MatplotLib

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

Some of the major Pros of Matplotlib are:

- Generally easy to get started for simple plots
- Support for custom labels and texts
- Great control of every element in a figure
- High-quality output in many formats
- Very customizable in general

- **plt.bar():** Make a bar plot.
- **plt.plot():** Can be used to make a line graph.

- **plt.subplots():** Add a subplot to the current figure. It returns two arguments fig and ax where fig is figure and ax can be either a single Axes object or an array of Axes objects.

- **plt.xticks():** A list of positions at which ticks should be placed.
- **plt.xlabel(label_name):** Set the label for the x-axis.
- **plt.ylabel(label_name):** Set the label for the y-axis.
- **plt.show():** Display a figure.
- **plt.grid():** Configure the grid lines.

Refer to the [matplotlib documentation (https://matplotlib.org/3.3.3/tutorials/index.html)](https://matplotlib.org/3.3.3/tutorials/index.html) for more details.

In [1]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
df = pd.DataFrame({"Length": np.random.randn(500)})
df.head()
```

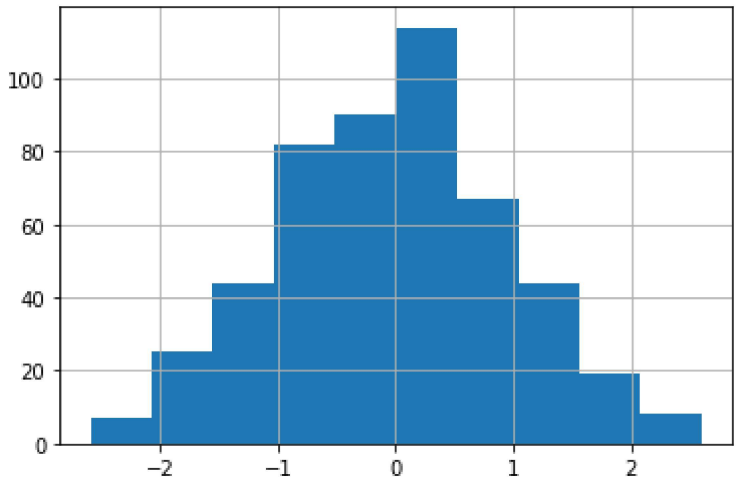
Out[2]:

	Length
0	-0.301656
1	-0.493286
2	-0.673308
3	1.006873
4	1.386056

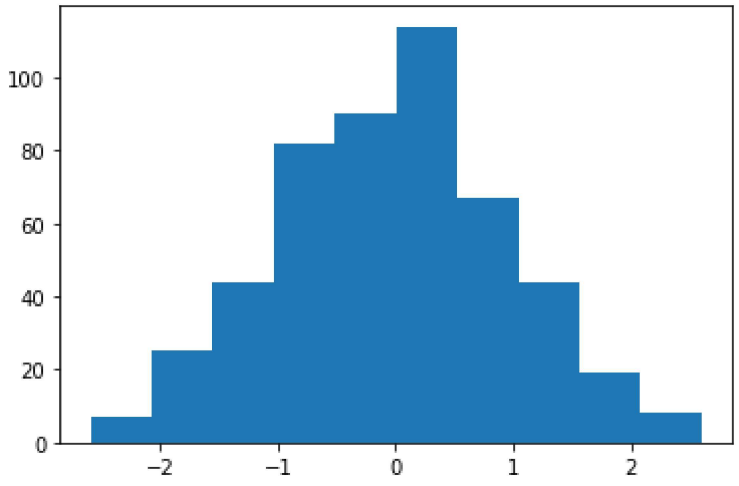
Histogram

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

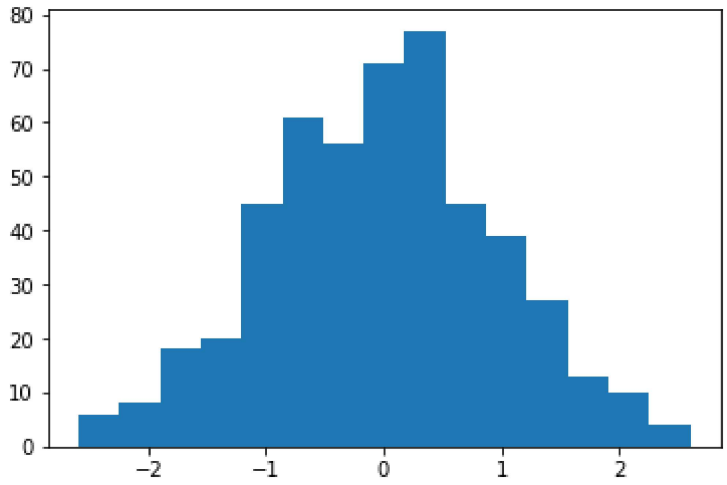
In [3]:
```
df['Length'].hist()  #using pandas
plt.show()
```



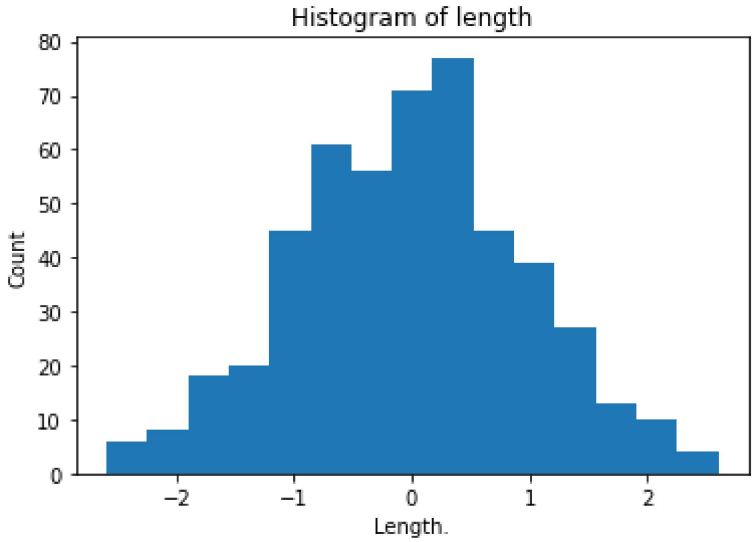
In [4]:
```
plt.hist(df['Length'])
plt.show()                          # 10 bins by default
```



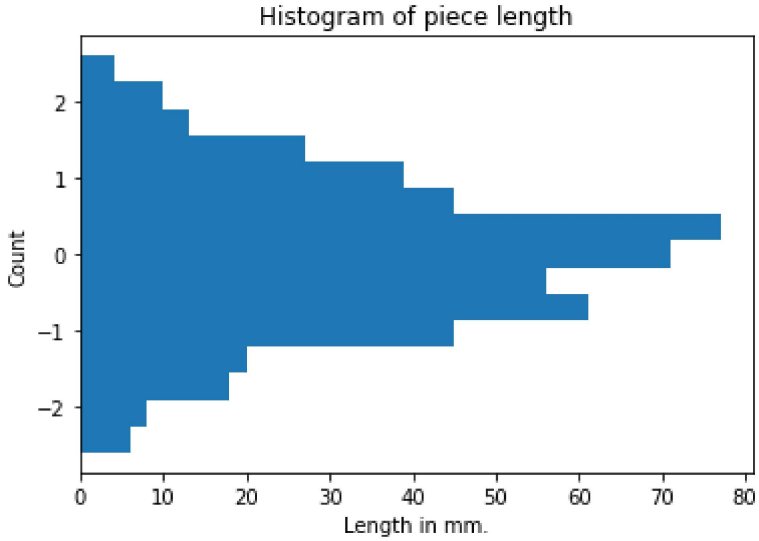
In [5]:
```
plt.hist(df['Length'], bins=15)
plt.show()
```



In [6]:
```
plt.hist(df['Length'], bins=15)
plt.xlabel("Length.")
plt.ylabel("Count")
plt.title("Histogram of length")
plt.show()
```

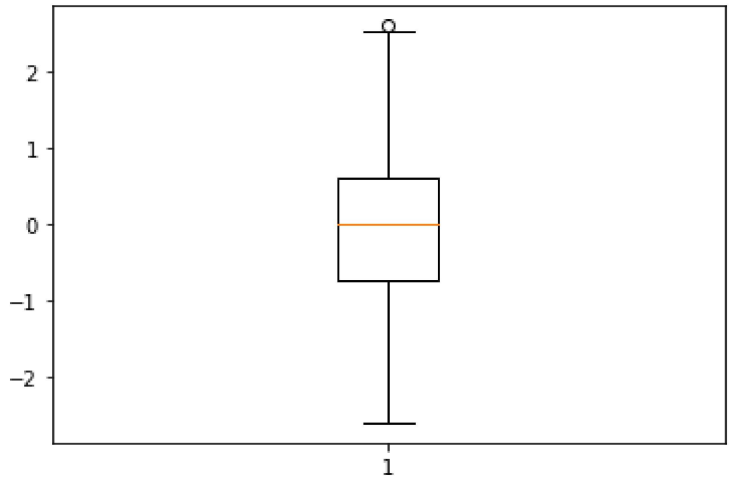


In [7]:
```
#horizontal orientatio
plt.hist(df['Length'], bins=15, orientation='horizontal')
plt.xlabel("Length in mm.")
plt.ylabel("Count")
plt.title("Histogram of piece length")
plt.show()
```

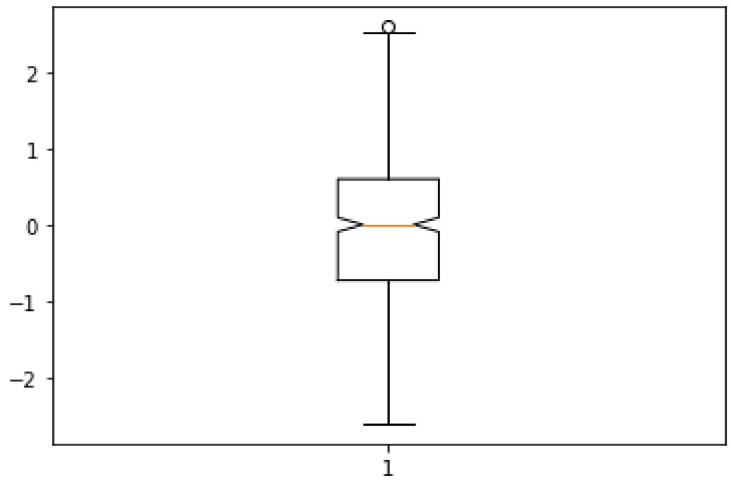


Box Plot

In [8]:
```
plt.boxplot(df['Length'])
plt.show()
```

