LEARN PYTHON VISUALIZATION IN 6 HOURS

PLOT AND UNDERSTAND MORE THAN 20 GRAPHS

WWW.THEDATAMONK.COM

Making graph is different and creating informative visualization is completely different. You need to put in a lot of effort to understand the type of graph which will provided maximum information in a limited space. There are times when you should use something as basic as histogram and then there will be times where you need to put in a lot of information in just one chart.

In this book we will start with basics of visualization in Python, will then move to some cool graphs, and after that we will tackle some interview questions which are asked in Data Science interviews related to Visualizations.

If you like the book, please review it on Amazon and get any one book from below for free by sending a mail to contact@thedatamonk.com

1. The Monk who knew Linear Regression (Python): Understand, Learn and Crack Data Science Interview
2. 100 Python Questions to crack Data Science/Analyst Interview
3. Complete Linear Regression and ARIMA Forecasting project using R
4. 100 Hadoop Questions to crack data science interview: Hadoop Cheat Sheet
5. 100 Questions to Crack Data Science Interview
6. 100 Puzzles and Case Studies To Crack Data Science Interview
7. 100 Questions To Crack Big Data Interview
8. 100 Questions to Learn R in 6 Hours
9. Complete Analytical Project before Data Science interview
10. 112 Questions To Crack Business Analyst Interview Using SQL
11. 100 Questions To Crack Business Analyst Interview
12. A to Z of Machine Learning in 6 hours
13. In 2 Hours Create your first Azure ML in 23 Steps
14. How to Start A Career in Business Analysis
15. Web Analytics - The Way we do it
16. Write better SQL queries + SQL interview Questions
17. How To Start a Career in Data Science
18. Top Interview Questions And All About Adobe Analytics
19. Business Analyst and MBA Aspirant's Complete Guide to Case Study -

Case Study Cheat sheet

Do check out our website www.thedatamonk.com for 100 Days Data Science Challenge, Interview Questions, and much more.

Index

1.Introduction

Data visualization is the discipline of trying to understand data by placing it in a visual context so that patterns, trends and correlations that might not otherwise be detected can be exposed. It is one of the basic but a very important weapon in your Data Science career.
Python is blessed with some good libraries for visualizations.

Keep practicing while reading the book.

Open Jupyter notebook or any other IDE of your preference.

Library to use – There are a lots of good visualization libraries, but matplot library is the most preferred one to start with because of its simple implementation.
So,We will mostly concentrate on matplot library.

Importing the library and giving it the standard alias as plt.

```
import matplotlib.pyplot as plt
```


Following are the two important functions which will come handy in this book:-

To display a chart you should use – **plt.show()**
To save the chart as an image, use the code – **plt.savefig("Filename.png")**

Popular plotting libraries in Python are:-

1. Matplotlib – Best to start with. It provides easy implementation and gives a lot of freedom
2. Seaborn – It has a high level interface and great default styles

3. Plotly – To create interactive plots
4. Pandas Visualization – Easy interface, built on Matplotlib

2.Line Chart

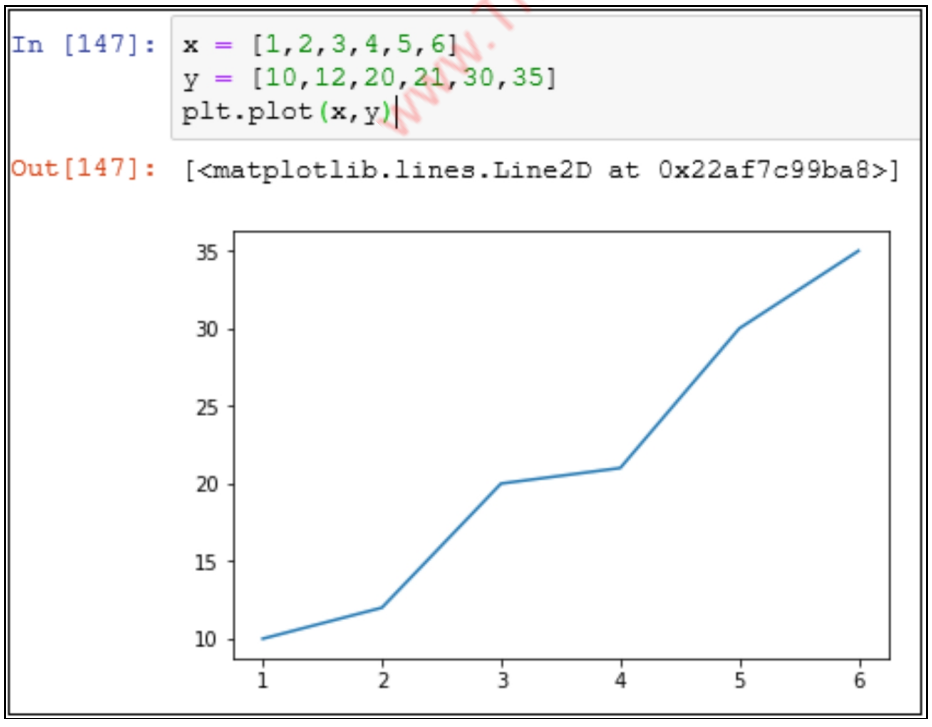
A line chart or line graph is a type of chart which displays information as a series of data points called ‘markers’ connected by straight line segments.

So, a line plot is a very basic plot which is used to show observations collected after a regular interval. The x-axis represents the interval and the y-axis represents the values.

Lets plot our first graph

```
import matplotlib.pyplot as plt
x = [1,2,3,4,5,6]
y = [10,12,20,21,30,35]
plt.plot(x,y)
```

Here is what you will get



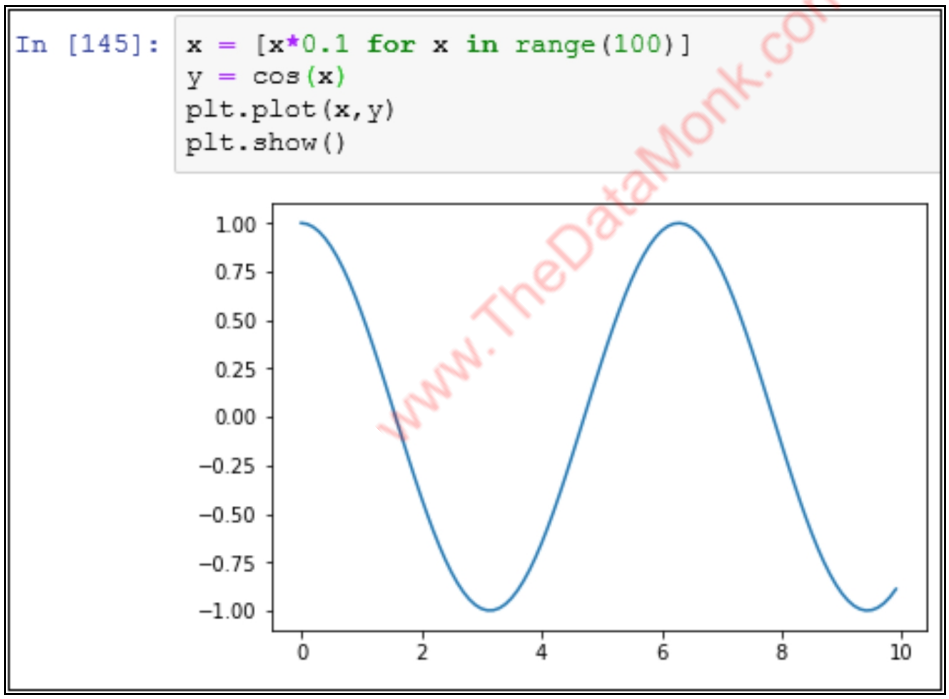
Graph 1 – Basic Line Chart

Plot a sin graph using line plot

import matplotlib.pyplot as plt
from numpy import cos

*x = [x*0.01 for x in range(100)]*
y = cos(x)
plt.plot(x,y)
plt.show()

Here is what you get as a cos graph



Graph 2 – Cos graph using line plot

You know how to plot a line graph, but there is one important thing missing in the graph i.e. the x and y-axis, and the plot title. Let’s create another line plot for number of students in a class for the following data

c = [1,2,3,4,5,6]
student = [40,52,50,61,70,78]

