MatplotLib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics.

Some of the major Pros of Matplotlib are:

- Generally easy to get started for simple plots
- Support for custom labels and texts
- High-quality output in many formats
- Very customizable in general

In [1]:

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
```

In [2]:

```
df = pd.DataFrame({'StudyHours':[10,11.5,9,16,9.25,1,11.5,9,8.5,14.5,15.5,13.75,9,8,15.
                   'Grade':[50,50,47,97,49,3,53,42,26,74,82,62,37,15,165]})
df
```

Out[2]:

	StudyHours	Grade
0	10.00	50
1	11.50	50
2	9.00	47
3	16.00	97
4	9.25	49
5	1.00	3
6	11.50	53
7	9.00	42
8	8.50	26
9	14.50	74
10	15.50	82
11	13.75	62
12	9.00	37
13	8.00	15
14	15.50	165

In [3]:

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   StudyHours  15 non-null     float64
 1   Grade       15 non-null     int64
dtypes: float64(1), int64(1)
memory usage: 368.0 bytes
```

Analysis

- Univariate Analysis --> analyzing single variable at a time
- Bivariate Analysis --> analyzing considering 2 variables at a time
- Multivariate Anaysis --> analyzing more than 2 variable at a time

Histogram

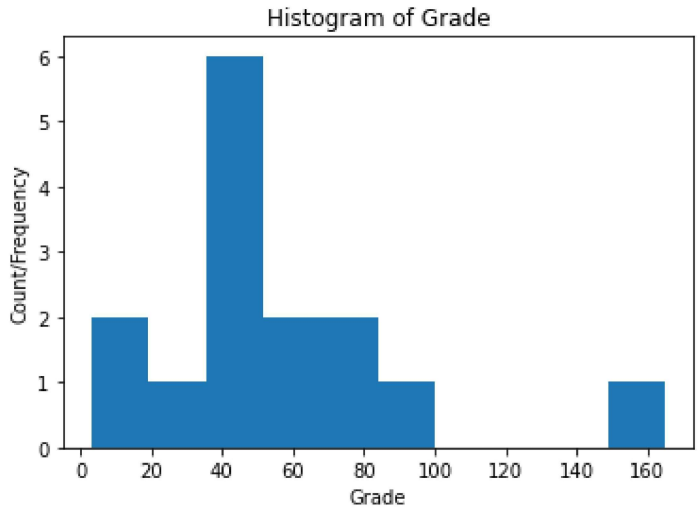
- In histograms, x axis contains a variable and y axis will be a frequency of that variable

Important Paramemeters in Histogram

- data
- bins (default=10)

In [4]:

```
1 plt.hist(df['Grade'])
2 plt.xlabel("Grade")                    # Set the label for the x-axis
3 plt.ylabel("Count/Frequency")          # Set the label for the y-axis
4 plt.title("Histogram of Grade")        # Set title of plot
5 plt.show()
```

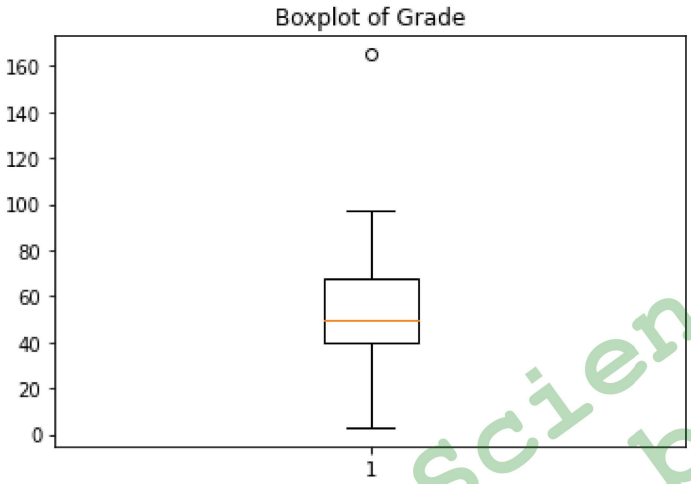


Box Plot

- **Use**: to check, the given data has outliers or not

In [5]:

```
1 plt.boxplot(df['Grade'])
2 plt.title("Boxplot of Grade")
3 plt.show()
```