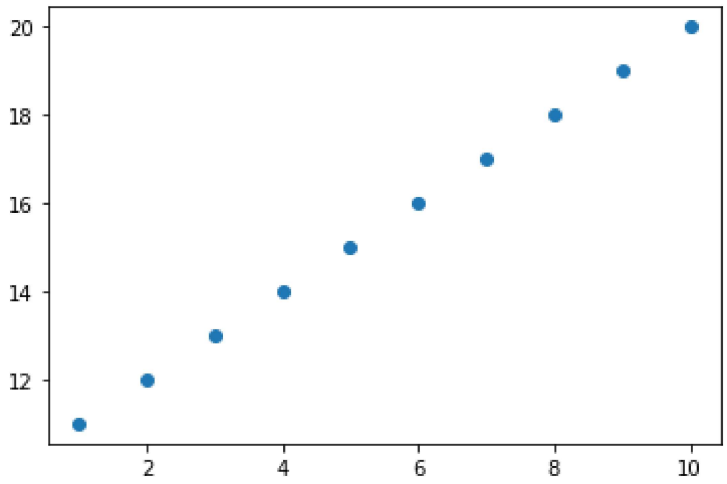


In [9]:
```
plt.boxplot(df['Length'],notch=True)
plt.show()
```

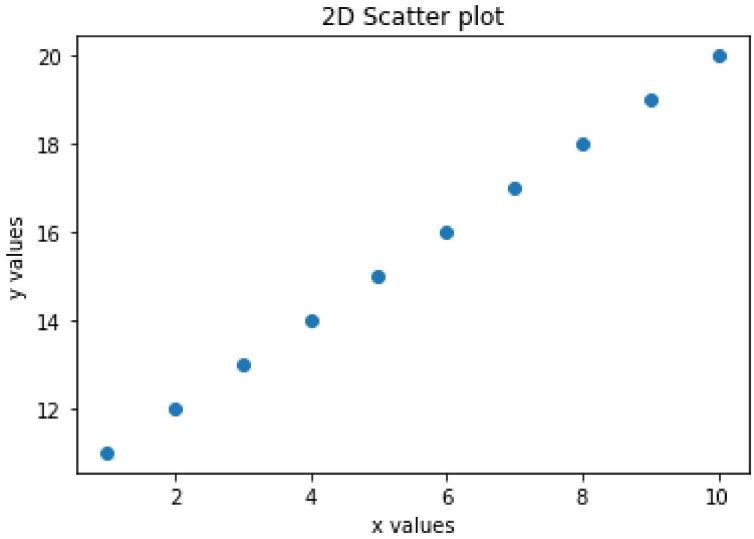


Scatter Plot

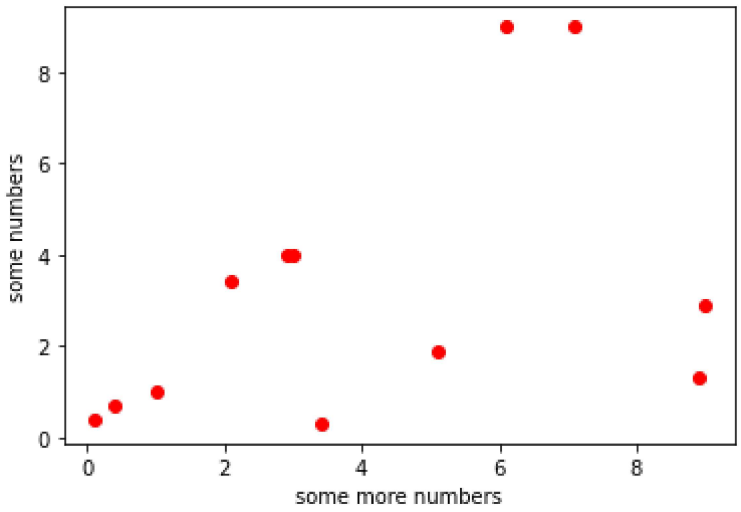
In [10]:
```
x=np.arange(1,11)
y=np.arange(11,21)
plt.scatter(x,y)
plt.show()
```



In [11]:
```
plt.scatter(x,y)
plt.xlabel('x values')
plt.ylabel('y values')
plt.title('2D Scatter plot')
plt.show()
```

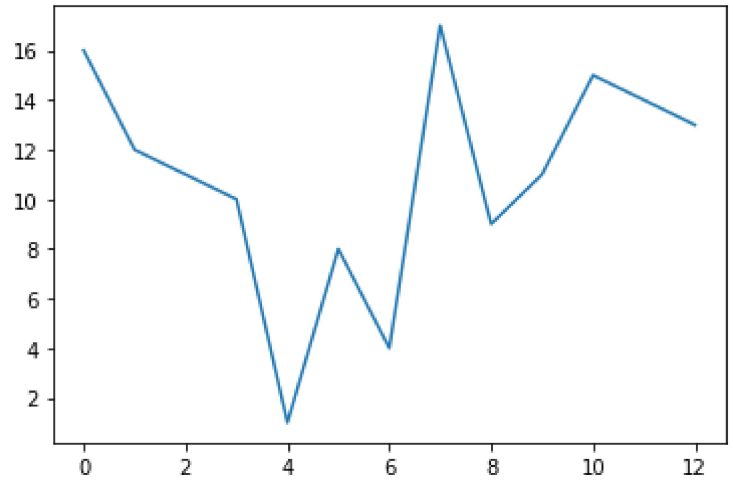


In [12]:
```
x = [1, 2.1, 0.4, 8.9, 7.1, 0.1, 3, 5.1, 6.1, 3.4, 2.9, 9]
y = [1, 3.4, 0.7, 1.3, 9, 0.4, 4, 1.9, 9, 0.3, 4.0, 2.9]
plt.scatter(x,y,color='red')
plt.ylabel('some numbers')
plt.xlabel('some more numbers')
plt.show()
```

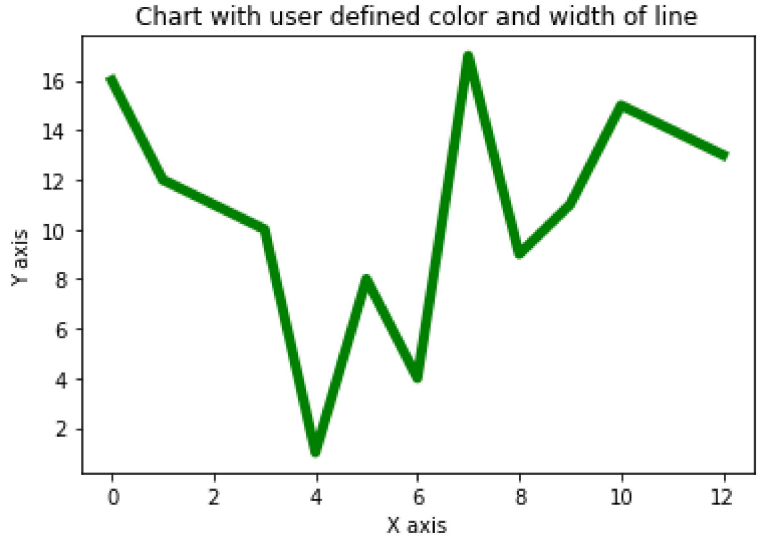


Line Plot

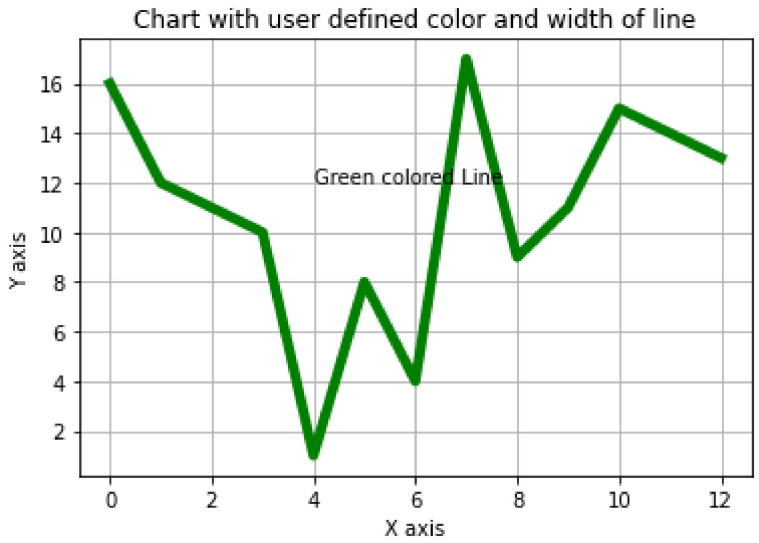
In [13]:
```
#Creating a line chart of single list
plt.plot([16,12,11,10,1, 8, 4, 17,9,11,15,14,13])
plt.show()
```



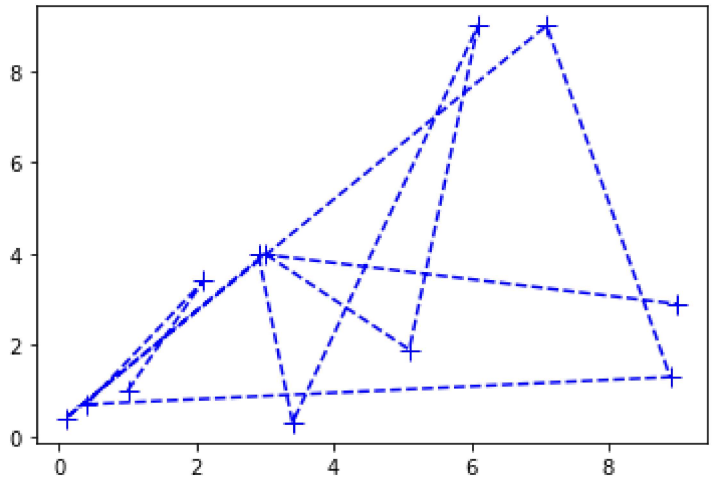
In [14]:
```
plt.plot([16,12,11,10, 1, 8, 4, 17,9,11,15,14,13],color='green',linewidth=5.0)
plt.title('Chart with user defined color and width of line')
plt.xlabel('X axis') #Label on x-axis.