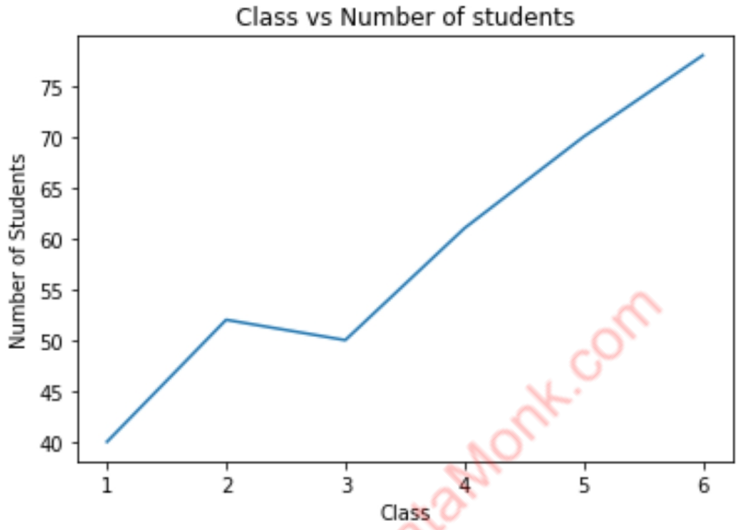
Following commands are used to put x-axis label, y-axis label, and chart title

plt.xlabel(“Label”)
plt.ylabel(“Label”)
plt.title(“Title”)

The code is given below

```
c = [1,2,3,4,5,6]
student = [40,52,50,61,70,78]
plt.xlabel("Class")
plt.ylabel("Number of Students")
plt.title("Class vs Number of students")
plt.plot(c, student)
```

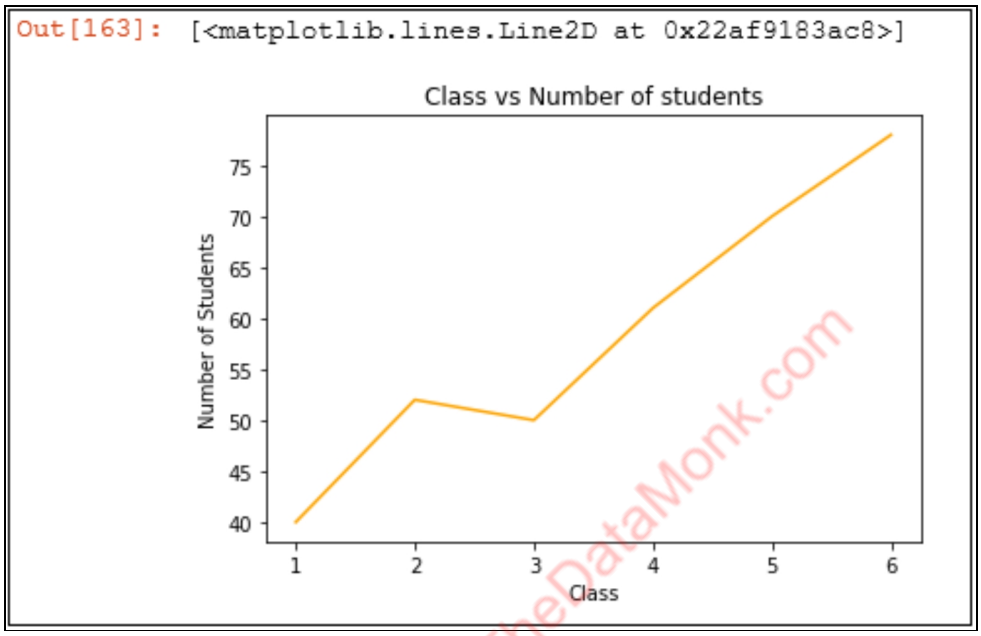
```
In [162]: c = [1,2,3,4,5,6]
          student = [40,52,50,61,70,78]
          avg_marks = [34,43,54,44,50,55]
          plt.xlabel("Class")
          plt.ylabel("Number of Students")
          plt.title("Class vs Number of students")
          plt.plot(c,student)

Out[162]: [<matplotlib.lines.Line2D at 0x22af80d9be0>]
```



Graph 3 - Class vs Number of Students chart with proper labels and plot title

You want to change the color of the line?
Try the following code instead to make the line green in color

plt.plot(c,student,color='g')

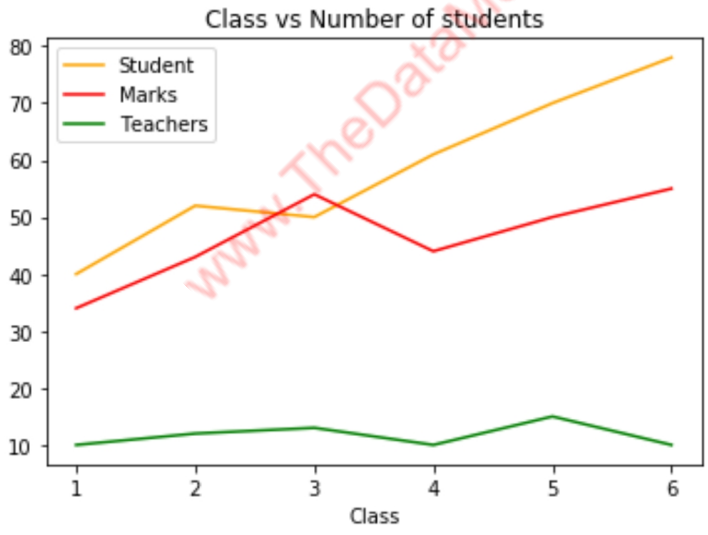


Graph 4 – Adding color to the same graph

Multi Line Chart

You can also add multiple plots in the same graph. Let's try to put a couple of new lines in the graph i.e. number of teachers and average marks

```
In [169]: c = [1,2,3,4,5,6]
          student = [40,52,50,61,70,78]
          avg_marks = [34,43,54,44,50,55]
          num_of_teachers = [10,12,13,10,15,10]
          plt.xlabel("Class")
          #plt.ylabel("Number of Students")
          plt.title("Class vs Number of students")
          plt.plot(c,student,color='orange',label='Student')
          plt.plot(c,avg_marks,color='red',label='Marks')
          plt.plot(c,num_of_teachers,color='green',label='Teachers')
          plt.legend(loc="upper left")

Out[169]: <matplotlib.legend.Legend at 0x22af939fb00>
```



Graph 5 – Adding multiple lines to a graph

To add a legend, you have to give label to each of the line which you want to plot and after that you specify a location to the legend

The code is self explanatory and is given below:-

```
c = [1,2,3,4,5,6]
student = [40,52,50,61,70,78]
avg_marks = [34,43,54,44,50,55]
num_of_teachers = [10,12,13,10,15,10]
plt.xlabel("Class")
#plt.ylabel("Number of Students")
plt.title("Class vs Number of students")
plt.plot(c,student,color='orange',label='Student')
plt.plot(c,avg_marks,color='red',label='Marks')
plt.plot(c,num_of_teachers,color='green',label='Teachers')
plt.legend(loc="upper left")
```

3.Bar Chart

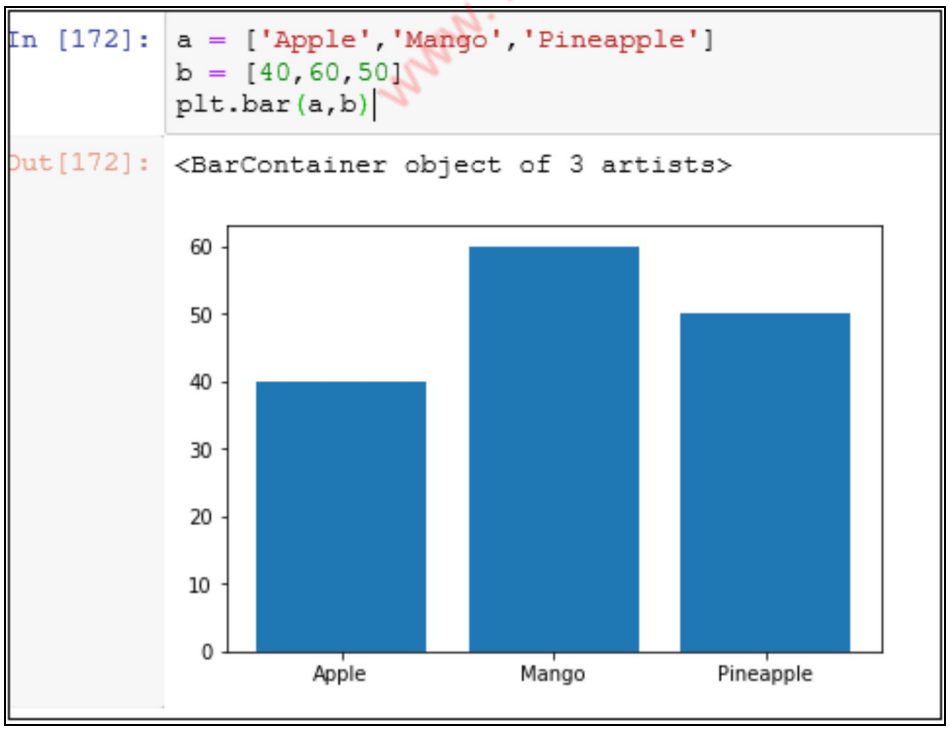
"A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent. The bars can be plotted vertically or horizontally."