Scatter Plot

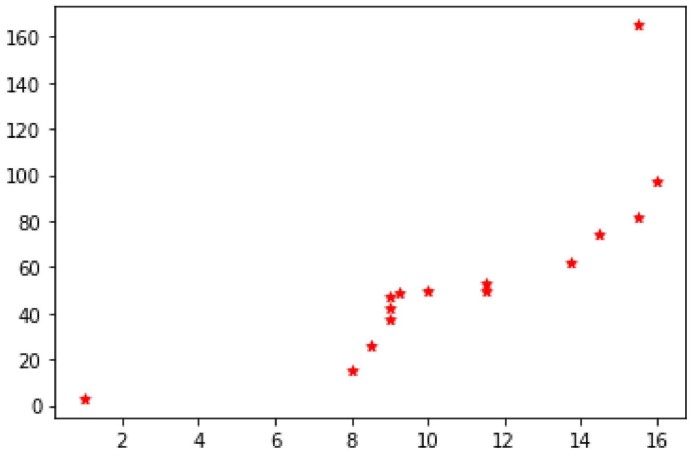
- Marking the data points on the graph
- **Use**: to check 1. linearity, 2. Direction, 3. Strength

Important Paramemeters in Scatterplot

- X-axis values
- y-axis values
- color (default=blue)
- marker (default = "o")
- markersize

In [6]:

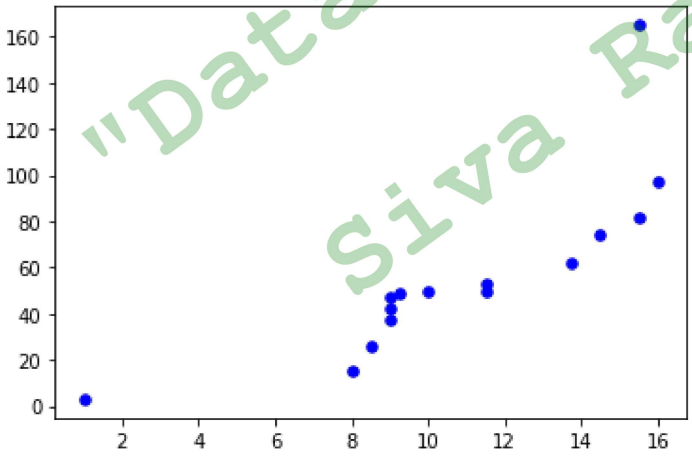
```
plt.scatter(df['StudyHours'],df['Grade'],color="red",marker="*",s=30)
plt.show()
```



In [7]:

```
x=df['StudyHours']
y=df['Grade']

plt.scatter(x,y,color="blue",marker="o",s=30)
plt.show()
```

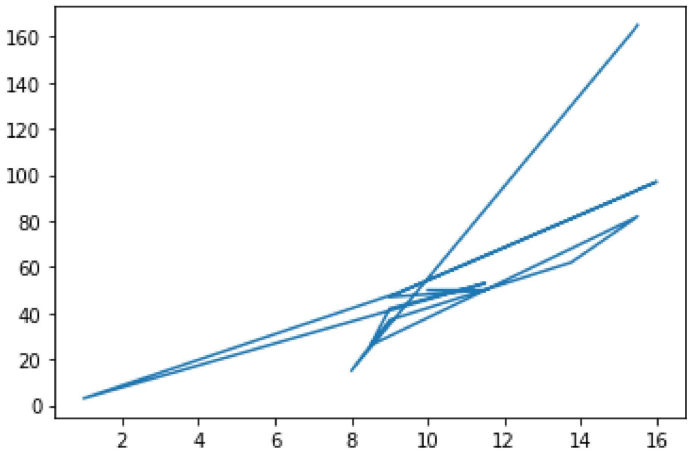


Line Plot

- **Use**: represents the time series data

In [8]:

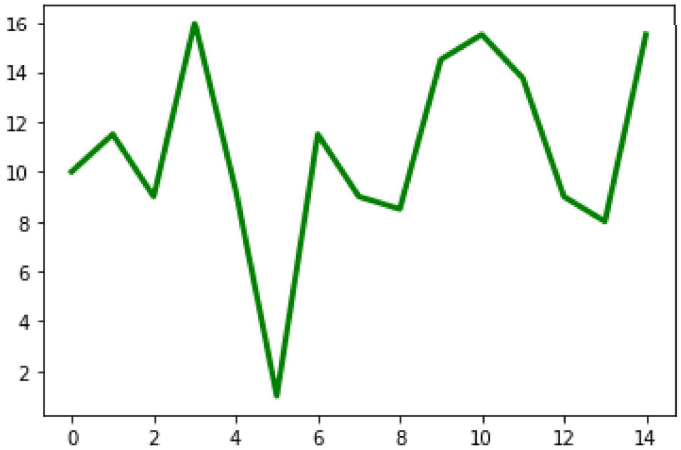
```
plt.plot(df['StudyHours'],df['Grade'])
plt.show()
```



- the x-axis values, should be in the sequential order

In [9]:

```
plt.plot(df["StudyHours"],color='green',linewidth=3.0)
plt.show()
```