plt.ylabel('Y axis') # Label on y-axis
plt.show()
```



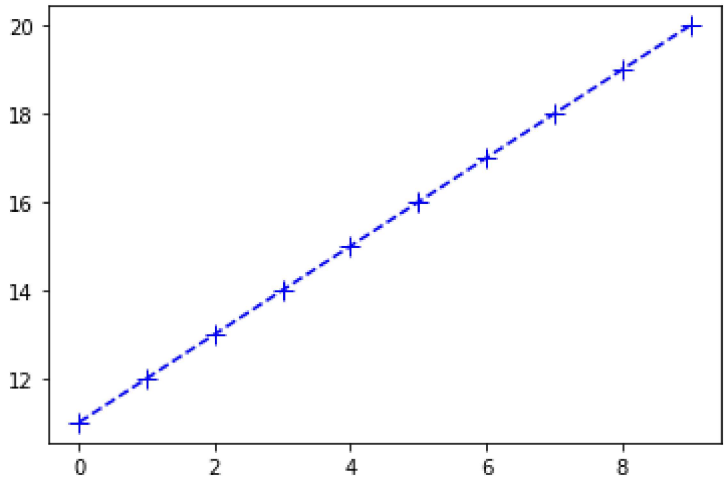
In [15]:
```
plt.plot([16,12,11,10, 1, 8, 4, 17,9,11,15,14,13],color='green',linewidth=5.0)
plt.title('Chart with user defined color and width of line')
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.grid() #Displaying a grid.
plt.text(4,12, 'Green colored Line')
plt.show()
```



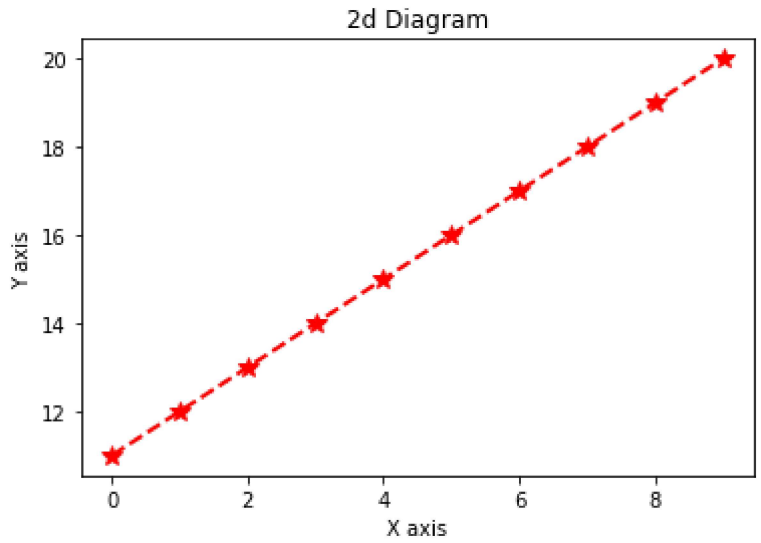
In [16]:
```
x = [1, 2.1, 0.4, 8.9, 7.1, 0.1, 3, 5.1, 6.1, 3.4, 2.9, 9]
y = [1, 3.4, 0.7, 1.3, 9, 0.4, 4, 1.9, 9, 0.3, 4.0, 2.9]
plt.plot(x,y,color='blue',marker='+',linestyle='--',markersize=10)
plt.show()
```



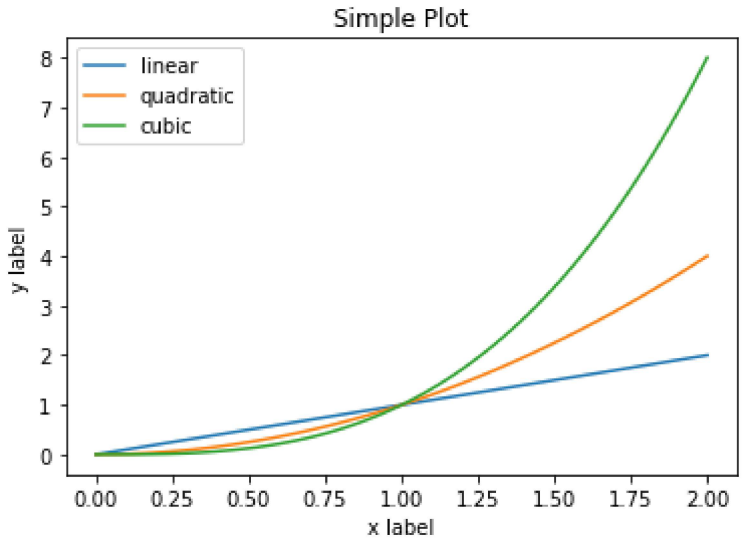
In [17]:
```
x=np.arange(0,10)
y=np.arange(11,21)
plt.plot(x,y,color='blue',marker='+',linestyle='--',markersize=10)
plt.show()
```



In [18]:
```
x=np.arange(0,10)
y=np.arange(11,21)
plt.plot(x,y,'r*',linestyle='dashed',linewidth=2, markersize=10)
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.title('2d Diagram')
plt.show()
```

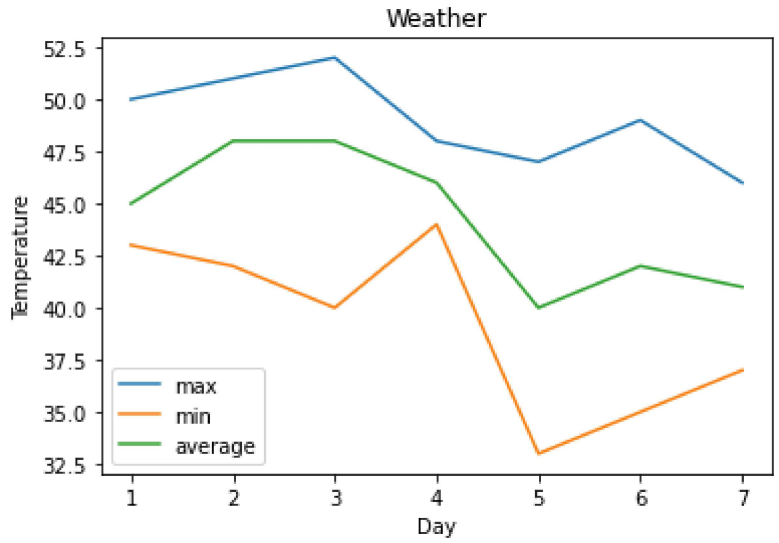


In [19]:
```
x = np.linspace(0, 2, 100)