After the line chart, the second basic but highly used chart is the bar chart

To create a bar chart – plt.bar(x,y)

We will plot few graphs first and then you can put labels, title, and legends later.

import matplotlib.pyplot as plt
a = ['Apple','Mango','Pineapple']
b = [40,60,50]
plt.bar(a,b)

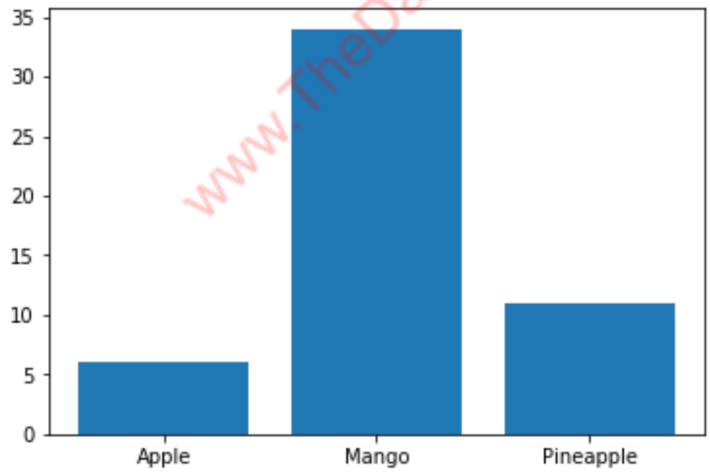


Graph 6 – A simple bar chart

Use random values between 1 and 100 to create the same graph.

import matplotlib.pyplot as plt
from random import seed
from random import randint
seed(123)
x = ['Apple','Mango','Pineapple']
y = [randint(0,100),randint(0,100),randint(0,100)]
plt.bar(x,y)

```
In [171]: from random import seed
          from random import randint
          seed(123)
          x = ['Apple','Mango','Pineapple']
          y = [randint(0,100),randint(0,100),randint(0,100)]
          plt.bar(x,y)
Out[171]: <BarContainer object of 3 artists>
```

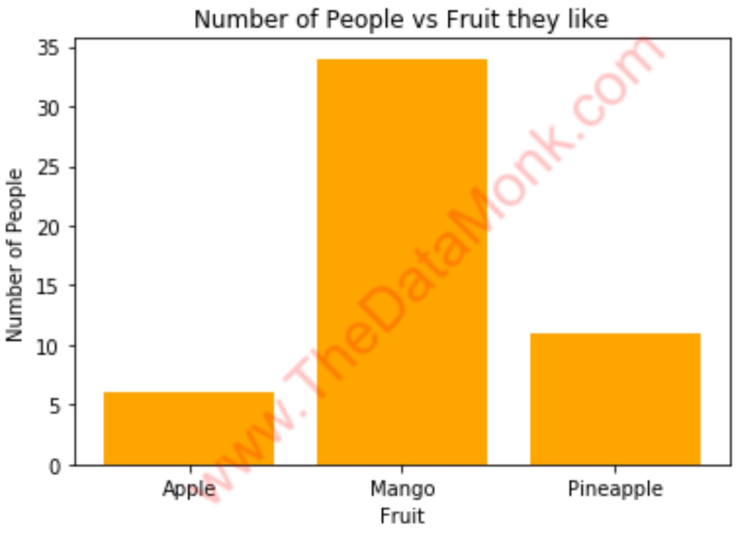


Graph 7 – Bar chart with random values

Adding color, labels, and title to the random values bar chart

In [176]:
```
from random import seed
from random import randint
seed(123)
x = ['Apple','Mango','Pineapple']
y = [randint(0,100),randint(0,100),randint(0,100)]
plt.xlabel("Fruit")
plt.ylabel("Number of People")
plt.title("Number of People vs Fruit they like")
plt.bar(x,y,color='orange')
```

Out[176]: <BarContainer object of 3 artists>

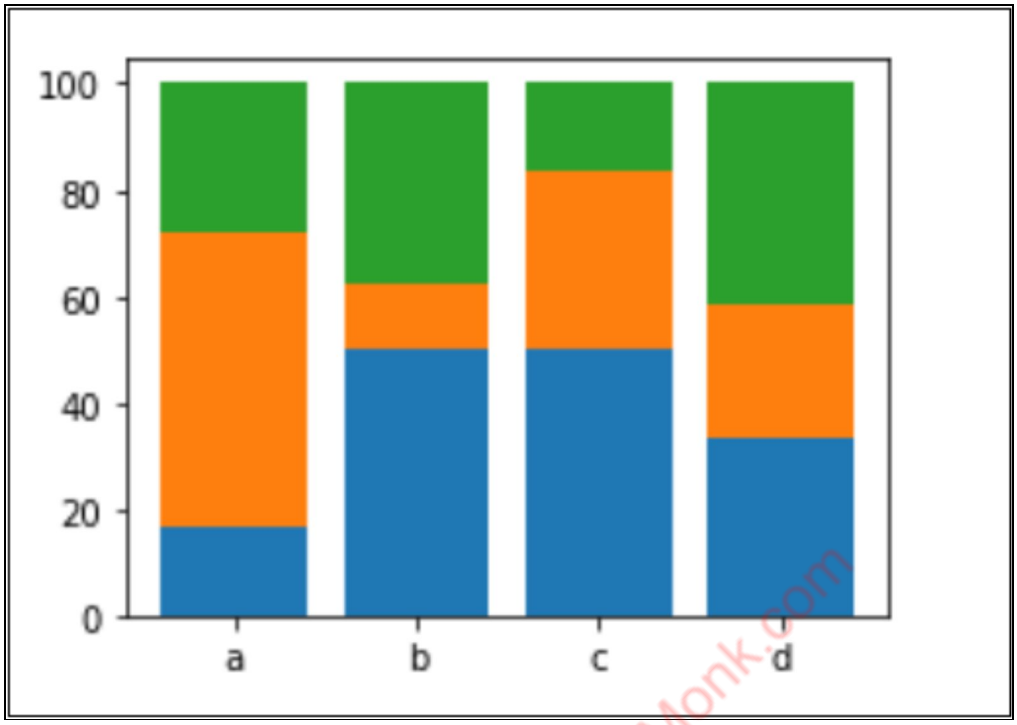


Stacked 100% bar chart with sub component

When you have to show components of components like the graph below

Example of 100% bar chart

```
x = ["a","b","c","d"]
y1 = np.array([3,8,6,4])
y2 = np.array([10,2,4,3])
y3 = np.array([5,6,2,5])

snum = y1+y2+y3

# normalization
y1 = y1/snum*100.
y2 = y2/snum*100.
y3 = y3/snum*100.
plt.figure(figsize=(4,3))

# stack bars

plt.bar(x, y1, label='y1')
plt.bar(x, y2 ,bottom=y1,label='y2')
plt.bar(x, y3 ,bottom=y1+y2,label='y3')
```



Graph 8 – A 100% stacked bar chart

4.Histogram

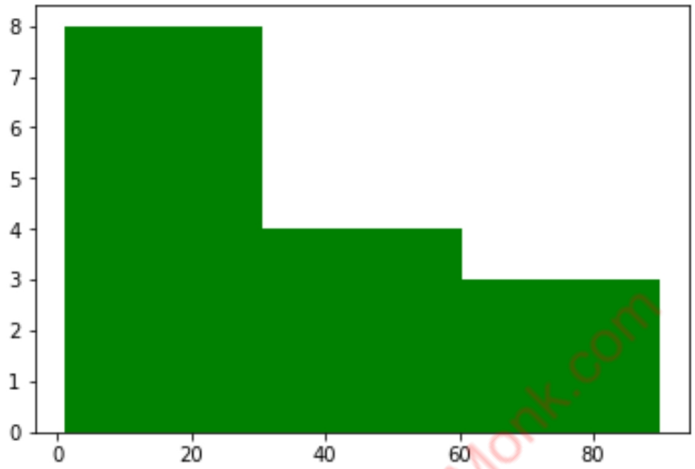
Histograms are density estimates. A density estimate gives a good impression of the distribution of the data. The idea is to locally represent the data density by counting the number of observations in a sequence of consecutive intervals (bins).