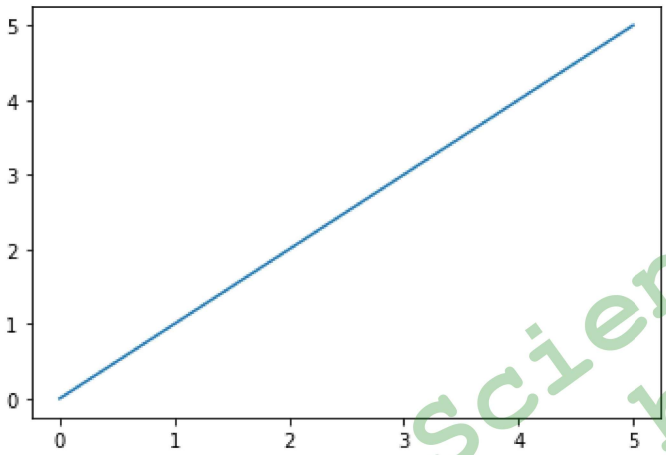


In [10]:

```
x = np.arange(6)
y = x
z = x**2
e = x**3
```

In [11]:

```
plt.plot(x, y)
plt.show()
```

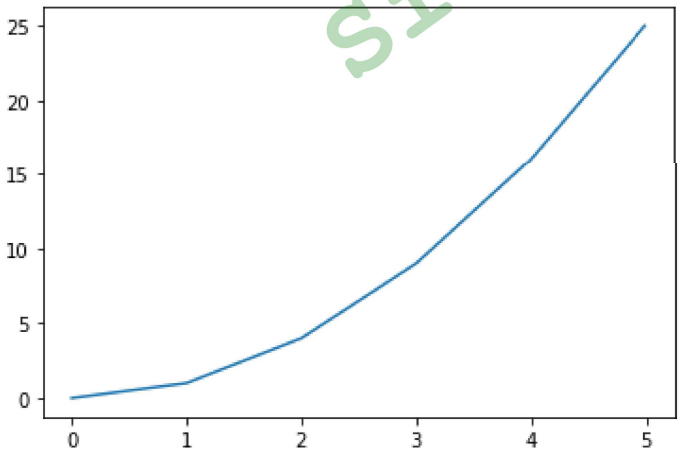


In [12]:

```
plt.plot(x, z)
```

Out[12]:

```
[<matplotlib.lines.Line2D at 0x2d0afd70a00>]
```

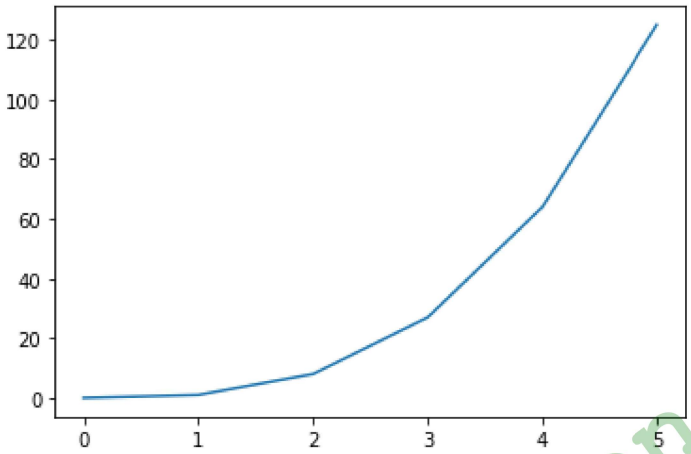


In [13]:

```
1 plt.plot(x, e)
```

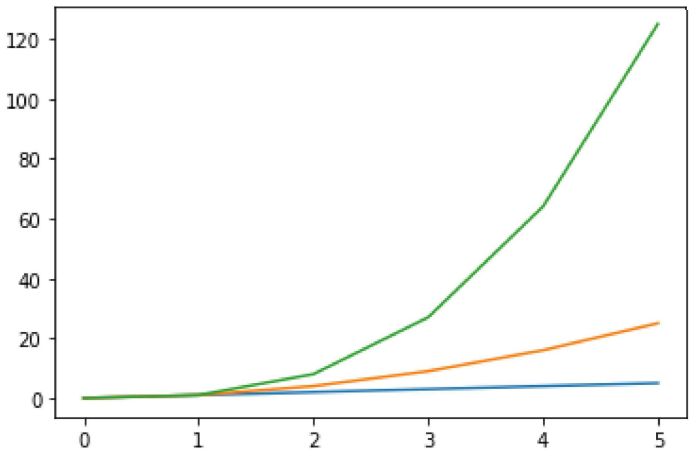
Out[13]:

```
[<matplotlib.lines.Line2D at 0x2d0afdceb80>]
```



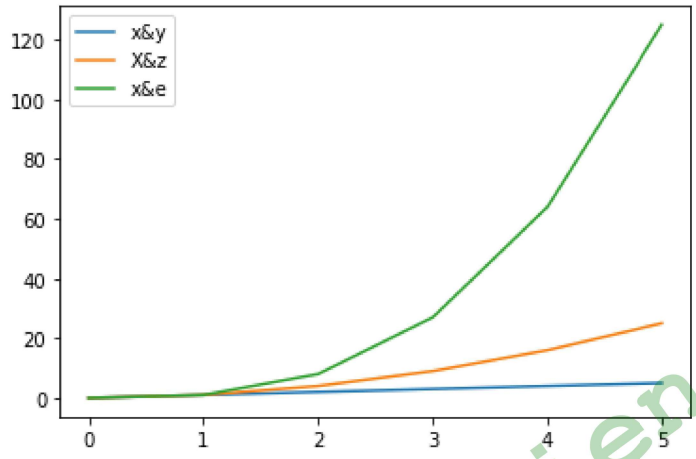
In [14]:

```
1 plt.plot(x, y)
2 plt.plot(x, z)
3 plt.plot(x, e)
4 plt.show()
```



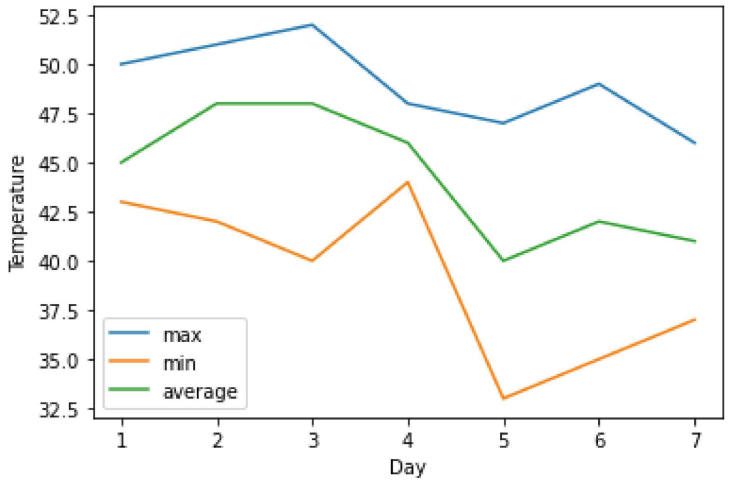
In [15]:

```
plt.plot(x, y, label='x&y')
plt.plot(x, z, label="X&z")
plt.plot(x, e, label='x&e')
plt.legend()
plt.show()
```