plt.plot(x, x, label='linear')  # Plot some data on the (implicit) axes.
plt.plot(x, x**2, label='quadratic')  # etc.
plt.plot(x, x**3, label='cubic')

plt.xlabel('x label')
plt.ylabel('y label')
plt.title("Simple Plot")

plt.legend()
plt.show()
```

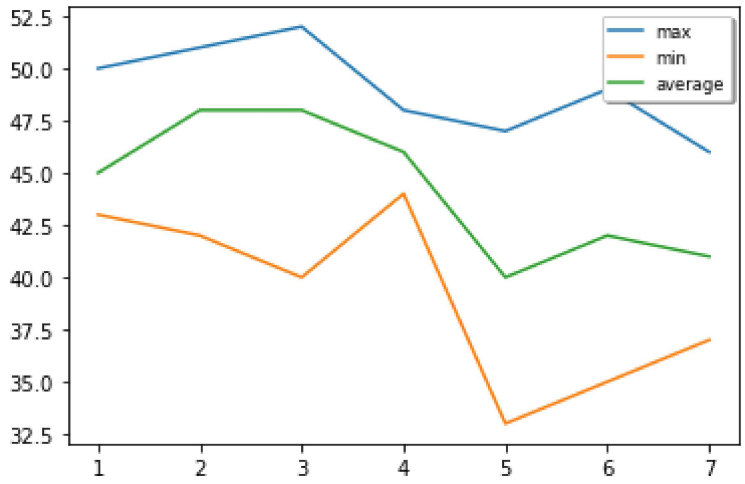


In [20]:

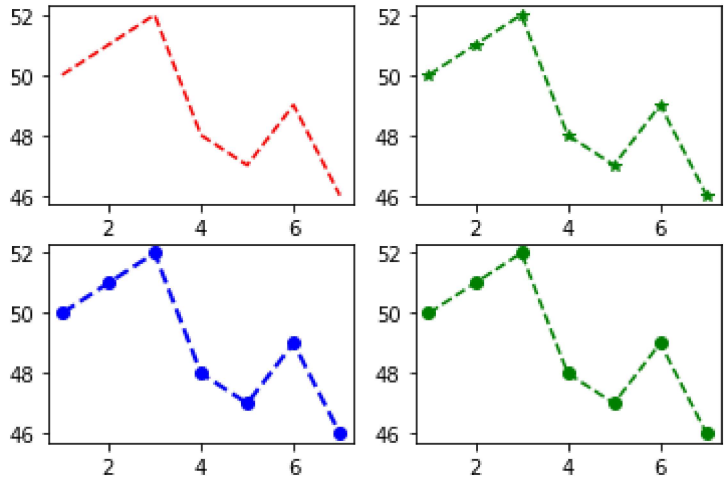
```
days=[1,2,3,4,5,6,7]
max_t=[50,51,52,48,47,49,46]
min_t=[43,42,40,44,33,35,37]
avg_t=[45,48,48,46,40,42,41]

plt.plot(days, max_t, label="max")
plt.plot(days, min_t, label="min")
plt.plot(days, avg_t, label="average")

# set axes labels and chart title
plt.xlabel('Day')
plt.ylabel('Temperature')
plt.title('Weather')

# set legend
plt.legend()
plt.show()
```



In [21]:
```
# Legend at different location with shadow enabled and fontsize set to large
plt.plot(days, max_t, label="max")
plt.plot(days, min_t, label="min")
plt.plot(days, avg_t, label="average")

plt.legend(loc='upper right',shadow=True,fontsize='small')
plt.show()
```



In [22]:
```
## Creating Subplots

x=[1,2,3,4,5,6,7]
y=[50,51,52,48,47,49,46]

plt.subplot(2,2,1)
plt.plot(x,y,'r--')

plt.subplot(2,2,2)
plt.plot(x,y,'g*--')

plt.subplot(2,2,3)
plt.plot(x,y,'bo',linewidth=2, linestyle='dashed')

plt.subplot(2,2,4)
plt.plot(x,y,'go',linestyle='dashed')

