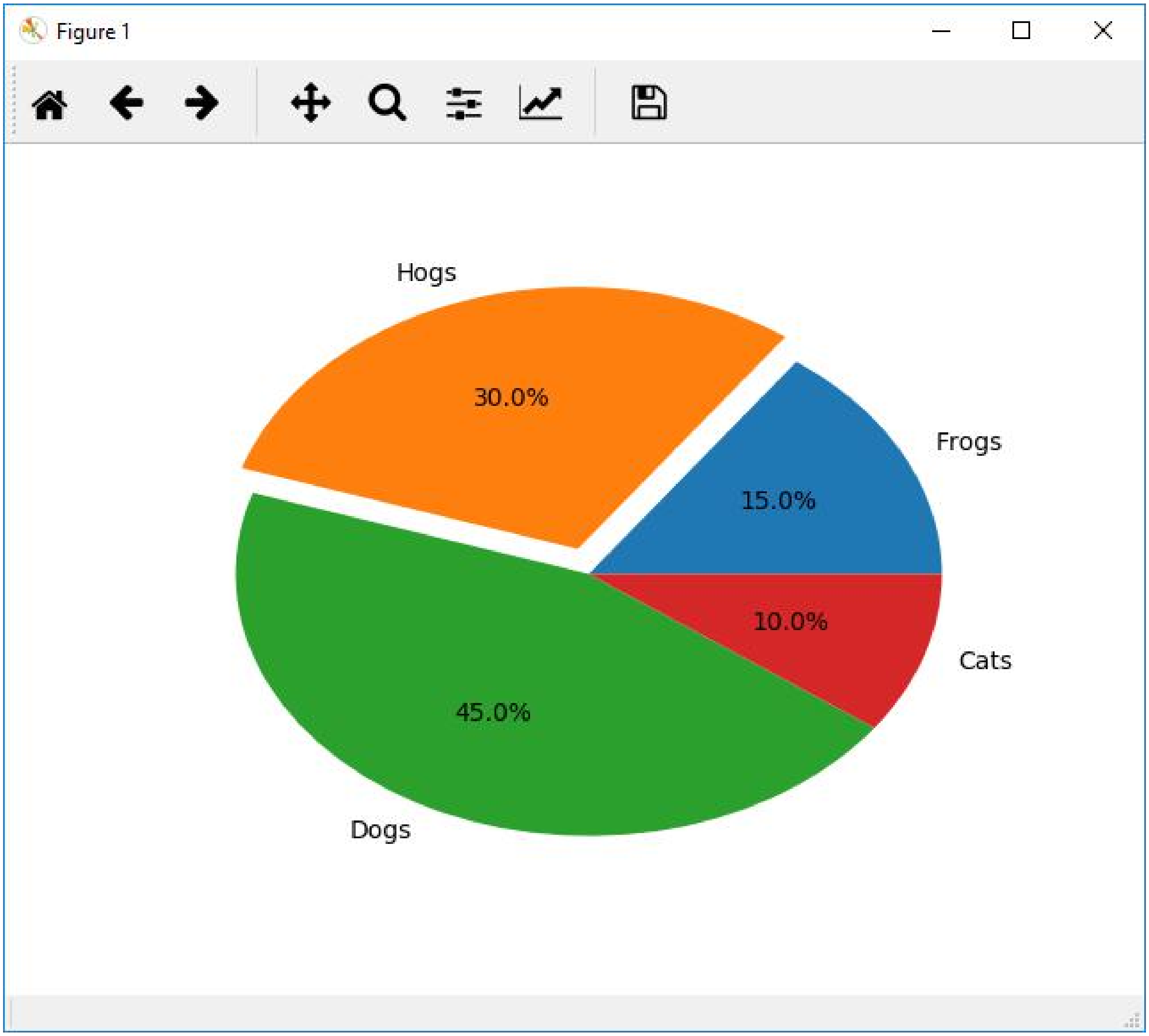
Plotting: Subplots;

```
import matplotlib.pyplot as plt
import numpy as np
plt.clf()
x = np.arange(1,11)
y = x**2

plt.subplot(2,1,1)
plt.plot(x,y,'r:*')
plt.xlabel('x')
plt.ylabel('y')
plt.title('y is equal to x squared')
plt.legend(['y = x**2'])

plt.subplot(2,1,2)
plt.plot(x,y,'k-.>')
plt.xlabel('x')
plt.ylabel('y')
plt.title('y is equal to x squared')
plt.legend(['y = x**2'])

plt.grid('on')
plt.grid('off')
plt.show()
```