To plot a histogram use this code – plt.hist(x,y)

A simple histogram plot

```
q = [1,2,34,5,44,66,66,90,33,45,2,1,2,3,4]
plt.hist(q,bins = 3,color='green')
```

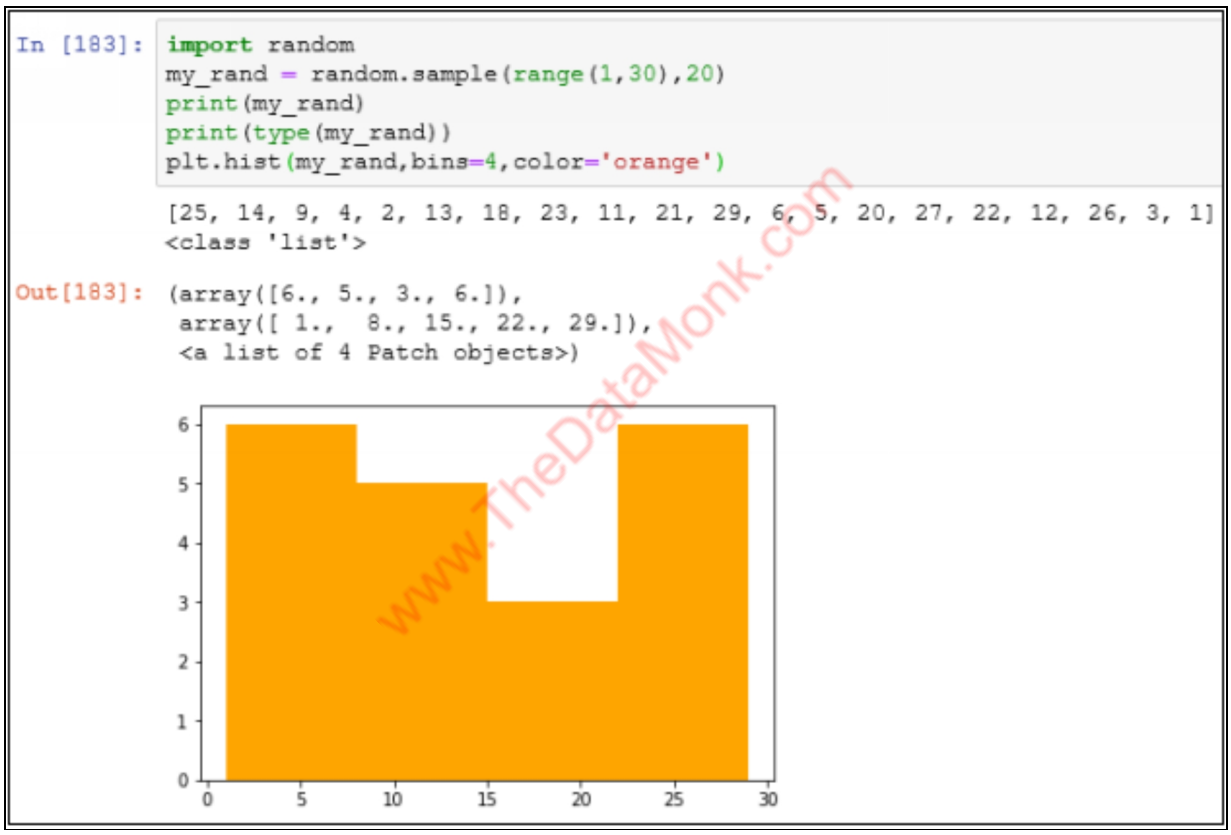
```
In [182]: q = [1,2,34,5,44,66,66,90,33,45,2,1,2,3,4]
          plt.hist(q,bins = 3,color='green')

Out[182]: (array([8., 4., 3.]),
           array([ 1.        , 30.66666667, 60.33333333, 90.
           <a list of 3 Patch objects>)
```



Graph 9 – A simple histogram

Create a list using random variables and plot it in 4 bins

```
import random
my_rand = random.sample(range(1,30),20)
print(my_rand)
print(type(my_rand))
plt.hist(my_rand,bins=4,color='orange')
```

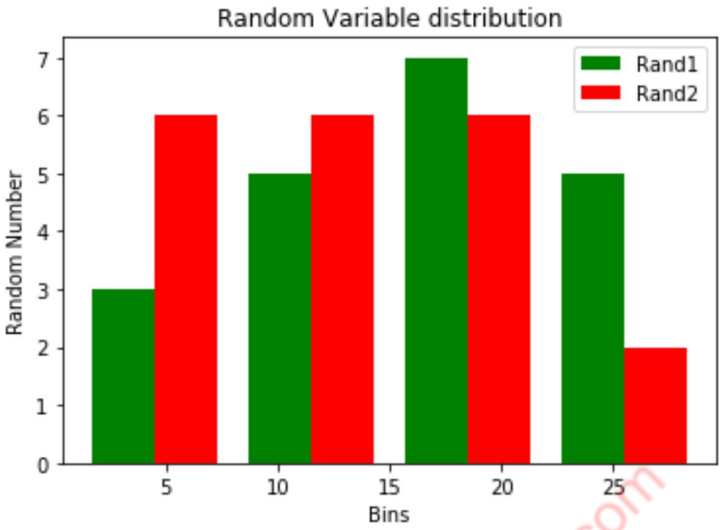


Graph 10 – A histogram made with random variables

In Histogram also you can add more than one data points to make parallel bars.

```
import random
my_rand = random.sample(range(1,30),20)
my_rand2 = random.sample(range(1,25),20)
print(my_rand)
print(type(my_rand))
plt.hist([my_rand,my_rand2],bins=4,color=['green','red'])
legend = ['Rand1','Rand2']
plt.legend(legend)
plt.xlabel("Bins")
plt.ylabel("Random Number")
plt.title("Random Variable distribution")
```

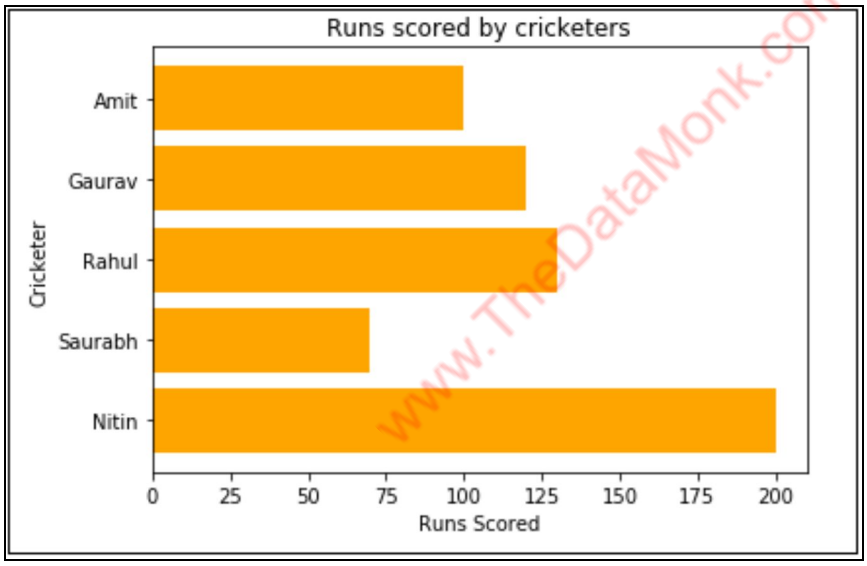
```
Out[191]: Text(0.5, 1.0, 'Random Variable distribution')
```



Graph 11 – Parallel histogram

Horizontal Histogram

```
import numpy as np
import matplotlib.pyplot as plt
name = ['Nitin','Saurabh','Rahul','Gaurav','Amit']
run = [200,70,130,120,100]
plt.barh(name,run,color='orange')
plt.xlabel("Runs Scored")
plt.ylabel("Cricketer")
plt.title("Runs scored by cricketers")
plt.show()
```