plt.show()
```

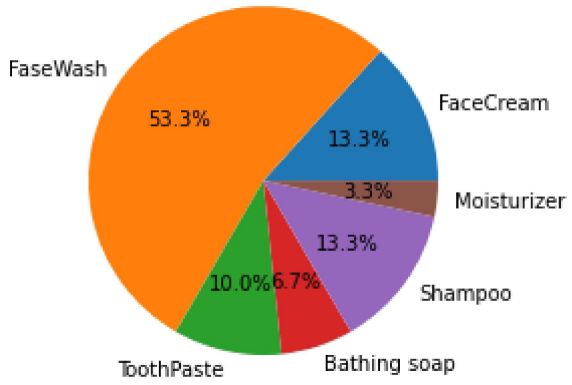


Pie Chart

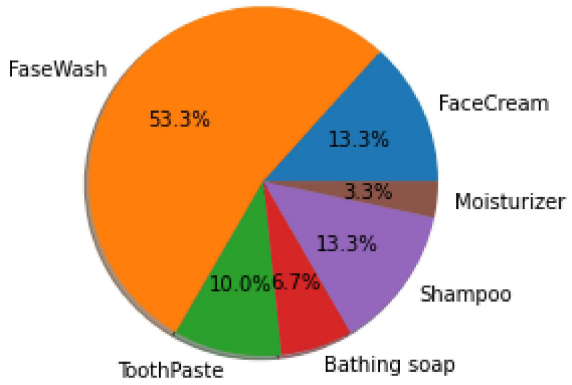
In [23]:
```
list=[20,80,15,10,20,5]
lbs = ['FaceCream', 'FaseWash', 'ToothPaste', 'Bathing soap', 'Shampoo', 'Moistur
```

In [24]:
```
#Creating and displaying a pie chart.
plt.pie(list,labels=lbs)
plt.show()
```

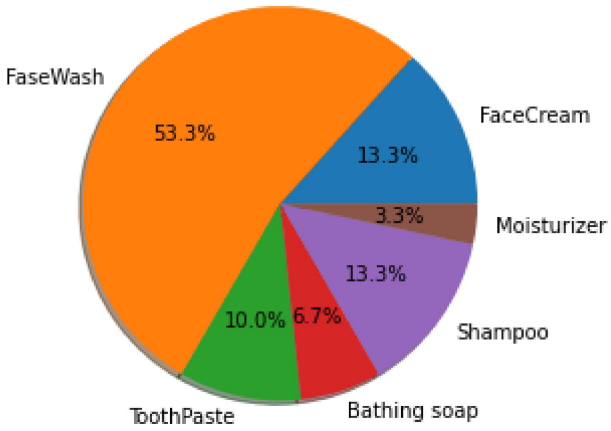
In [25]:
```
#Creating and displaying a pie chart.
plt.pie(list,labels=lbs,autopct='%1.1f%%')
plt.show()
```



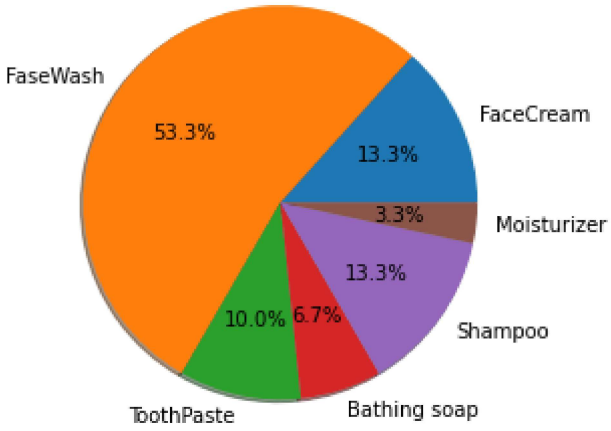
In [26]:
```
#Draw pie chart as perfect circle
plt.pie(list,labels=lbs,autopct='%1.1f%%',shadow=True)
plt.show()
```



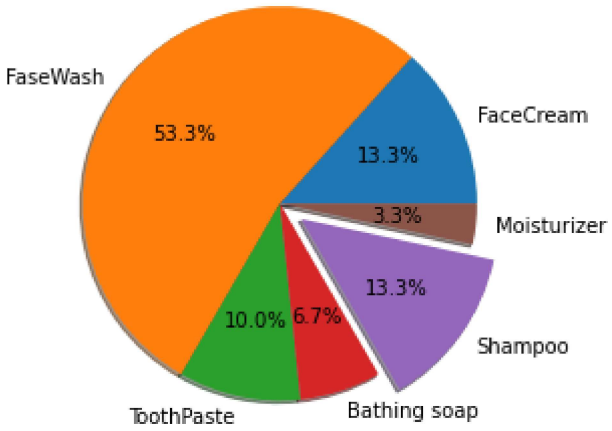
In [27]:
```
#Draw pie chart as perfect circle
plt.pie(list,labels=lbs,autopct='%1.1f%%',shadow=True)
plt.axis("equal")
plt.show()
```



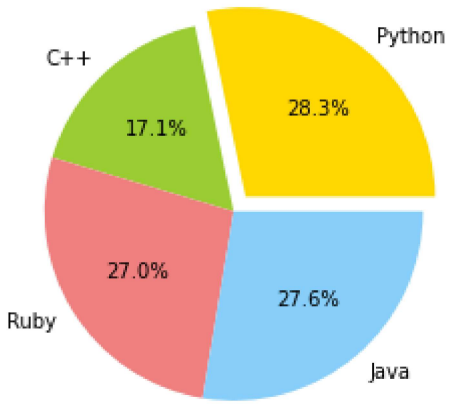
In [28]:
```
#Show percentages for every pie. Specify radius to increase chart
plt.pie(list,labels=lbs,autopct='%1.1f%%',shadow=True,radius=1.5)
plt.axis("equal")
plt.show()
```



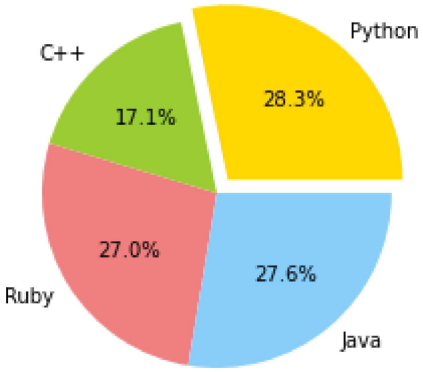
In [29]:
```
#counterclock and angle properties
plt.pie(list,labels=lbs,shadow=True, autopct='%1.1f%%',radius=1.5,explode=[0,0,0,
plt.axis("equal")
plt.show()
```



In [30]:
```
# Data to plot
la = 'Python', 'C++', 'Ruby', 'Java'
sizes = [215, 130, 205, 210]
co = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue']
explode = (0.1, 0, 0, 0)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=la, colors=co,autopct='%1.1f%%')

plt.axis('equal')
plt.show()
```

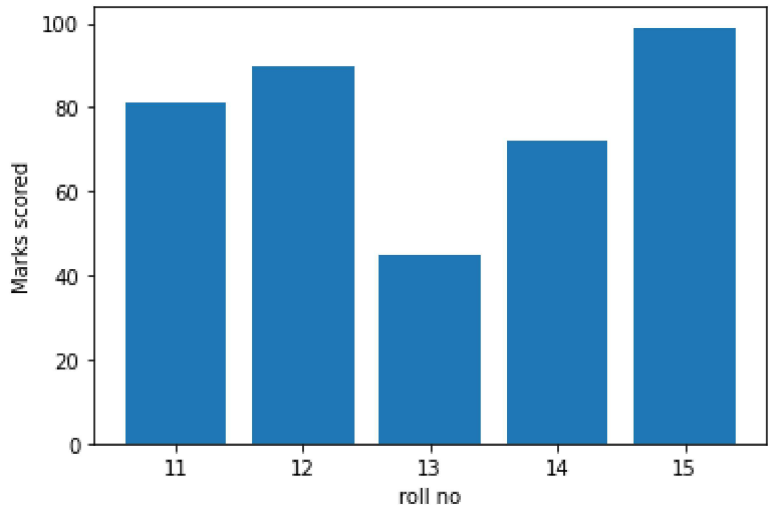


In [31]:
```
# Data to plot
la = 'Python', 'C++', 'Ruby', 'Java'
sizes = [215, 130, 205, 210]
co = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue']
explode = (0.1, 0, 0, 0)  # explode 1st slice