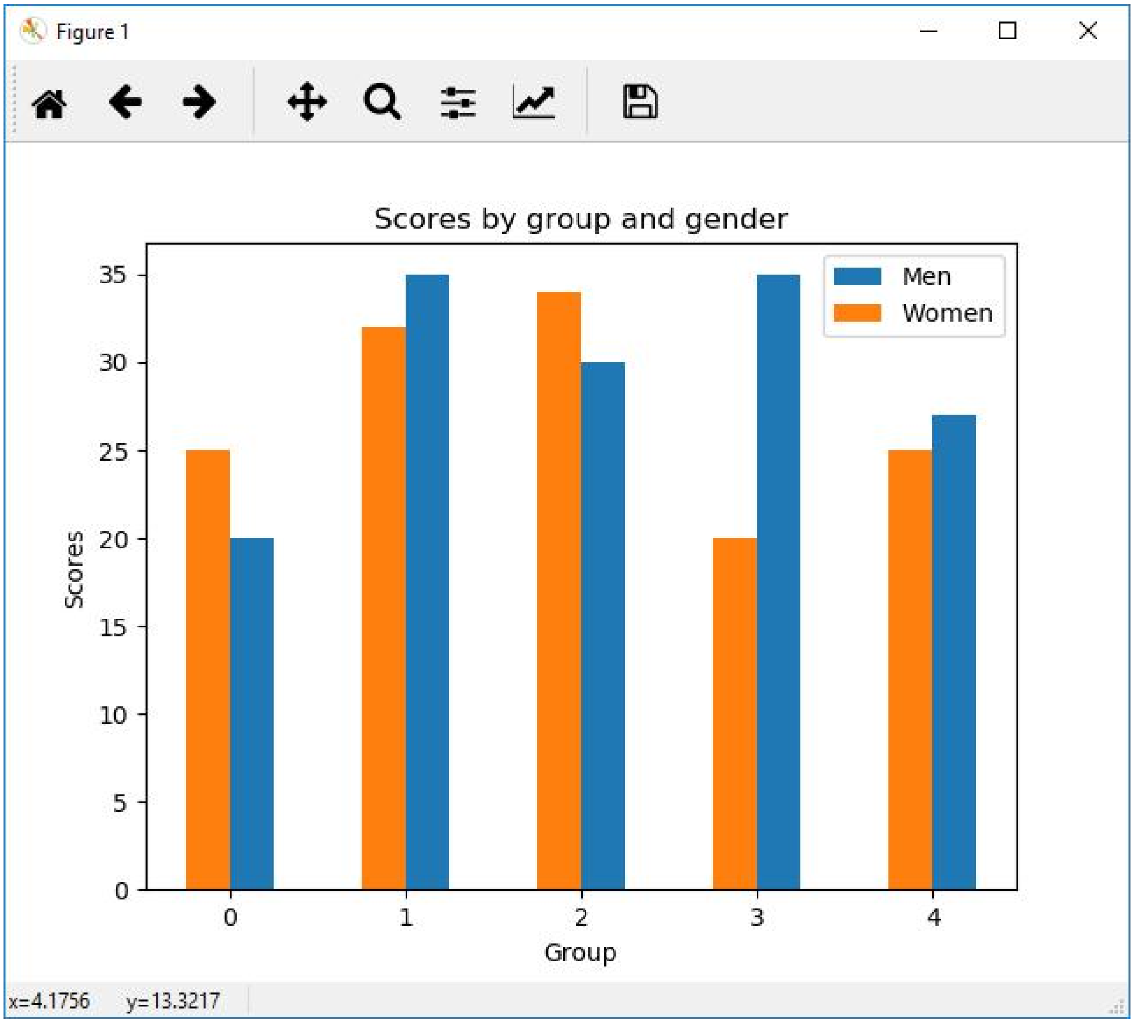
Pie Charts;

```
import matplotlib.pyplot as plt

plt.clf()

# Pie chart,slices will be ordered,
# plotted counter-clockwise:

labels = 'Frogs', 'Hogs', 'Dogs', 'Cats'
sizes = [15, 30, 45, 10]

# only "explode" the 2nd slice(i.e. 'Hogs')
explode = (0, 0.1, 0, 0)

plt.pie(sizes, explode=explode, labels=labels,autopct='%1.1f%%')
plt.show()
```