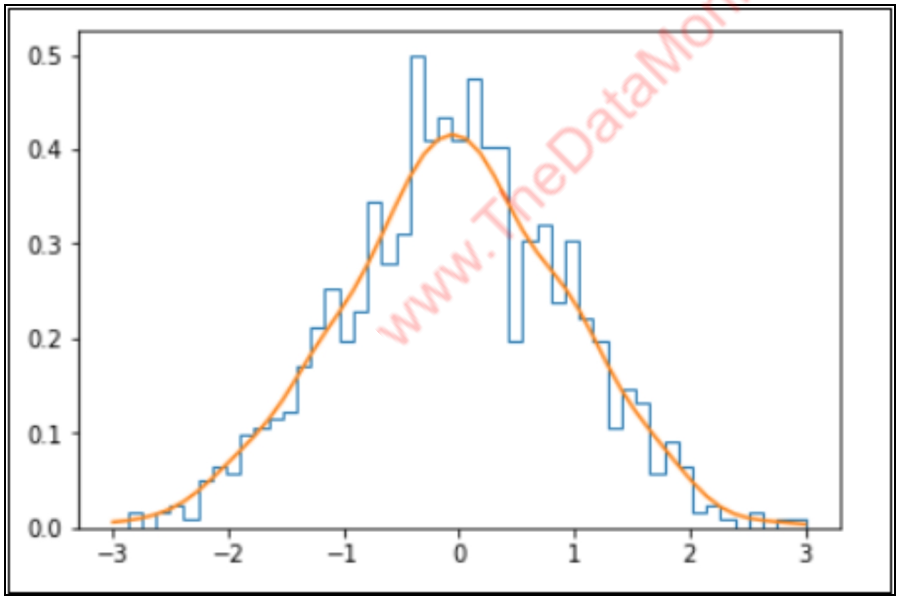


Graph 12 – A horizontal histogram

Line Histogram

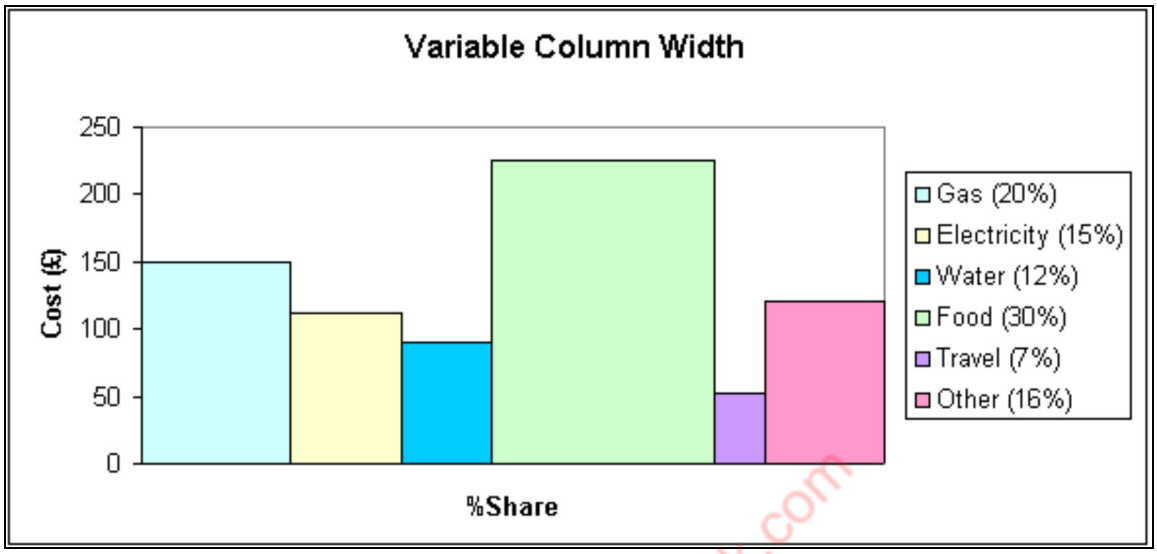
Now let’s create a line histogram with some random data

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats
noise = np.random.normal(0, 1, (1000, ))
density = stats.gaussian_kde(noise)
n, x, _ = plt.hist(noise, bins=np.linspace(-3, 3, 50),histtype=u'step',
density=True)
plt.plot(x, density(x))
plt.show()
```



Graph 13 – A line histogram

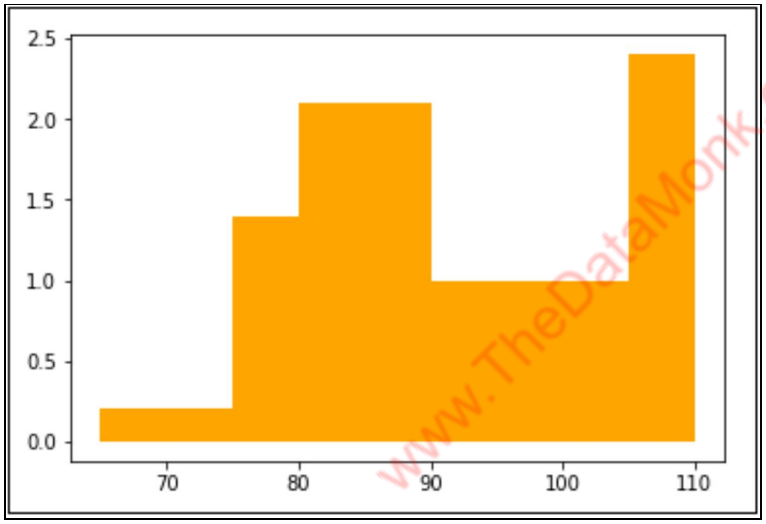
Variable Width histogram



This is how a variable column width histogram looks like

Let’s create one with our dataset

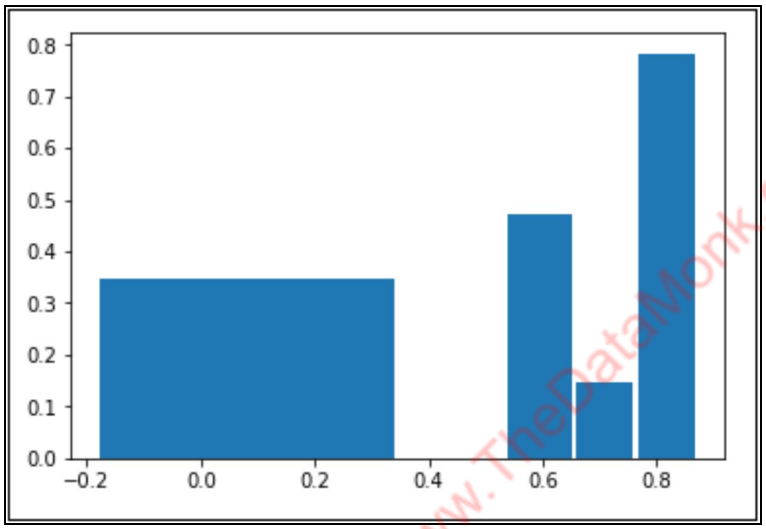
```
import numpy as np
import matplotlib.pyplot as plt
freqs = np.array([2, 7, 21, 15, 12])
bins = np.array([65, 75, 80, 90, 105, 110])
widths = bins[1:] - bins[:-1]
heights = freqs.astype(np.float)/widths
plt.fill_between(bins.repeat(2)[1:-1], heights.repeat(2), facecolor='orange')
plt.show()
```



Graph 14 – A variable width histogram

One more example below

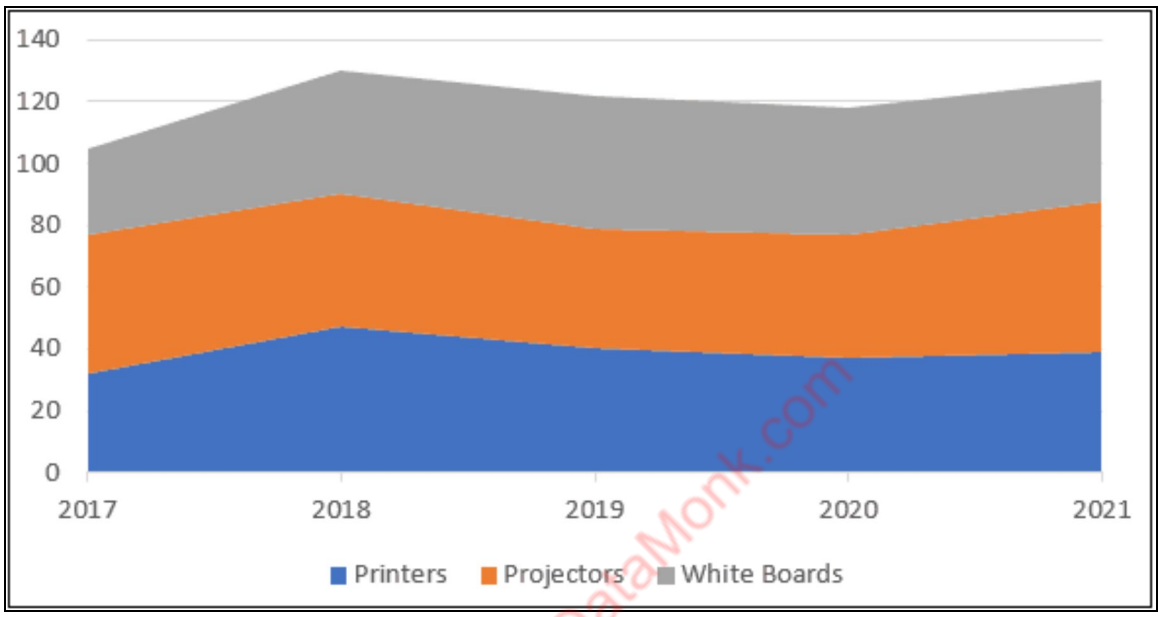
```
import numpy as np
import matplotlib.pyplot as plt
x = np.sort(np.random.rand(6))
y = np.random.rand(5)
plt.bar(x[:-1], y, width=x[1:] - x[:-1])
plt.show()
```



Graph 15 – Variable width histogram

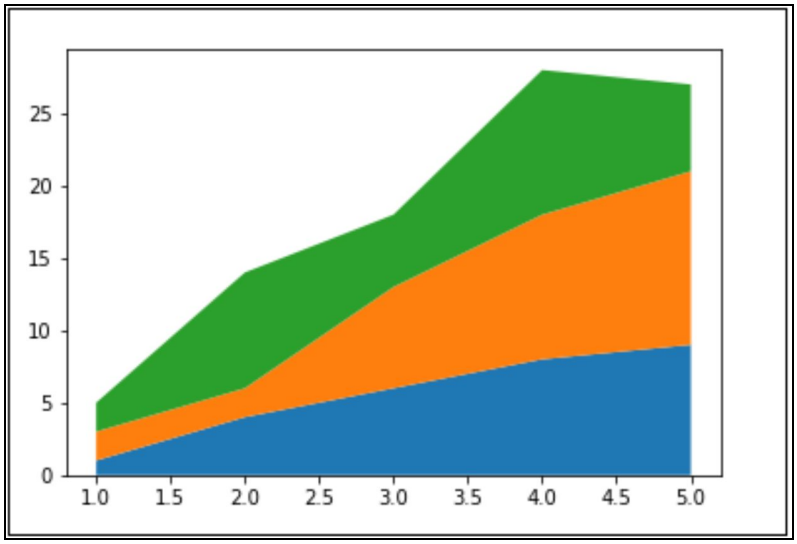
5.Area Chart

Below is how an area chart looks like:



Let’s create a basic area chart with some dummy data