# Plot
plt.pie(sizes, explode=explode, labels=la, colors=co,autopct='%1.1f%%')

### Saving the Plot
plt.savefig('Plot.jpg')

plt.show()
```

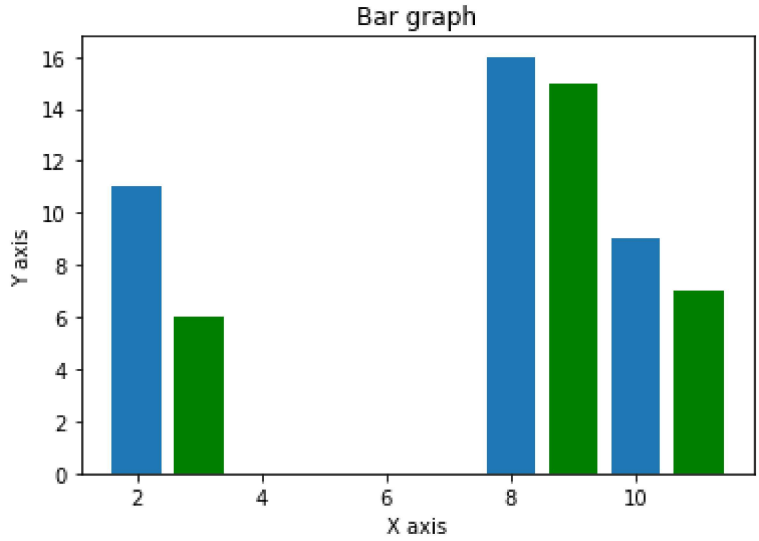


Bar plot

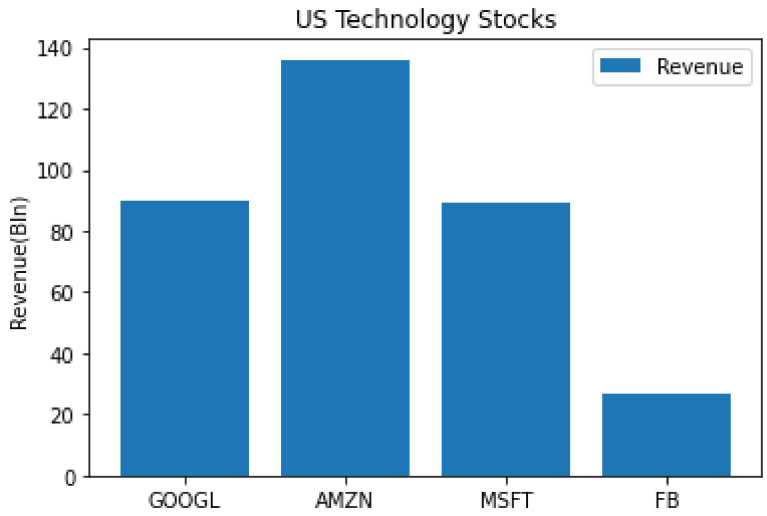
In [32]:
```
#5 students' marks are plotted on a bar graph.
marks_scored =[81,90,45,72,99]
roll_no = [11,12,13,14,15]
plt.bar(roll_no, marks_scored)
plt.xlabel('roll no')
plt.ylabel('Marks scored')
plt.show()
```



In [33]:
```
## Bar plot

x = [2,8,10]
y = [11,16,9]

x2 = [3,9,11]
y2 = [6,15,7]

plt.bar(x, y)
plt.bar(x2, y2, color = 'g')

plt.title('Bar graph')
plt.ylabel('Y axis')
plt.xlabel('X axis')

plt.show()
```

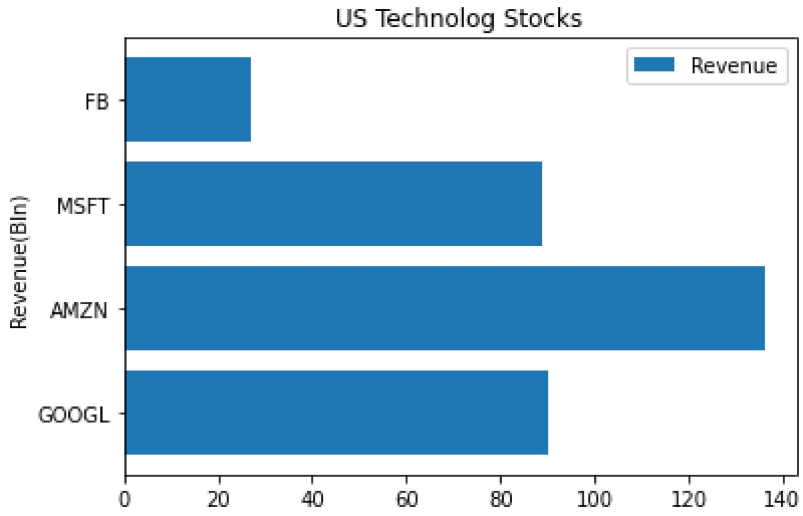


In [34]:
```
#Simple bar chart showing revenues of major US tech companies
company=['GOOGL','AMZN','MSFT','FB']
revenue=[90,136,89,27]