Bar Charts;

```
import matplotlib.pyplot as plt
import numpy as np
plt.clf()
n_groups = 5
index = np.arange(n_groups)
means_men = (20, 35, 30, 35, 27)
means_women = (25, 32, 34, 20, 25)
bar_width = 0.25
plt.bar(index, means_men, bar_width, align='edge')
plt.bar(index, means_women, -bar_width, align='edge')
plt.xlabel('Group')
plt.ylabel('Scores')
plt.title('Scores by group and gender')
plt.legend(['Men', 'Women'])
plt.show()
```



Histogram;

- Often you want to get an understanding of the distribution of certain numerical variables within it when exploring a dataset.
- One way of visualizing the distribution of a single numerical variable is by using a histogram.
- A histogram divides the values within a numerical variable into “bins” and counts the number of observations that fall into each bin.
- By visualizing these binned counts in a columnar fashion, we can obtain an understanding of the distribution of values within a variable.
- A histogram can only be used to plot numerical values and it is usually used for large data sets. It is useful for detecting outliers and/or gaps in the data set.

Histogram – Creating Manually;

Example: We want to construct a histogram of the following scores in a math exam where the maximum possible mark is 20.

Scores:

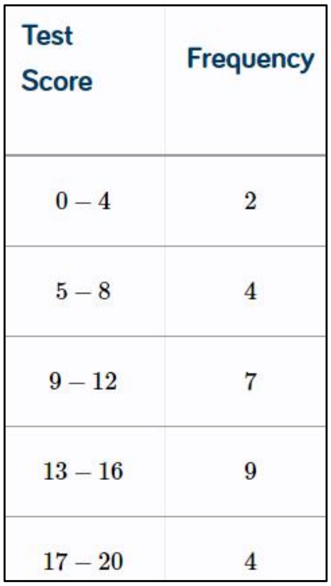
[2, 4, 14, 14, 16, 17, 13, 16, 7,2, 4, 14, 14, 16, 17, 13, 16, 7,8, 9, 10, 11, 19, 18, 15, 15, 16,8, 9, 10, 11, 19, 18, 15, 15, 16,13, 12, 7, 8, 9, 12, 11, 18]

1) First order the data to make it easier to group.
   [2, 2, 4, 4, 7, 7, 7, 8, 8, 8, 9, 9, 9, 10, 10, 11, 11, 11, 12, 12, 13, 13, 13, 14, 14, 14, 14, 15, 15, 15, 15, 16, 16, 16, 16, 16, 16, 17, 17, 18, 18, 18, 19, 19]

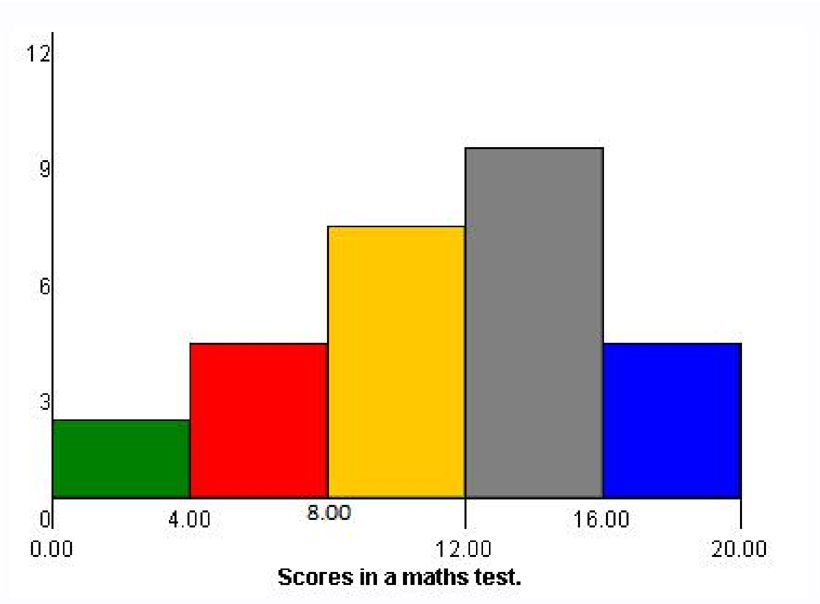
2) Next, look at the data and find the minimum and maximum values to choose the width of each ‘bucket’. The minimum is 2 and the maximum is 19. Let’s choose a width of 4. So there will be 5 buckets: 0-4, 5-8, 9-12, 13-16, 17-20.

3) Next, calculate the number of each values in each bucket.

4) Finally, create a histogram that displays the frequency data.