```
Import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# Data
x=range(1,6)
y=[ [1,4,6,8,9], [2,2,7,10,12], [2,8,5,10,6] ]

# Plot
plt.stackplot(x,y, labels=['A','B','C'])
plt.legend(loc='upper left')
plt.show()
```



Graph 16 – A basic area chart

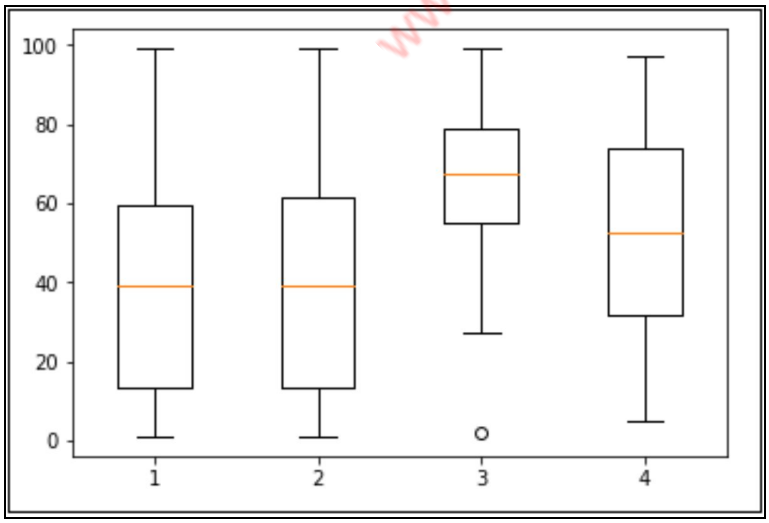
You already know how to add x-labels, y-labels, title, etc.
Go ahead and add these in the graph above

6.Box and Whisker Plot

A box and whisker plot, or boxplot for short, is generally used to summarize the distribution of a data sample.
The x-axis is used to represent the data sample, where multiple boxplots can be drawn side by side on the x-axis if desired.

Box plot is one of the most common type of graphics. It gives a nice type of summary of one or more numeric variables. The line that divides the box in the two half is the median of the numbers.
The end of the boxes represents

```
seed(123)
a = random.sample(range(1,100),20)
b = random.sample(range(1,100),20)
c = random.sample(range(1,100),20)
d = random.sample(range(1,100),20)
list_Ex = [a,b,c,d]
plt.boxplot(list_Ex)
```

Graph 17 – A basic Box-Whisker graph

Now we will try to make the graph look better by adding color to the plot. The box-plot shows median, 25th and 75th percentile, and outliers. You should try to give different color to these points to make the plot more appealing.

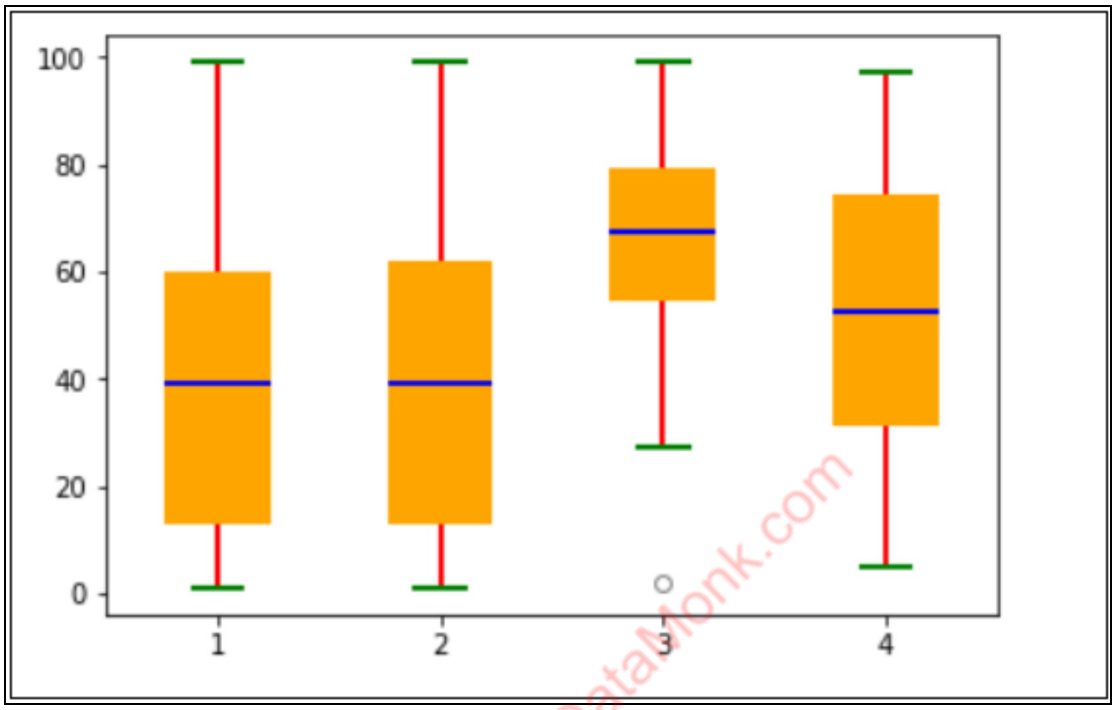
When you plot a boxplot, you can use the following 5 attributes of the plot:-
a. box – To modify the color, line width, etc. of the central box
b. whisker – To modify the color and line width of the line which connects the box to the cap i.e. the horizontal end of the box plot
c. cap – The horizontal end of the box
d. median – The center of the box
e. flier

The box denotes the 1st and 3rd Quartile and it is called IQR i.e. the Inter Quartile Range. The lower fence is at Q1 - 1.5*IQR and the upper fence is at Q3 + 1.5*IQR. Any point which falls above or below it is called fliers or outliers

Following is the code with some fancy colors to help you understand each term individually.

