In [1]:
```
from datetime import date
today=date.today()
print("Today's date:",today)
```

```
Today's date: 2022-01-17
```

Plots- 2D and 3D, Graph Customization

Table of Content

There are different visualization libraries in python that provides an interface for drawing various graphics. Some most widely used libraries are Matplotlib and Seaborn.

Import the required libraries

In [3]:
```
# import the library NumPy
import numpy as np

# import the library Pandas
import pandas as pd

# import the library matplotlib
import matplotlib.pyplot as plt

# import thelibrary seaborn
import seaborn as sns

# to suppress warnings
# import the library warnings
import warnings
# filterwarnings() filters a warning
# action: the action to be taken if a warning is shown
warnings.filterwarnings(action = 'ignore')
```

Seaborn library provides a variety of datasets. Plot different visualization plots using various libraries for the 'tips' dataset.

In [4]:
```
# load the 'tips' dataset from seaborn
df_tips = sns.load_dataset('tips')

# display head() of the dataset
df_tips.head()
```

Out[4]:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

1. Plots using Matplotlib

Matplotlib is a Python 2D plotting library. Many libraries are built based on it and use its functions in the backend. pyplot is a subpackage of matplotlib that provides a MATLAB-like way of plotting.

matplotlib.pyplot is a mostly used package because it is elementary to use, and it generates plots in less time.

How to install Matplotlib?

1. You can use-
   `!pip install matplotlib`

CLASS 1

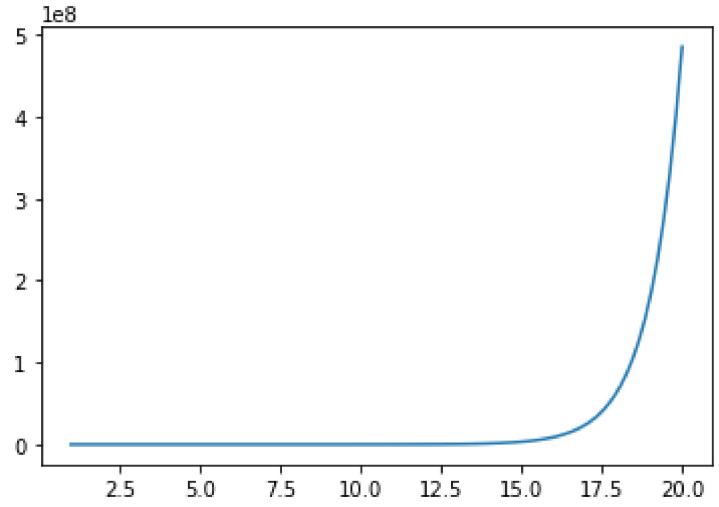
1.1 Line Plot

A line graph is the most uncomplicated plot that displays the relationship between the one independent and one dependent dataset. In this plot, the points are joined by straight line segments.

In [ ]:
```
# line plot
plt.plot(X,Y)
```

1. Write a python program to get exponential graph for x ranges from 0 to 20

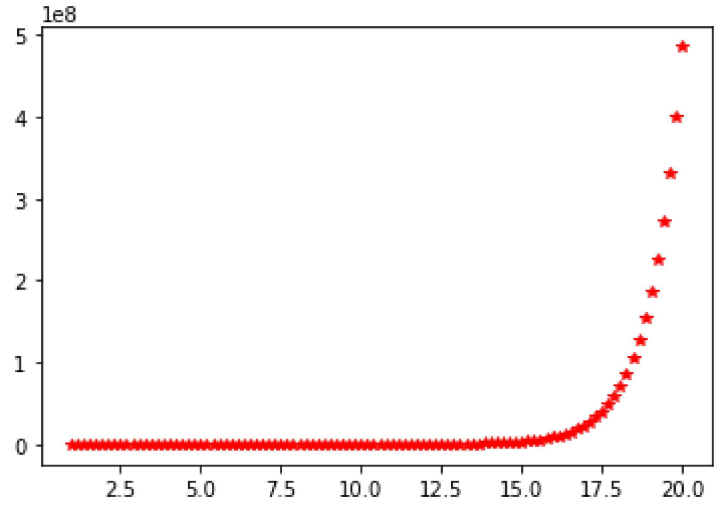
In [5]:
```
# data
# create an array using linspace
X = np.linspace(1,20,100)
# create an array by taking exponential of X
Y = np.exp(X)

# line plot
plt.plot(X,Y)

# display the plot
plt.show()
```



From the plot, it can be observed that as X increases, Y increases exponential.