Test Score	Frequency
0 – 4	2
5 – 8	4
9 – 12	7
13 – 16	9
17 – 20	4



Histogram – Example with Scores;

```
import pandas as pd
import matplotlib.pyplot as plt


df_score = pd.read_csv('scores.csv')

plt.figure(1) #opens a new active figure window
plt.clf() #clears the figure window

plt.subplot(1,2,1)
plt.hist(df_score['grades'],5)
#plots scores with 5 bins/buckets
plt.xlabel('Bins set to 5')

plt.subplot(1,2,2)
plt.hist(df_score['grades'], color = 'red')
plt.xlabel('Bins not specified (default:10)')
plt.show()
```

<u>Scores.csv content;</u>

```
grades
    2
    4
   14
   14
   16
   17
   13
   16
    7
    2
    4
   14
    .
    .
```

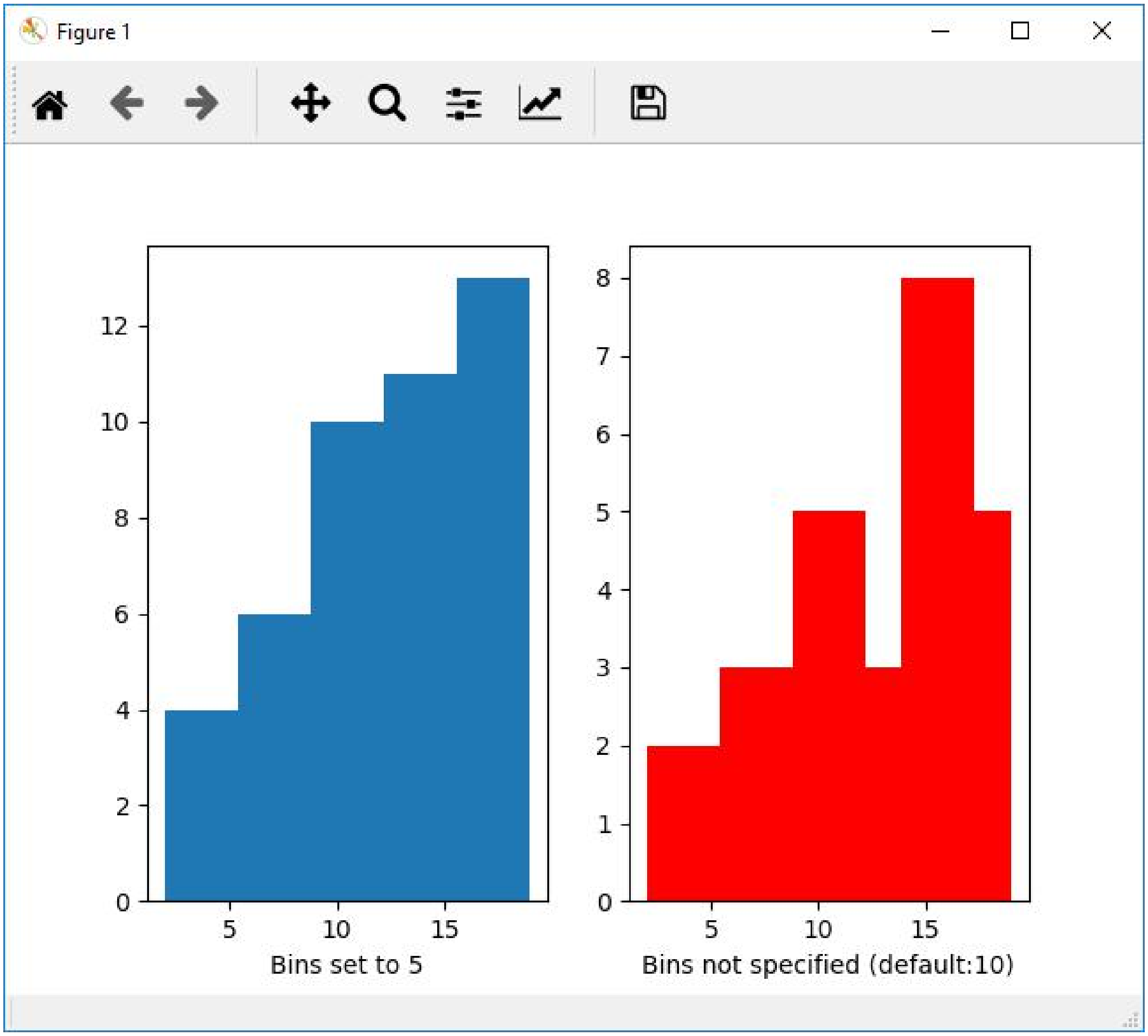


Figure 1
0
2
4
6
8
10
12
5
10
15
Bins set to 5
0
1
2
3
4
5
6
7
8
5
10
15
Bins not specified (default:10)