```
bp=plt.boxplot(list_Ex,patch_artist = True)
for box in bp['boxes']:
    box.set(color='orange',linewidth=2
for whisker in bp['whiskers']:
    whisker.set(color = 'red',linewidth=2)
for cap in bp['caps']:
    cap.set(color='green',linewidth=2)
for median in bp['medians']:
    median.set(color='blue',linewidth=2)
for flier in bp['fliers']:
    flier.set(marker='o',color = 'black', alpha=0.5)
```

The graph produced is below:-

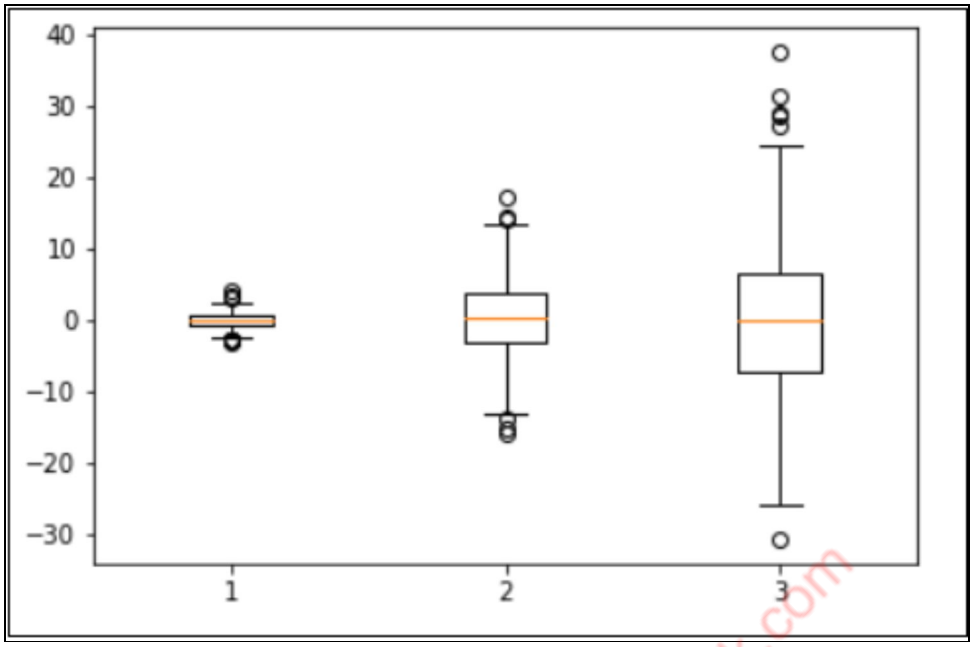


Graph 18 – Box Whisker Chart

Box-plot practice

Following is one more code with the help of which you can replicate a Gaussian distribution

```
from numpy.random import seed
from numpy.random import randn
from matplotlib import pyplot

seed(1)
# random numbers drawn from a Gaussian distribution
x = [randn(1000), 5 * randn(1000), 10 * randn(1000)]
# create box and whisker plot
pyplot.boxplot(x)
# show line plot
pyplot.show()
```



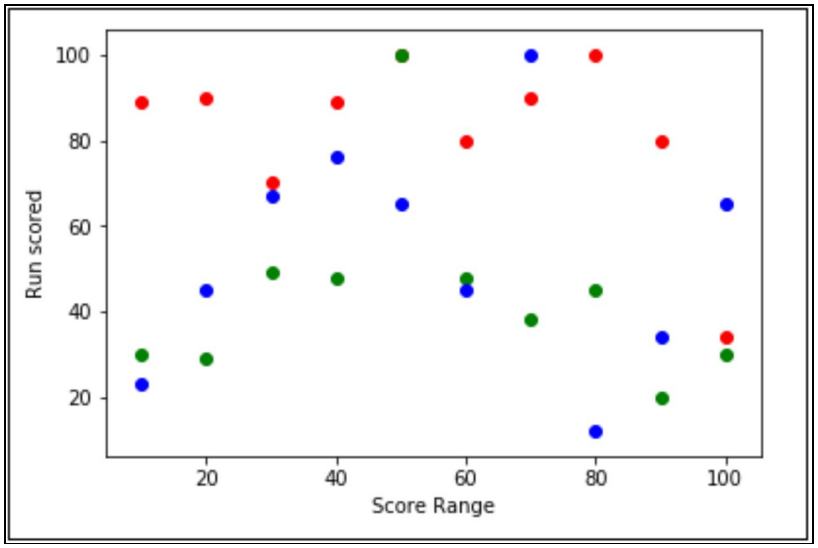
Graph 19 – A Box-Whisker Plot

7.Scatter plot

Scatter plot is an easy to make but interesting visualization which gives a clear picture of how the data is distributed.

Let’s take example of 10 innings played by Sachin, Dhoni, and Kohli and see how their scores are distributed. The code is fairly easy to understand

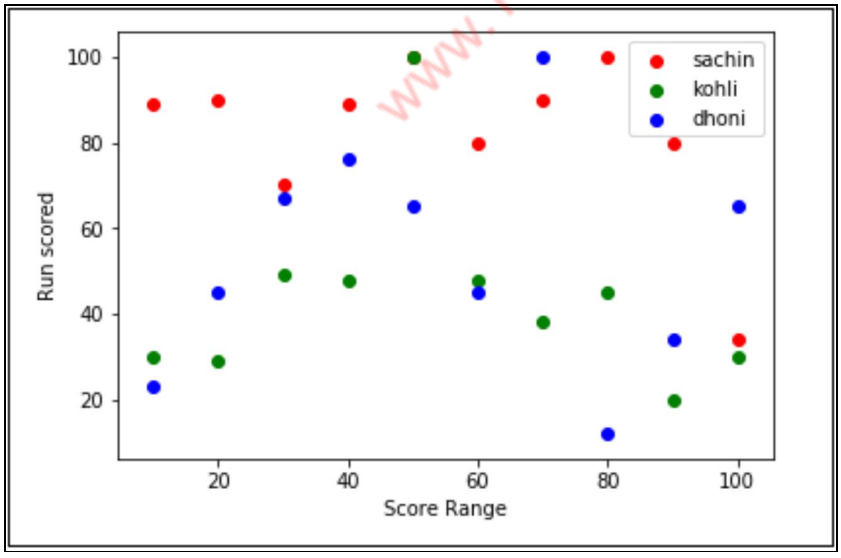
```
sachin = [89, 90, 70, 89, 100, 80, 90, 100, 80, 34]
kohli = [30, 29, 49, 48, 100, 48, 38, 45, 20, 30]
dhoni = [23,45,67,76,65,45,100,12,34,65]
run = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
plt.scatter(run, sachin, color='red')
plt.scatter(run, kohli, color='green')
plt.scatter(run,dhoni,color='blue')
plt.xlabel('Score Range')
plt.ylabel('Run scored')
plt.show()
```



You can also add legend in the plot by using the following command

legend = ['sachin','kohli','dhoni']
plt.legend(legend)

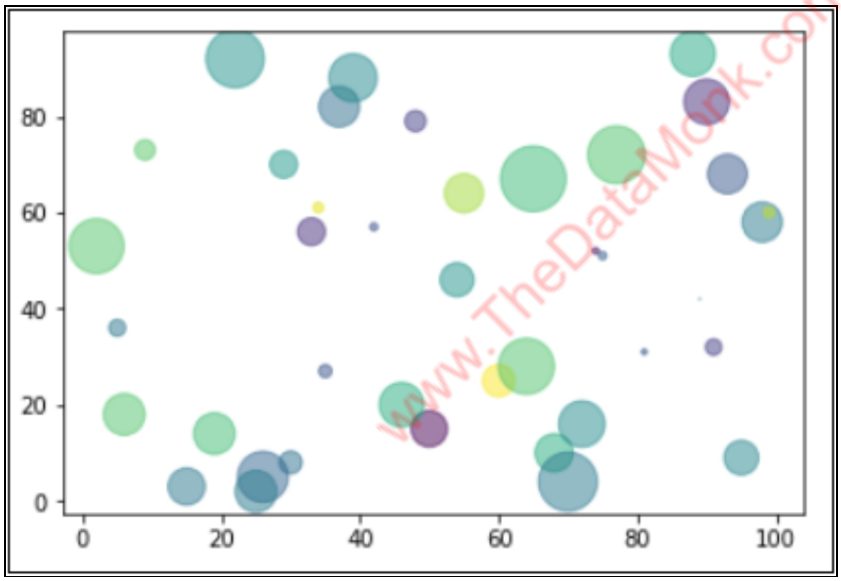
The plot will now look like this



Graph 20 – A scatter plot

Below is one more scatter plot where you give weighted area and the size of the circle will be on the basis of the circle

```
import numpy as np
np.random.seed(123)
x = random.sample(range(1,100),40)
y = random.sample(range(1,100),40)
colors = np.random.rand(N)
area = (30*np.random.rand(N))**2
plt.scatter(x,y,s=area,c=colors,alpha=0.5)
plt.show()
```



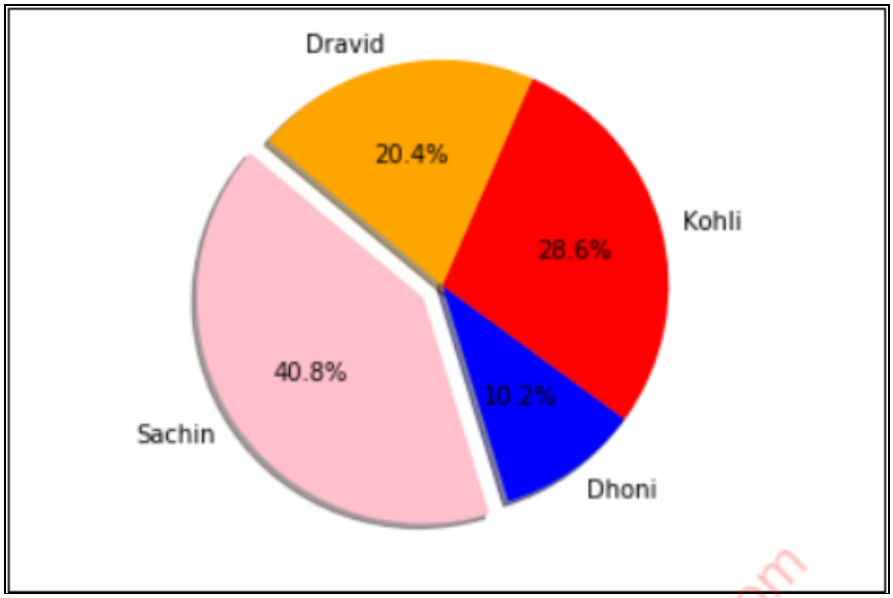
Graph 21 – A scatter plot with area of bubble denoting the volume

8.Pie Chart

Create a pie chart for the number of centuries scored by Sachin, Dhoni, Dravid, and Kohli.