The above plot can be represented not only by a solid line, but also a dotted line with varied thickness. The points can be marked explicitly using any symbol.

In [8]:
```
# data
# create an array using linspace
X = np.linspace(1,20,100)
# create an array by taking exponential of X
Y = np.exp(X)

# line plot
# the argument 'r*' plots each point as a red '*'
plt.plot(X,Y, 'r*')

# display the plot
plt.show()
```

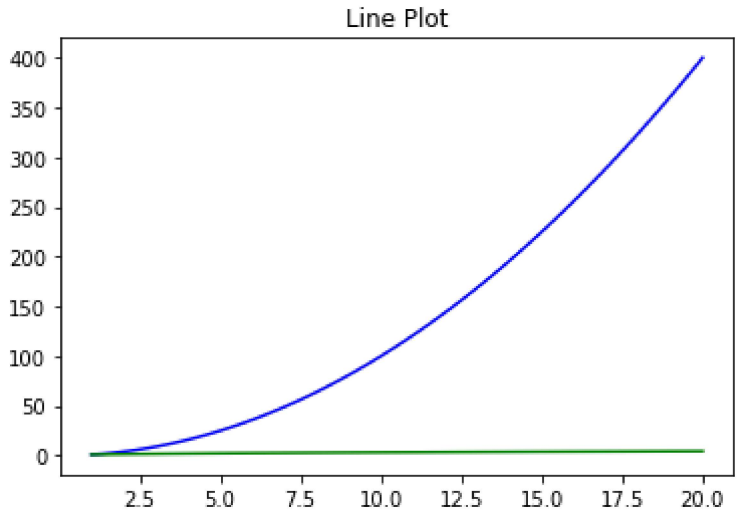


We can change the colours or shapes of the data points.

There can be multiple line plots in one plot. Let us plot three plots together in a single graph. Also, add a plot title.

2.Write a python program to plot the graphs of square and square root of x in range 1 to 20

In [12]:
```
# data
# create an array using linspace
X = np.linspace(1,20,100)
# create an array by taking square of X
Y_square = np.square(X)
# create an array by taking squareroot of X
Y_sqrt = np.sqrt(X)

# line plot
plt.plot( X,Y_square,'b', X,Y_sqrt,'g')

# add title to the plot
plt.title('Line Plot')

# display the plot
plt.show()
```



3.Consider the average heights and weights of persons aged 8 to 16 stored in the following two lists:

height = [121.9,124.5,129.5,134.6,139.7,147.3, 152.4, 157.5,162.6]

weight= [19.7,21.3,23.5,25.9,28.5,32.1,35.7,39.6, 43.2]

Let us plot a line chart where:

i. x axis will represent weight

ii. y axis will represent height

iii. x axis label should be “Weight in kg”

iv. y axis label should be “Height in cm”

v. colour of the line should be green

vi. use * as marker

vii. Marker size as10

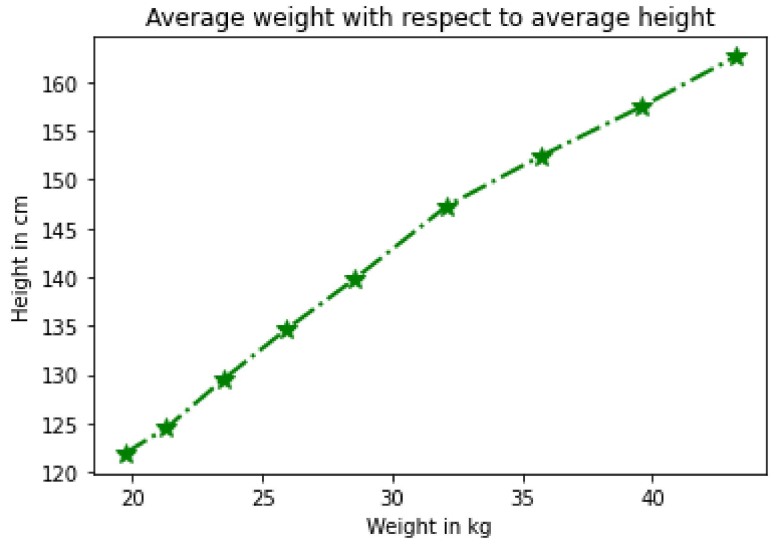
viii. The title of the chart should be “Average

weight with respect to average height”.

ix. Line style should be dashed

x. Linewidth should be 2.