In [16]:

```
days=[1,2,3,4,5,6,7]
max_t=[50,51,52,48,47,49,46]
min_t=[43,42,40,44,33,35,37]
avg_t=[45,48,48,46,40,42,41]
plt.plot(days, max_t, label="max")
plt.plot(days, min_t, label="min")
plt.plot(days, avg_t, label="average")
plt.xlabel('Day')
plt.ylabel('Temperature')
plt.legend()
plt.show()
```



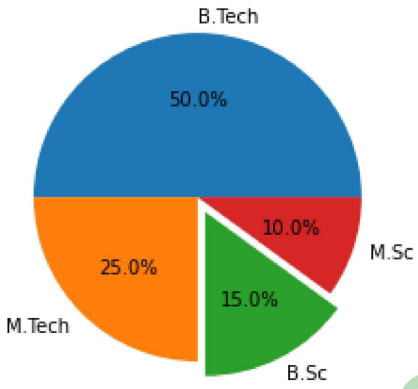
Pie Chart

In [17]:

```
education=['B.Tech', 'M.Tech', 'B.Sc', 'M.Sc']
no_of_students=[50,25,15,10]
```

In [18]:

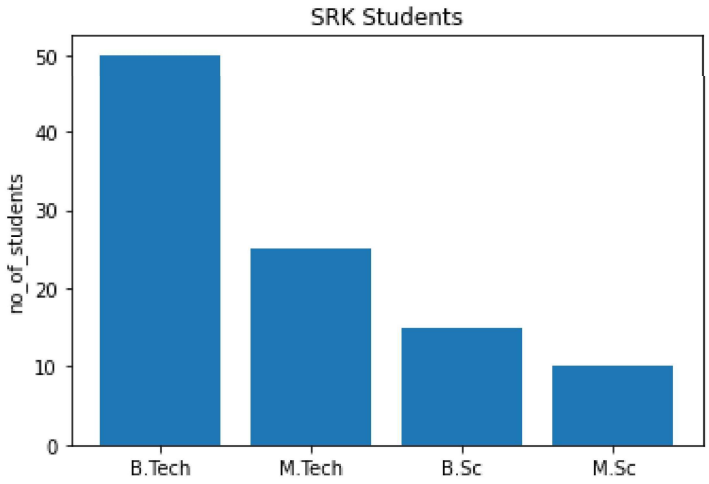
```
plt.pie(no_of_students,labels=education,autopct='%1.1f%%',explode=[0,0,0.1,0])
plt.show()
```



Bar plot

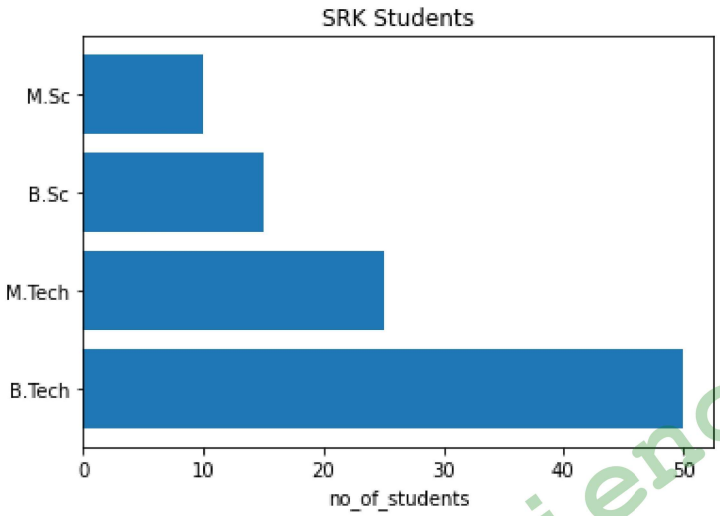
In [19]:

```
plt.bar(education,no_of_students)
plt.ylabel("no_of_students")
plt.title('SRK Students')
plt.show()
```



In [20]:

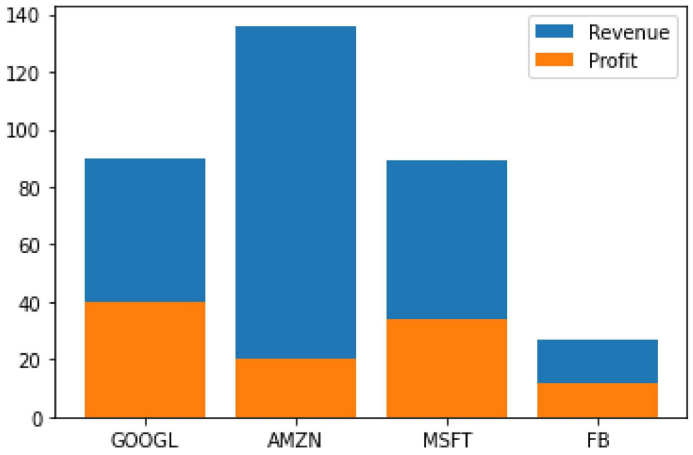
```
plt.barh(education,no_of_students)
plt.xlabel("no_of_students")
plt.title('SRK Students')
plt.show()
```



Stacked Bar plot

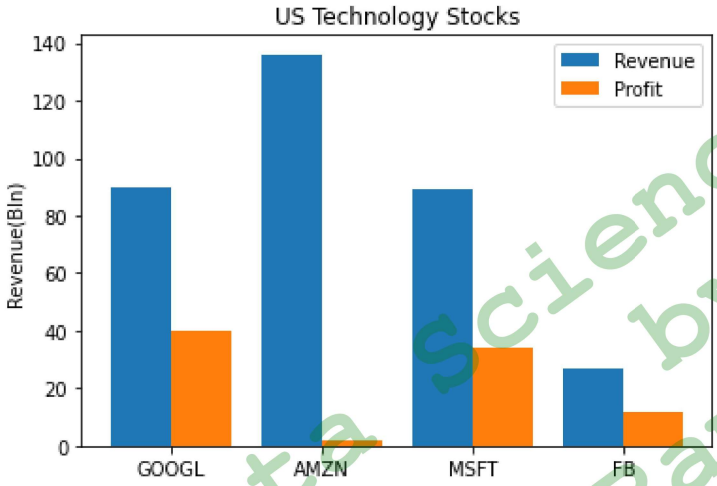
In [21]:

```
company=['GOOGL','AMZN','MSFT','FB']
revenue=[90,136,89,27]
profit=[40,20,34,12]
plt.bar(company,revenue, label="Revenue")
plt.bar(company,profit,label="Profit")
plt.legend()
plt.show()
```



In [22]:

```
company=['GOOGL','AMZN','MSFT','FB']
revenue=[90,136,89,27]
profit=[40,2,34,12]
xpos = np.arange(len(company))
plt.bar(xpos-0.2,revenue, width=0.4, label="Revenue")
plt.bar(xpos+0.2,profit, width=0.4,label="Profit")
plt.xticks(xpos,company)
plt.ylabel("Revenue(Bln)")
plt.title('US Technology Stocks')
plt.legend()
plt.show()
```



- Revenue wise --> Amazon is high
- Profit wise --> Google is high

In [23]:

```
#Program for drawing multiple bar charts on one image.
ecommerce=['Snapdeal', 'Alibaba', 'Amazon', 'Flipkart']
Q1_Profit=[45, 100, 70, 40]
Q2_Profit=[40, 105, 65, 45]
Q3_Profit=[42, 120, 72, 50]
Q4_Profit=[34, 115, 60, 48]

#Creating different bar charts on one image using subplot () function.
plt.figure(figsize=(10,10))

#Creating bar chart in first cell of figure having 3 rows, 2 columns.
plt.subplot(3,2,1)
plt.bar(ecommerce,Q1_Profit)
plt.title('Quarter1 Profit')

#Creating a bar chart in second cell.
plt.subplot(3,2,2)
plt.bar (ecommerce,Q2_Profit)
plt.title('Quarter2 Profit')

#Creating a bar chart in third cell.
plt.subplot(3,2,3)
plt.bar(ecommerce, Q3_Profit)
plt.title('Quarter3 Profit')

#Creating a bar chart in sixth cell.
plt.subplot(3,2,6)
plt.bar(ecommerce,Q4_Profit)
plt.title('Quarter4 Profit')

#Displaying the chart
plt.show ()
```