```
labels = 'Sachin','Dhoni','Kohli','Dravid'
size = [100,25,70,50]
colors = ['pink','blue','red','orange']
explode = (0.1,0,0,0)
plt.pie(size,explode=explode,labels=labels,colors=colors,autopct='%1.1f%%',shadow=True,startangle=140)
plt.axis('equal')
plt.show()
```

explode is used to set apart the first part of the pie chart. Everything else in the code is self explanatory. Below is the plot



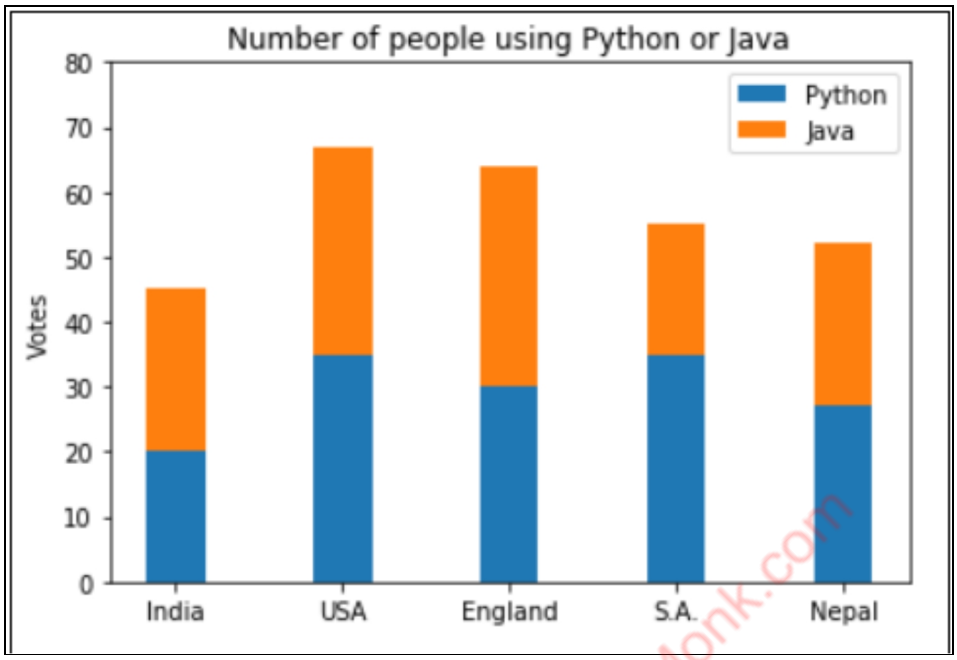
Graph 22 – Pie chart showing performance of cricketers

9.Some cool Visualizations

Create a stacked chart to demonstrate the number of people voting for either Python or Java in 5 countries, namely, India, USA, England, S.A., Nepal

```
import numpy as np
import matplotlib.pyplot as plt
Python = (20, 35, 30, 35, 27)
Java = (25, 32, 34, 20, 25)
width = 0.35       # the width of the bars: can also be len(x) sequence
p1 = plt.bar(ind, Python, width)
p2 = plt.bar(ind, Java, width,bottom=Python)
plt.ylabel('Votes')
plt.title('Number of people using Python or Java')
plt.xticks(ind, ('India', 'USA', 'England', 'S.A.', 'Nepal'))
plt.yticks(np.arange(0, 81, 10))
plt.legend((p1[0], p2[0]), ('Python', 'Java'))
plt.show()
```

xticks is used to give labels to the x-axis and yticks give labels to the y-axis.

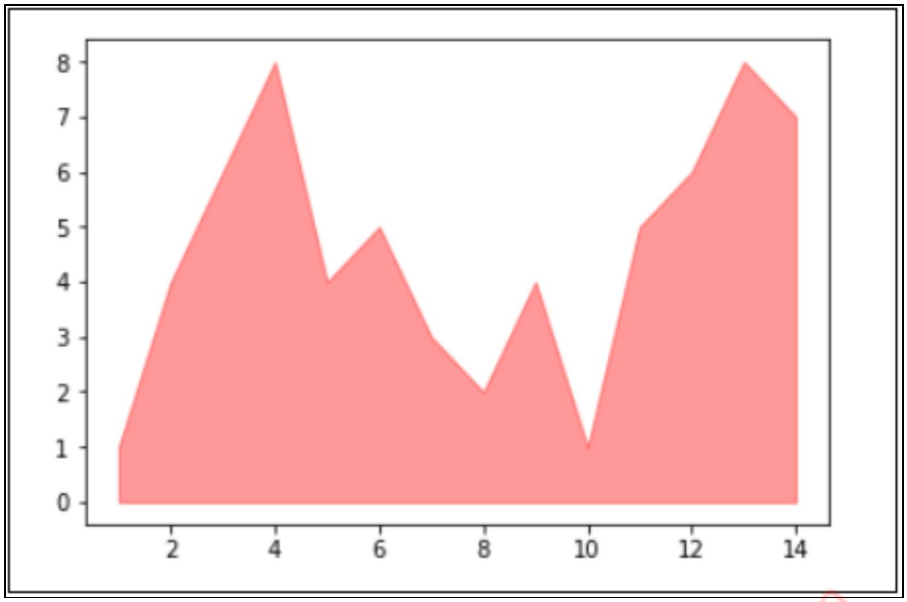


Graph 23 – Stacked Bar graph

A cool area graph

```
import numpy as np
import matplotlib.pyplot as plt
# create data
x=range(1,15)
y=[1,4,6,8,4,5,3,2,4,1,5,6,8,7]

# Change the color and its transparency
plt.fill_between( x, y, color="red", alpha=0.4)
plt.show()
# Same, but add a stronger line on top (edge)
plt.fill_between( x, y, color="red", alpha=0.2)
plt.plot(x, y, color="red", alpha=0.6)
```

The parameter alpha is used to give weight age to the density of color. 0.4 is given to the edge and 0.2 is given to the fill



Graph 24 – An area graph

One of the most important thing is to understand when to use which graph and a list of all the graphs in your knowledge.

There are four types of information which we can display using any plot:-

1. Distribution
2. Comparison
3. Relationship
4. Composition

1. Distribution shows how diversely the data is distributed in your data set. How many people are from which state of the country?

a Histogram – If you have few data point
b. Line Histogram – When you have a lot of data points
c. Scatter plot – When you have to show the distribution of 2-3 variables

2. Comparison – When you have to compare something over 2 or more categories

a. Variable width chart – When you have to compare two variables per item
b. Tables with embedded charts – When there are many categories, basically a matrix of charts
c. Horizontal or Vertical Histogram – When there are few categories in a data set
d. If you want to compare something over time

i. **Line Chart**
ii. **Bar Vertical Chart**
iii. **Many categories line chart**

3. Relationship Charts – When you want to see the relationship between two or more variables then you have to use relationship charts

a. Scatter Plot
b. Scatter plot bubble chart

4. Composition Charts – When you have to show a percentage or composition of variables.

a. Pie Chart – Very basic plot when there are 3-6 categories
b. Stacked 100% bar chart with sub component – When you have to show components of components
c. Stacked 100% bar chart – When you have to look into the contribution of each component.
d. Stacked area chart – When relative and absolute difference matters

Take time to understand the use of different graphs. Plot a lot of graphs.
It will not require more than 6 hours to complete the book. So, do give it a try and if you like the book, please review or write feedback
Drop your mail to contact@thedatamonk.com to get free learning material.

Keep learning