In [13]:
```
import matplotlib.pyplot as plt
import pandas as pd
height=[121.9,124.5,129.5,134.6,139.7,147.3,152.4,157.5,162.6]
weight=[19.7,21.3,23.5,25.9,28.5,32.1,35.7,39.6,43.2]
df=pd.DataFrame({"height":height,"weight":weight})
#Set xlabel for the plot
plt.xlabel('Weight in kg')
#Set ylabel for the plot
plt.ylabel('Height in cm')
#Set chart title:
plt.title("Average weight with respect to average height")
#plot using marker'-*' and line colour as green
plt.plot(df.weight, df.height, marker='*',  markersize=10,color='green', linewidt
plt.show()
```



4.Smile NGO has participated in a three week cultural mela. Using Pandas, they have stored the sales (in Rs) made day wise for every week in a CSV file named “MelaSales.csv”.

Depict the sales for the three weeks using a Line chart. It should have the following:

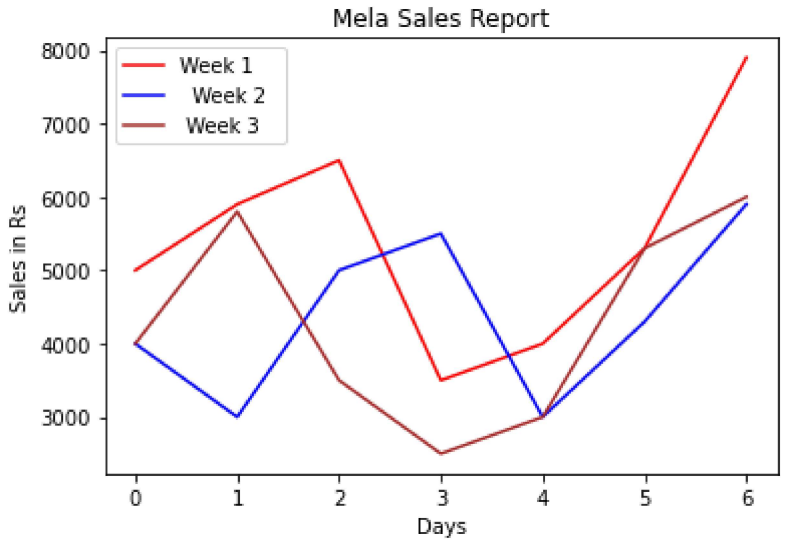
i. Chart title as “Mela Sales Report”.

ii. axis label as Days.

iii. axis label as “Sales in Rs”.

Line colours are red for week 1, blue for week 2 and brown for week 3.

In [14]:
```
import pandas as pd
import matplotlib.pyplot as plt
# reads "MelaSales.csv" to df by giving path to the file
df=pd.read_csv("D:\python\Library\MelaSales.csv")
#create a line plot of different color for each week
df.plot(kind='line', color=['red','blue','brown'])
# Set title to "Mela Sales Report"
plt.title('Mela Sales Report')
# Label x axis as "Days"
plt.xlabel('Days')
# Label y axis as "Sales in Rs"
plt.ylabel('Sales in Rs')
#Display the figure
plt.show()
```



5.Customize the above graph with following conditions.

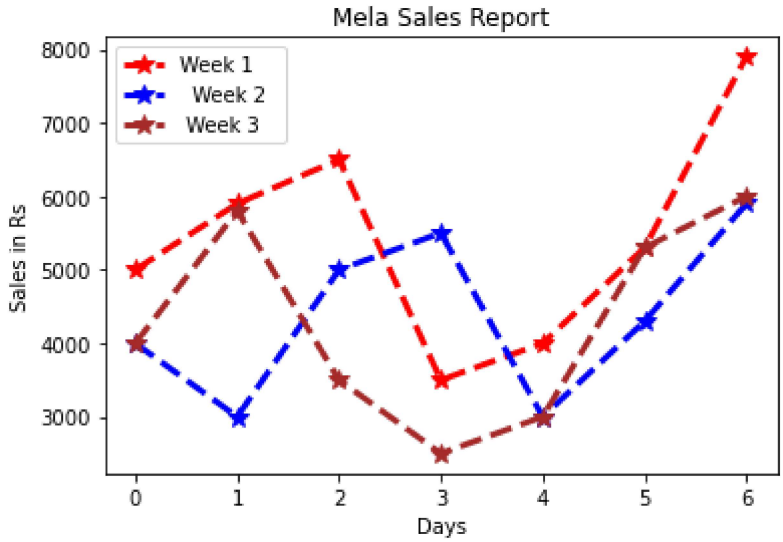
1.Maker : "*"