plt.bar(company,revenue, label="Revenue")
plt.ylabel("Revenue(Bln)")
plt.title('US Technology Stocks')
plt.legend()
plt.show()
```

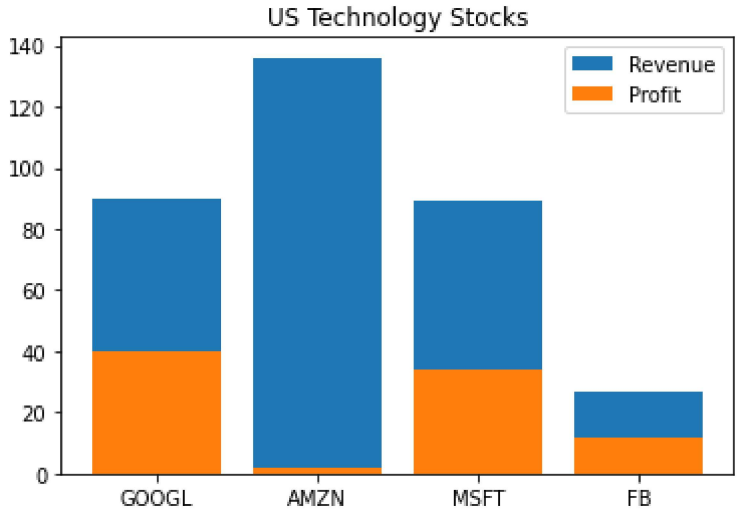


In [35]:
```
#Horizontal bar chart using barh function
plt.barh(company,revenue, label="Revenue")
plt.ylabel("Revenue(Bln)")
plt.title('US Technolog Stocks')
plt.legend()
```

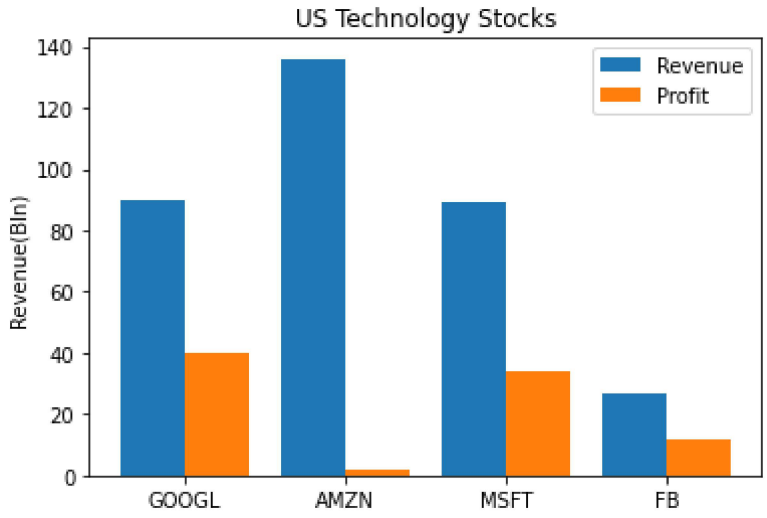
Out[35]:
```
<matplotlib.legend.Legend at 0x12b421fcc10>
```



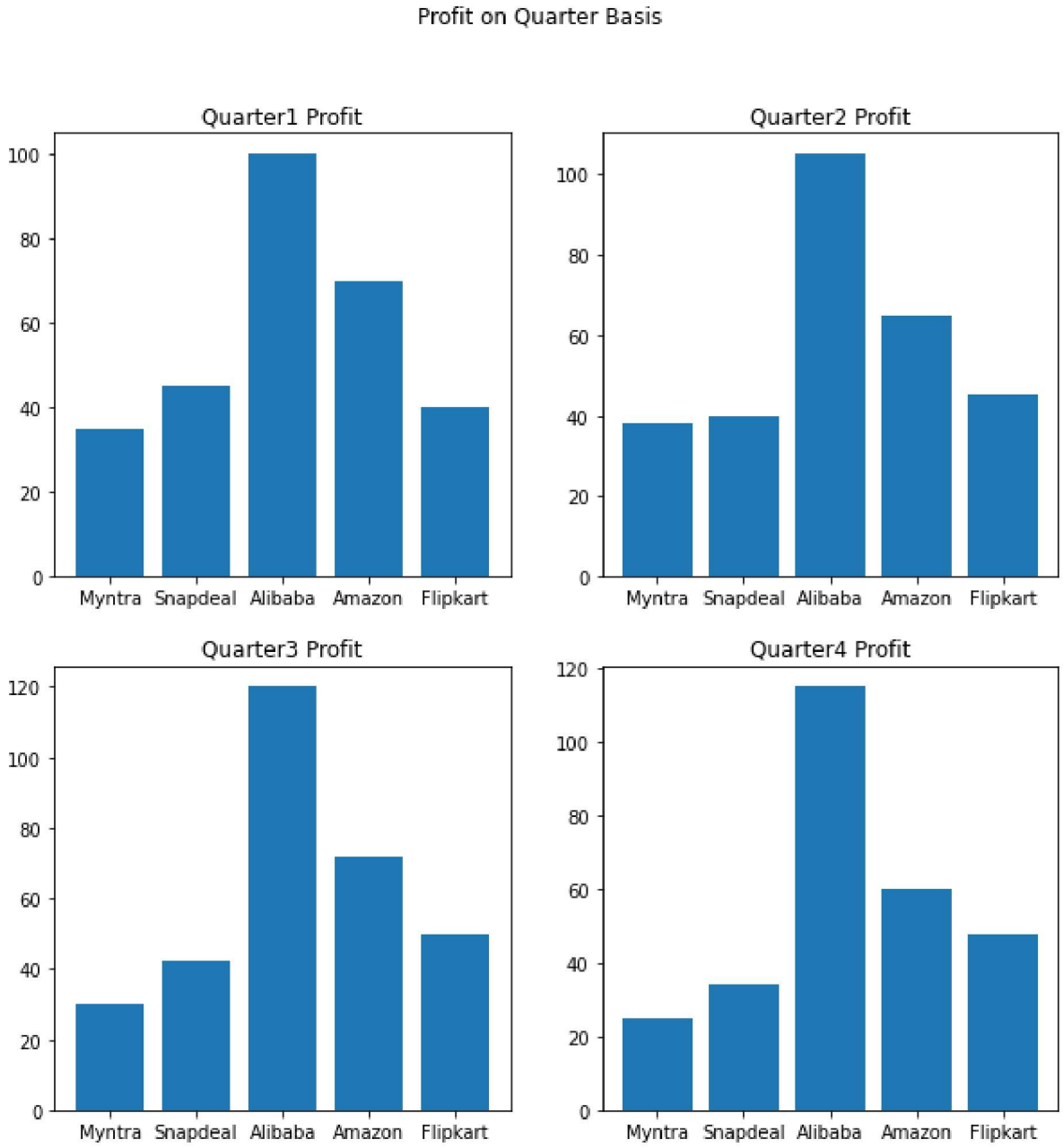
In [36]:
```
#Multiple Bars showing revenue and profit of major US tech companies
company=['GOOGL','AMZN','MSFT','FB']
revenue=[90,136,89,27]
profit=[40,2,34,12]

plt.bar(company,revenue, label="Revenue")
plt.bar(company,profit,label="Profit")

plt.title('US Technology Stocks')
plt.legend()
plt.show()
```



In [37]:
```
#Multiple Bars showing revenue and profit of major US tech companies
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()
```



In [40]:
```
#Program for drawing multiple bar charts on one image.
ecommerce=['Myntra', 'Snapdeal', 'Alibaba', 'Amazon', 'Flipkart']
Q1_Profit=[35, 45, 100, 70, 40]
Q2_Profit=[38, 40, 105, 65, 45]
Q3_Profit=[30, 42, 120, 72, 50]
Q4_Profit=[25, 34, 115, 60, 48]

#Creating different bar charts on one image using subplot () function.
plt.figure(1,figsize=(10,10))

#Creating bar chart in first cell of figure having 2 rows, 3 columns.
plt.subplot(221)
plt.bar(ecommerce,Q1_Profit)
plt.title('Quarter1 Profit')

#Creating a bar chart in second cell.
plt.subplot(222)
plt.bar (ecommerce,Q2_Profit)
plt.title('Quarter2 Profit')

#Creating a bar chart in third cell.
plt.subplot(223)
plt.bar(ecommerce, Q3_Profit)
plt.title('Quarter3 Profit')

#Creating a bar chart in fourth cell.
plt.subplot(224)
plt.bar(ecommerce, Q4_Profit)
plt.title('Quarter4 Profit')

#Adding a Main title for the figure.
plt.suptitle('Profit on Quarter Basis')

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


Profit on Quarter Basis
Quarter1 Profit
Quarter2 Profit
Quarter3 Profit
Quarter4 Profit
Myntra
Snapdeal
Alibaba
Amazon
Flipkart