2.Marker size : 10

3.linestyle :"--"

4.Linewidth :3

In [15]:
```
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv("D:\python\Library\MelaSales.csv")
#creates plot of different color for each week
df.plot(kind='line', color=['red','blue','brown'],marker="*",markersize=10,linewi
plt.title('Mela Sales Report')
plt.xlabel('Days')
plt.ylabel('Sales in Rs')
#store converted index of DataFrame to a list
ticks = df.index.tolist()
plt.show()
```



1.2 Scatter Plot

A scatter plot is a set of points plotted on horizontal and vertical axes. The scatter plot can be used to study the correlation between the two variables. One can also detect the extreme data points using a scatter plot.

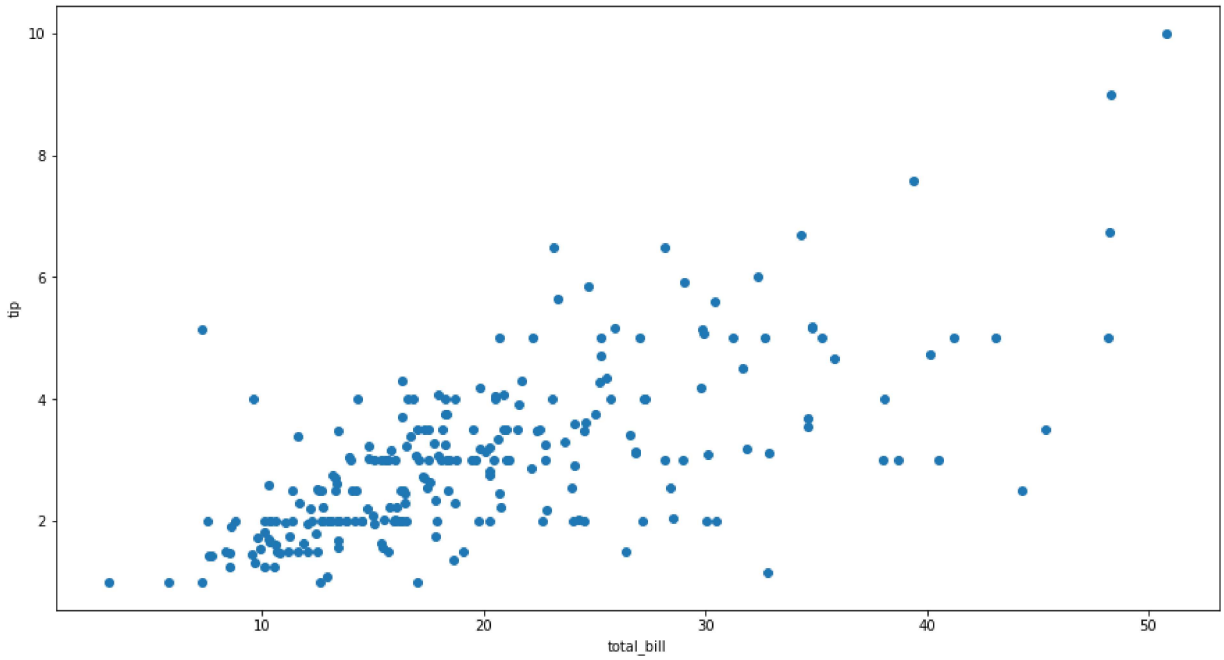
In [ ]:
```
# check the head() of the tips dataset
df_tips.head()
```

#code to plot the scatter plot

plt.scatter(X,Y)

6. Plot the scatter plot for the variables 'total_bill' and 'tip'

In [6]:
```
# data
# let X be the column of total bill
X = df_tips['total_bill']
# let Y be the column of the tip collected
Y = df_tips['tip']

# set the plot size
plt.figure(figsize=(15,8))

# plot the scatter plot
plt.scatter(X,Y)

# add the axes labels to the plot
plt.xlabel('total_bill')
plt.ylabel('tip')

# display the plot
plt.show()
```

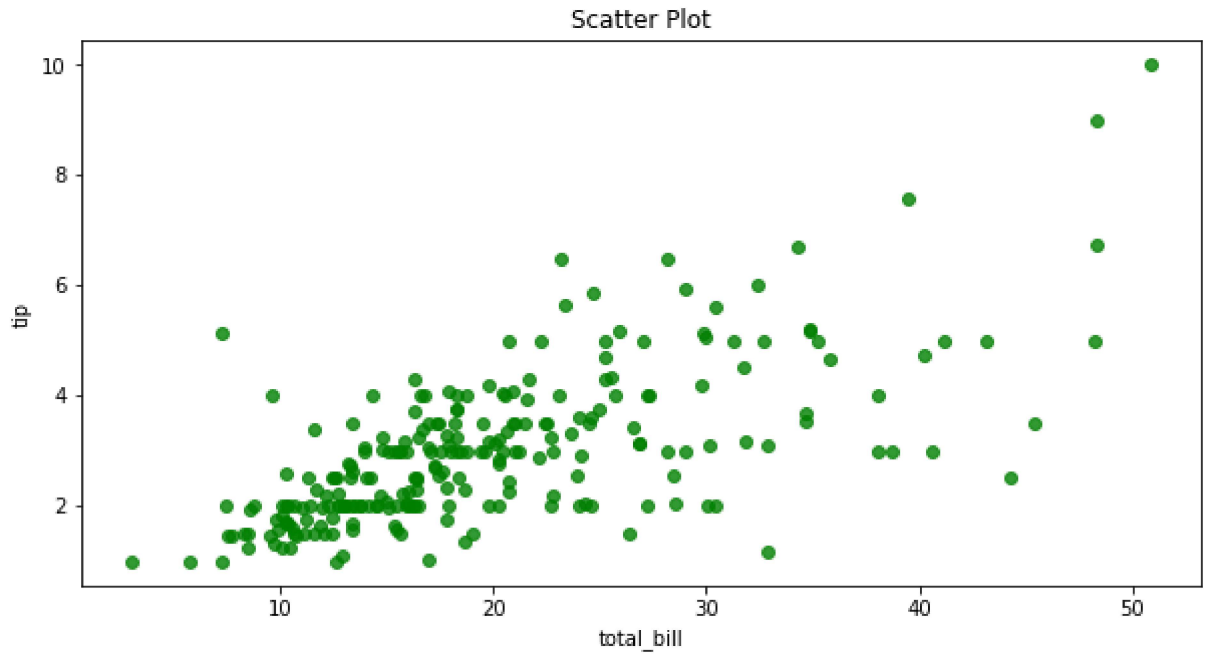


We can add different colours, opacity, and shape of data points. Let us customize the above plot.

7.Customize the plot with following conditions

i. figure size : (10,8) ii. colour : green iii. opacity : 0.1 and 0.8

In [25]:
```
# plot the scatter plot for the variables 'total_bill' and 'tip'

# data
# let X be the column of total bill
X = df_tips['total_bill']
# let Y be the column of the tip collected
Y = df_tips['tip']

# set the plot size
plt.figure(figsize=(10,5))

# plot the scatter plot
# c is for colour, alpha is for opacity (0 < alpha < 1)
####Opacity to make most objects be transparent by specifying a value from 0-1
plt.scatter(X, Y, c= 'green', alpha= 0.8)

# add title
plt.title('Scatter Plot')

# add the axes labels to the plot
plt.xlabel('total_bill')
plt.ylabel('tip')

# display the plot
plt.show()
```



The bubbles with higher radius display that the tip amount is more as compared to the bubbles with less radius.

In [26]:
```
# plot the scatter plot for the variables 'total_bill' and 'tip'

# data
# let X be the column of total bill
X = df_tips['total_bill']
# let Y be the column of the tip collected
Y = df_tips['tip']

# set the plot size
plt.figure(figsize=(10,5))

# plot the scatter plot
# c is for colour, alpha is for opacity (0 < alpha < 1)
####Opacity to make most objects be transparent by specifying a value from 0-1
plt.scatter(X, Y, c= 'green', alpha= 0.1)

# add title
plt.title('Scatter Plot')

# add the axes labels to the plot
plt.xlabel('total_bill')
plt.ylabel('tip')

# display the plot
plt.show()
```