Matplotlib

Matplotlib is a python library which is used for building charts for data visualization

In [175]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

%matplotlib inline
```

Important charts:

1. Vertical Bar plots
2. Horizontal Bar plots
3. Stacked Bar Plots
4. Histograms
5. Line Charts
6. Scatter Plots
7. Box Plots
8. Pie Charts
9. Subplots
10. Heatmaps

In [6]:

```
# reading the titanic dataset

df = pd.read_excel(r"C:\Users\Bhupendra\Desktop\Python Work\new_titanic.xls"
df
```

Out[6]:

	Unnamed: 0	pclass	survived	name	sex	age	sibsp	parch	ticket	fare
0	0	Upper	1	Allen, Miss. Elisabeth Walton	female	29.000000	0	0	24160	211.3375
1	1	Upper	1	Allison, Master. Hudson Trevor	male	0.916700	1	abcd	113781	151.5500
2	2	Upper	0	Allison, Miss. Helen Loraine	female	2.000000	1	2	113781	151.5500
3	3	Upper	0	Allison, Mr. Hudson Joshua Creighton	male	30.000000	1	2	113781	151.5500
4	4	Upper	0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.000000	1	2	113781	151.5500
...	...	...	...	...	...	...	...	...	...	...
1301	1304	Lower	0	Zabour, Miss. Hileni	female	14.500000	1	0	2665	14.4542
1302	1305	Lower	0	Zabour, Miss. Thamine	female	29.813199	1	0	2665	14.4542
1303	1306	Lower	0	Zakarian, Mr. Mapriededer	male	26.500000	0	0	2656	7.2250
1304	1307	Lower	0	Zakarian, Mr. Ortin	male	27.000000	0	0	2670	7.2250
1305	1308	Lower	0	Zimmerman, Mr. Leo	male	29.000000	0	0	315082	7.8750

1306 rows × 11 columns

Bar Plots

- used for categorical-numerical analysis

In [9]:

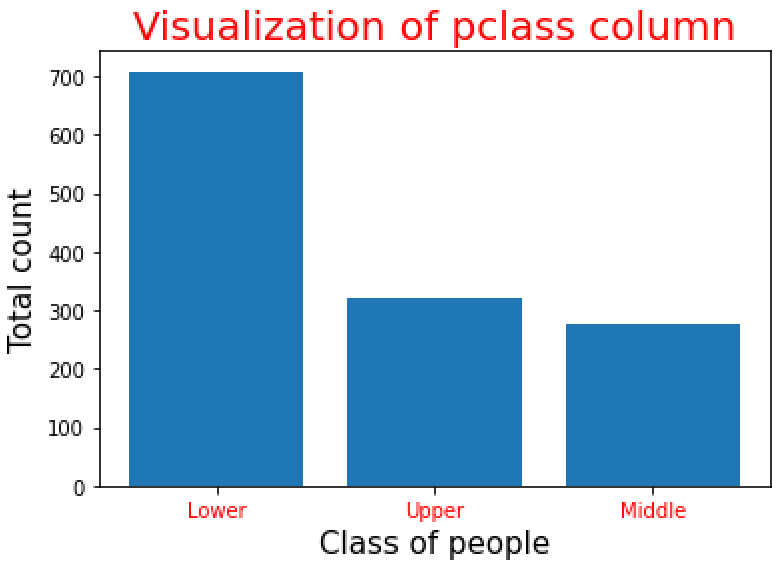
```
pc = df.pclass.value_counts()
pc
```

Out[9]:

```
Lower     708
Upper     321
Middle    277
Name: pclass, dtype: int64
```

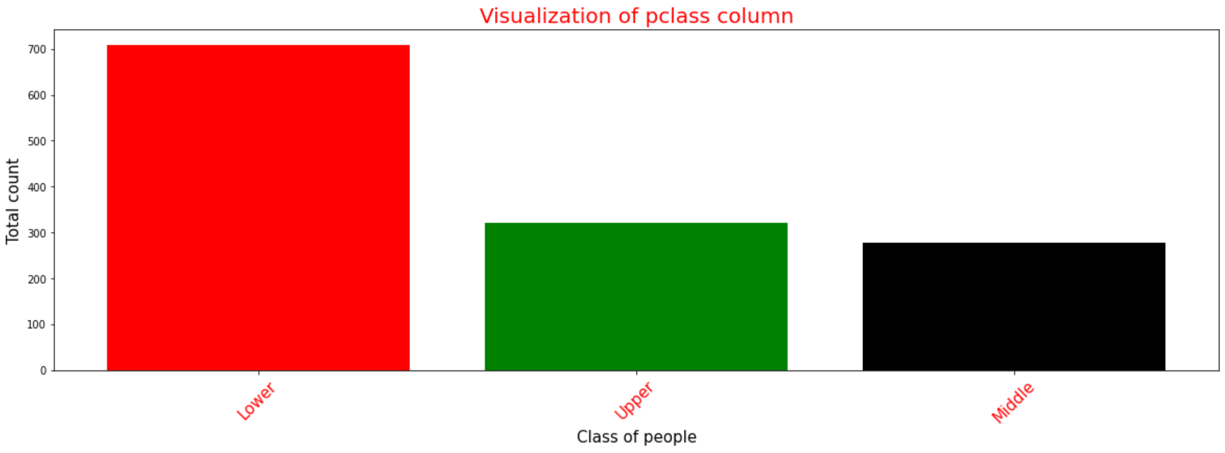
In [23]:

```
plt.bar(x = pc.index, height = pc.values, width = 0.8)
plt.xlabel('Class of people', size = 15)
plt.ylabel('Total count',size = 15)
plt.xticks(color = 'red')
plt.title("Visualization of pclass column", size = 20, color = 'red')
plt.show()
```



In [41]:

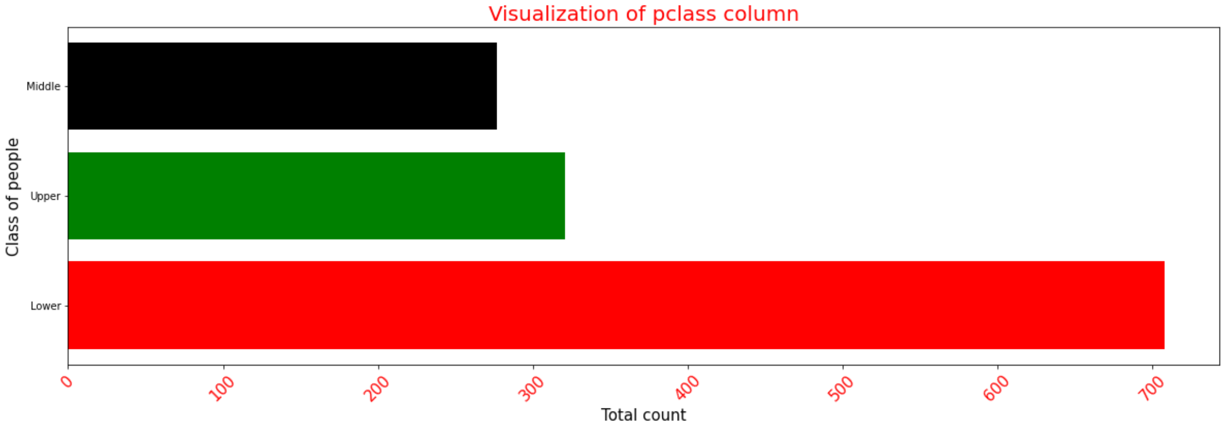
```
plt.figure(figsize = (20,6))
plt.bar(x = pc.index, height = pc.values, width = 0.8, color = ['red','green
plt.xlabel('Class of people', size = 15)
plt.ylabel('Total count',size = 15)
plt.xticks(color = 'red', rotation = 45, size = 15)
plt.title("Visualization of pclass column", size = 20, color = 'red')
plt.show()
```



Horizontal Bar charts

In [49]:

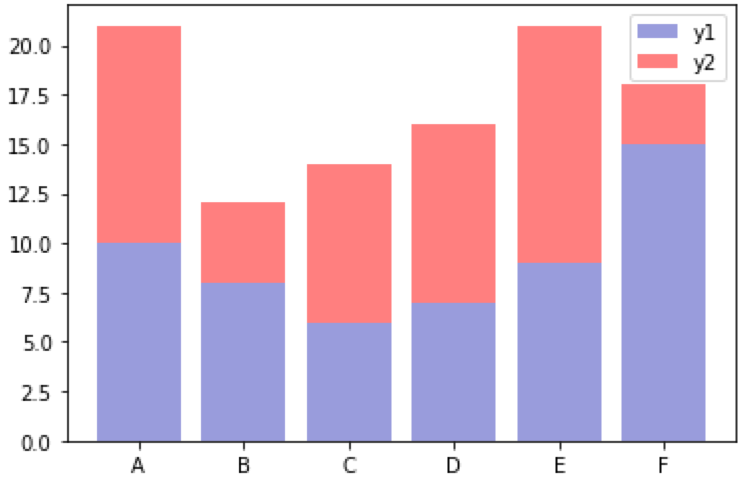
```
plt.figure(figsize = (20,6))
plt.barh(y = pc.index, width = pc.values, height = 0.8, color = ['red','gree
plt.xlabel('Total count', size = 15)
plt.ylabel('Class of people',size = 15)
plt.xticks(color = 'red', rotation = 45, size = 15)
plt.title("Visualization of pclass column", size = 20, color = 'red')
plt.show()
```



Stacked Bar Plots

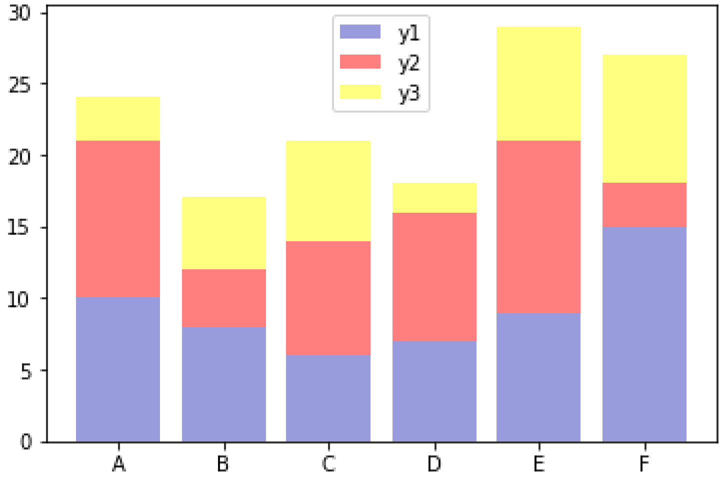
In [81]:

```
x =  ['A','B','C','D','E','F']
y1 = [10,8,6,7,9,15]
y2 = [11,4,8,9,12,3]

plt.bar(x = x, height = y1, color = '#343bba', label = 'y1', alpha = 0.5)
plt.bar(x = x, height = y2, color = 'red', label = 'y2', alpha = 0.5, bottom
plt.legend(loc = 0)      # loc = 0 will take the best location for legend
plt.show()
```



In [91]:

```
x =  ['A','B','C','D','E','F']
y1 = [10,8,6,7,9,15]
y2 = [11,4,8,9,12,3]
y3 = [3,5,7,2,8,9]

plt.bar(x = x, height = y1, color = '#343bba', label = 'y1', alpha = 0.5)
plt.bar(x = x, height = y2, color = 'red', label = 'y2', alpha = 0.5, bottom
plt.bar(x = x, height = y3, color = 'yellow', label = 'y3', alpha = 0.5, bot

plt.legend(loc = 0)
plt.show()
```

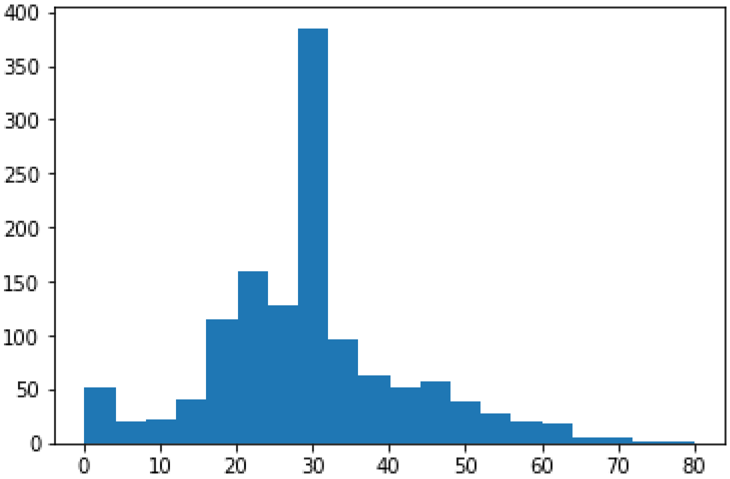


Histograms

- is used to visualize the frequnecy of numeric/categorical data

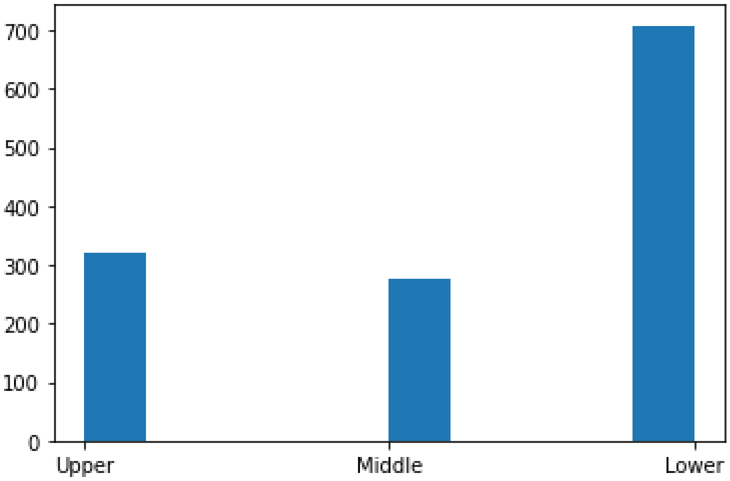
In [101]:

```
# taking numeric column(age)

plt.hist(x = df.age, bins = 20)
plt.show()
```



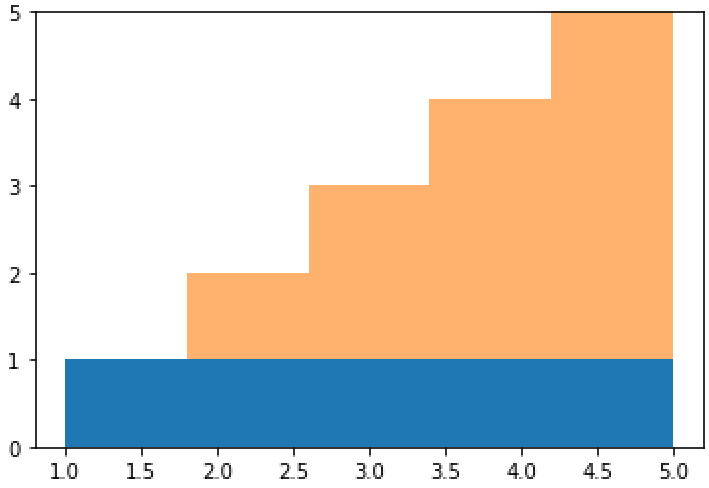
In [109]:

```
# taking categorical column(pclass)

plt.hist(x = df.pclass, bins = 10)
plt.show()
```



In [126]:

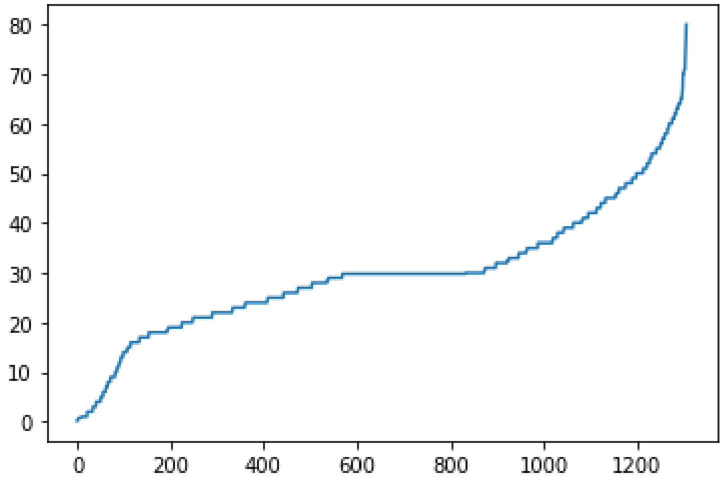
```python
# stacking histograms

y1 = [1,2,3,4,5]
y2 = [1,2,3,4,5]

plt.hist(x = y1, bins = 5)
plt.hist(x = y2, bins = 5, alpha = 0.6, bottom = y1, stacked = True)
plt.show()
```



Line Charts

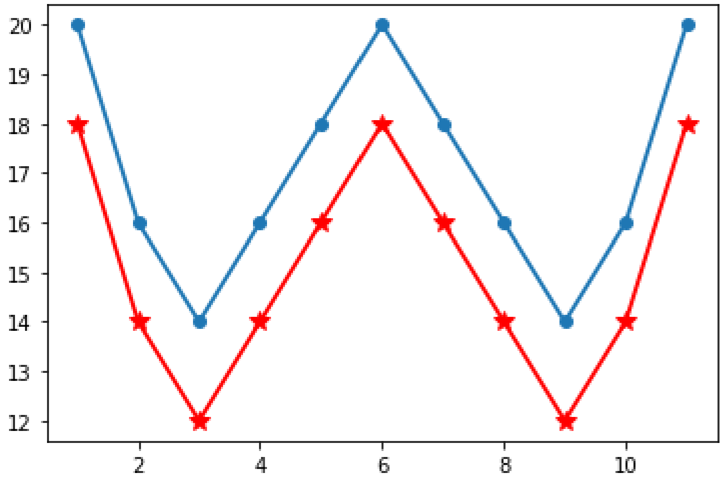
In [139]:

```python
age = df['age'].sort_values()

plt.plot(range(len(age)), age)
plt.show()
```



In [152]:

```
x = [1,2,3,4,5,6,7,8,9,10,11]
y1 = [20,16,14,16,18,20,18,16,14,16,20]
y2 = [18,14,12,14,16,18,16,14,12,14,18]

plt.plot(x,y1, lw = 2, ls = '-', marker = 'o')
plt.plot(x,y2, lw = 2, ls = '-', marker = '*', color = 'red', ms = 10)

plt.show()
```



Scatter Plots

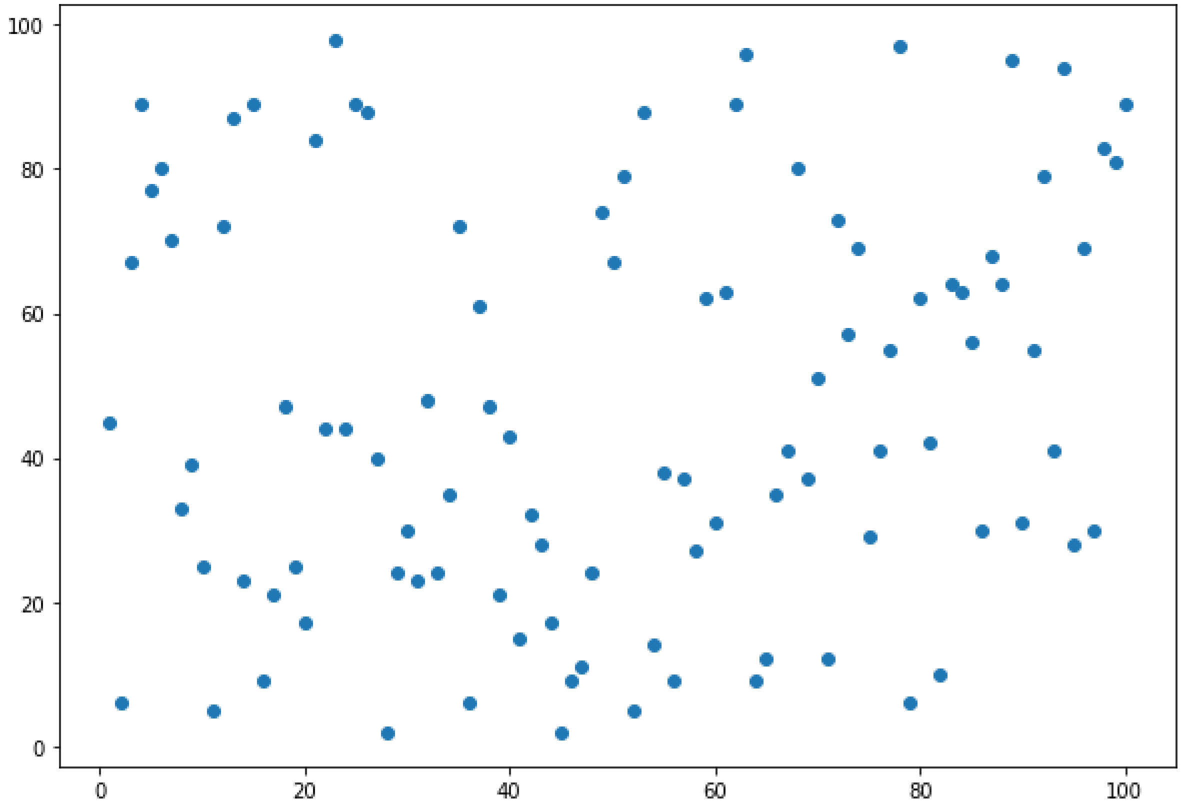
- used to check **distribution of data**
- used for continuous data
- also useful in **detecting outliers**

In [165]:

```
x = np.arange(1,101)
y = np.random.randint(1,100,100)
```

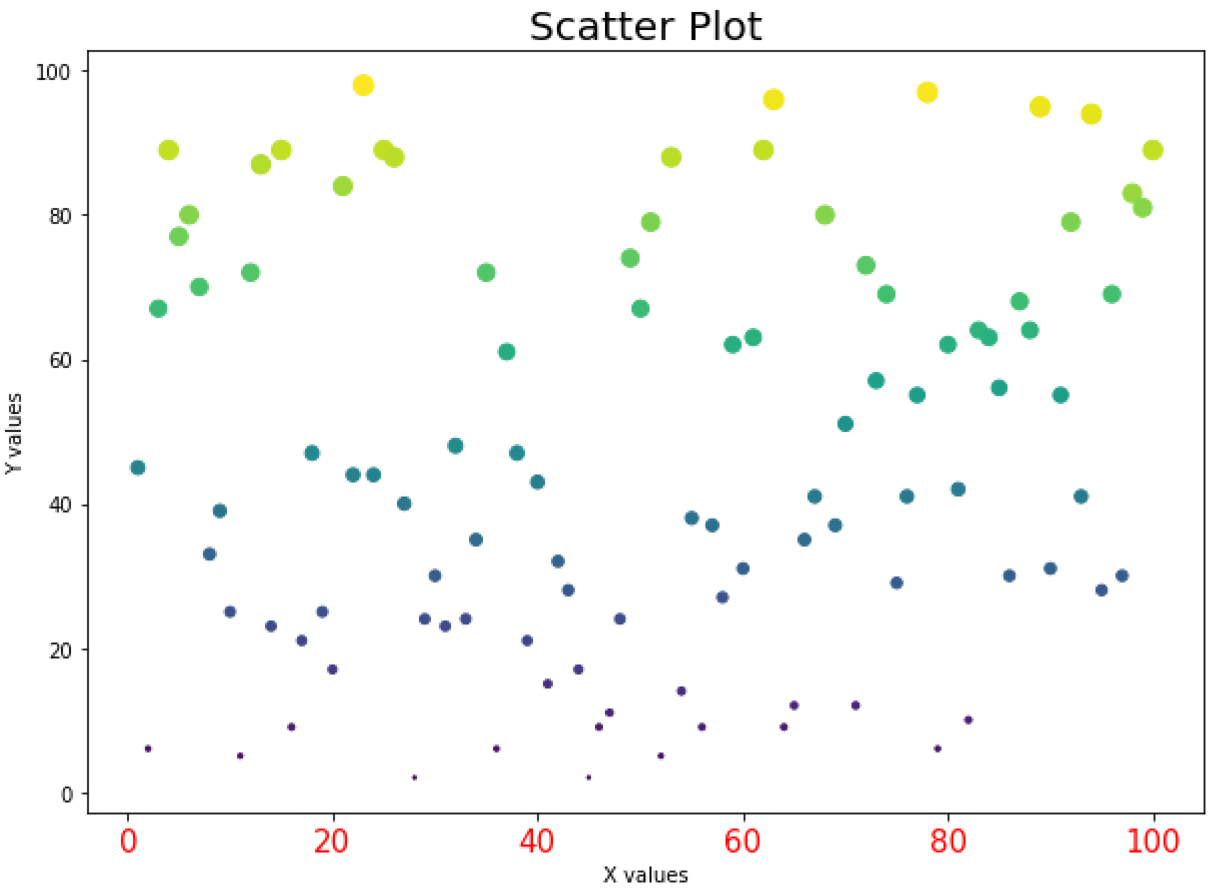
In [166]:

```
plt.figure(figsize = (10,7))
plt.scatter(x,y)
plt.show()
```



In [180]:

```
plt.figure(figsize = (10,7))
plt.scatter(x,y, s = y, c = y)   # s = size, c = color
plt.title("Scatter Plot", size = 20)
plt.xlabel('X values')
plt.ylabel('Y values')
plt.xticks(size = 15, color = 'red')
plt.savefig('myscatterplot.jpeg', dpi = 100);
```

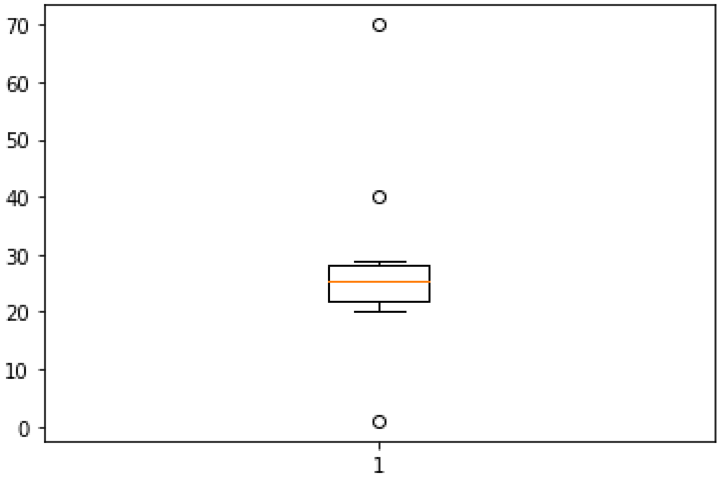


Box Plots

- very useful for getting more statistical information
- mostly preferred for detecting outliers

In [193]:

```
y = [20,25,26,21,1,28,22,22,29,27,40,70]

plt.boxplot(y)
plt.show()
```

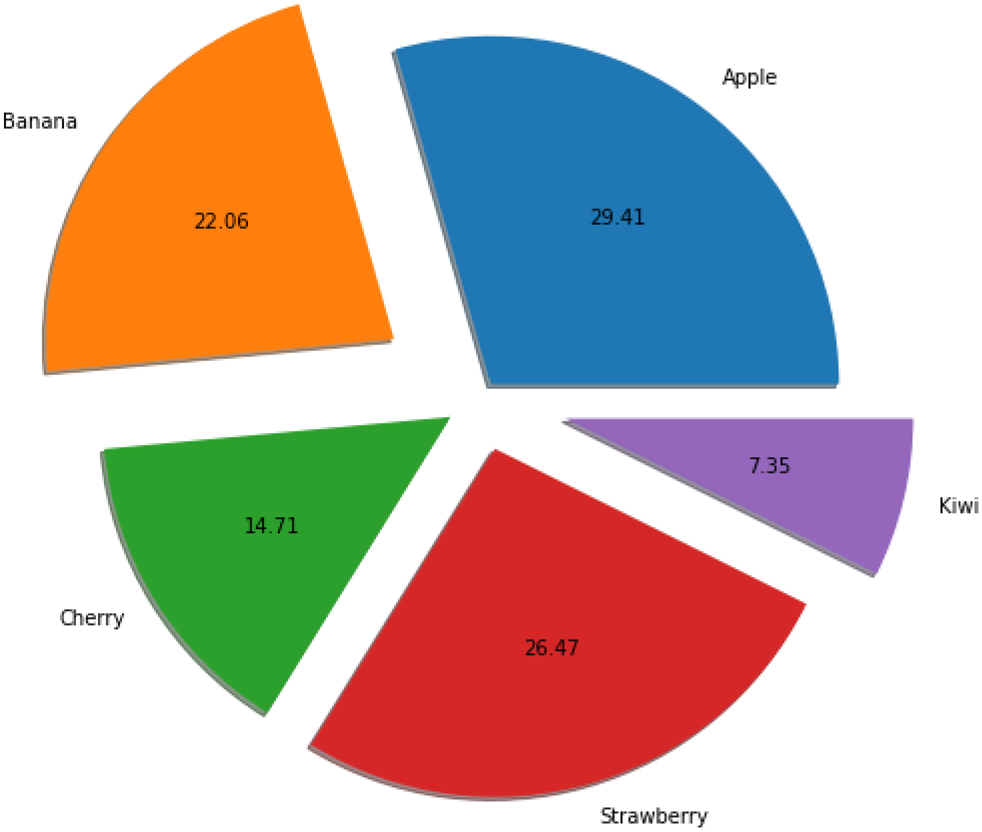


Pie Charts

- for visualizing the proportions in the data

In [217]:

```
x = ['Apple','Banana','Cherry','Strawberry','Kiwi']
y = [200,150,100,180,50]

plt.pie(y , explode = [0.1,0.6,0.2,0.3,0.5], labels = x, autopct = "%.2f", s
plt.show()
```



Heatmaps

- used for identifying the correlation of variables

In [220]:

```
df.corr()     # correlation of numeric variables
```

Out[220]:

	Unnamed: 0	survived	age	sibsp	ticket	fare
Unnamed: 0	1.000000	-0.291582	-0.299347	0.065457	0.157312	-0.480430
survived	-0.291582	1.000000	-0.051907	-0.027228	-0.023623	0.242960
age	-0.299347	-0.051907	1.000000	-0.189654	-0.044760	0.170543
sibsp	0.065457	-0.027228	-0.189654	1.000000	-0.029157	0.161030
ticket	0.157312	-0.023623	-0.044760	-0.029157	1.000000	-0.089871
fare	-0.480430	0.242960	0.170543	0.161030	-0.089871	1.000000

In [234]:

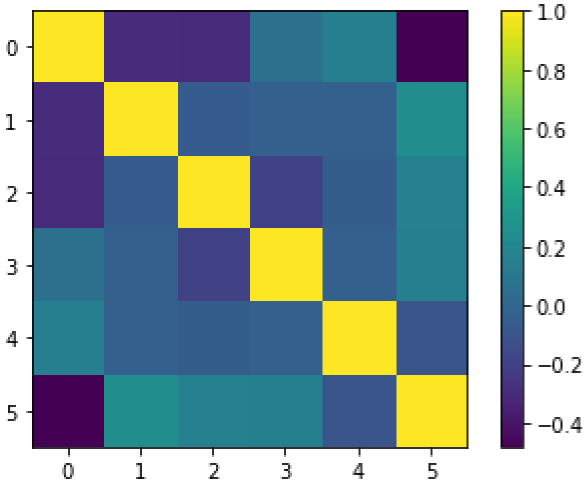
```
x = df.corr().columns
```

Out[234]:

```
Index(['Unnamed: 0', 'survived', 'age', 'sibsp', 'ticket', 'fare'], dtype='obje
ct')
```

In [240]:

```
plt.imshow(df.corr())
plt.colorbar()
plt.show()
```

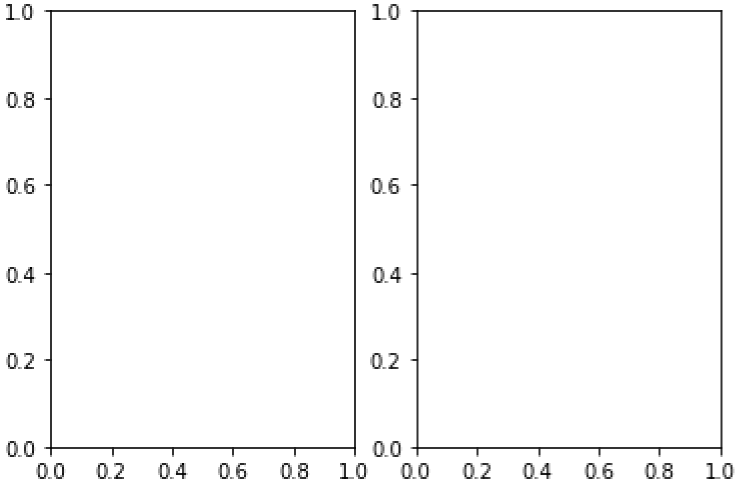


Subplots

- making multiple plots within a single plot

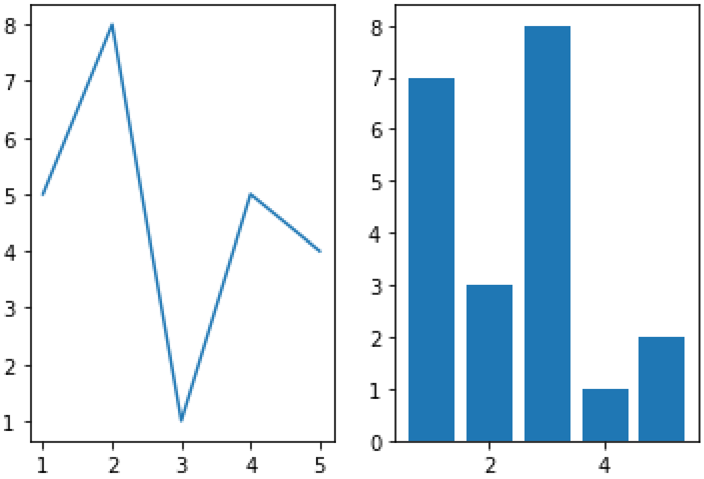
In [241]:

```
x = range(1,6)
y1 = np.random.randint(1,10,5)
y2 = np.random.randint(1,10,5)

fig, ax = plt.subplots(1,2)    # creating empty subplots area
```



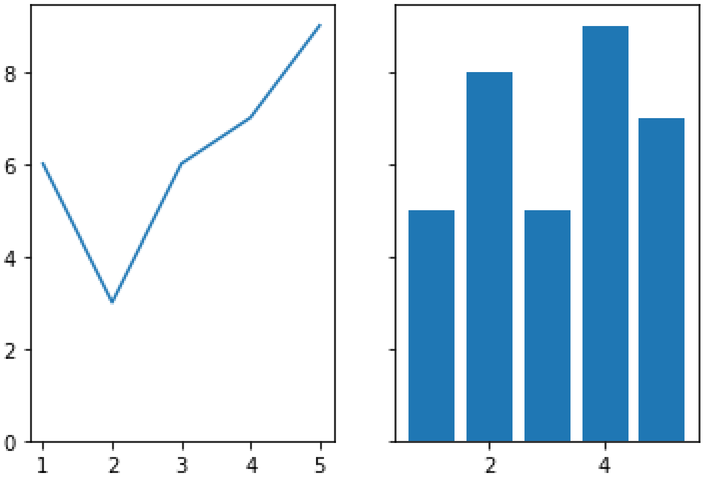
In [273]:

```
x = range(1,6)
y1 = np.random.randint(1,10,5)
y2 = np.random.randint(1,10,5)

fig, (ax1,ax2) = plt.subplots(1,2)   # creating empty subplots area

ax1.plot(x,y1)
ax2.bar(x,y2)
plt.show()
```



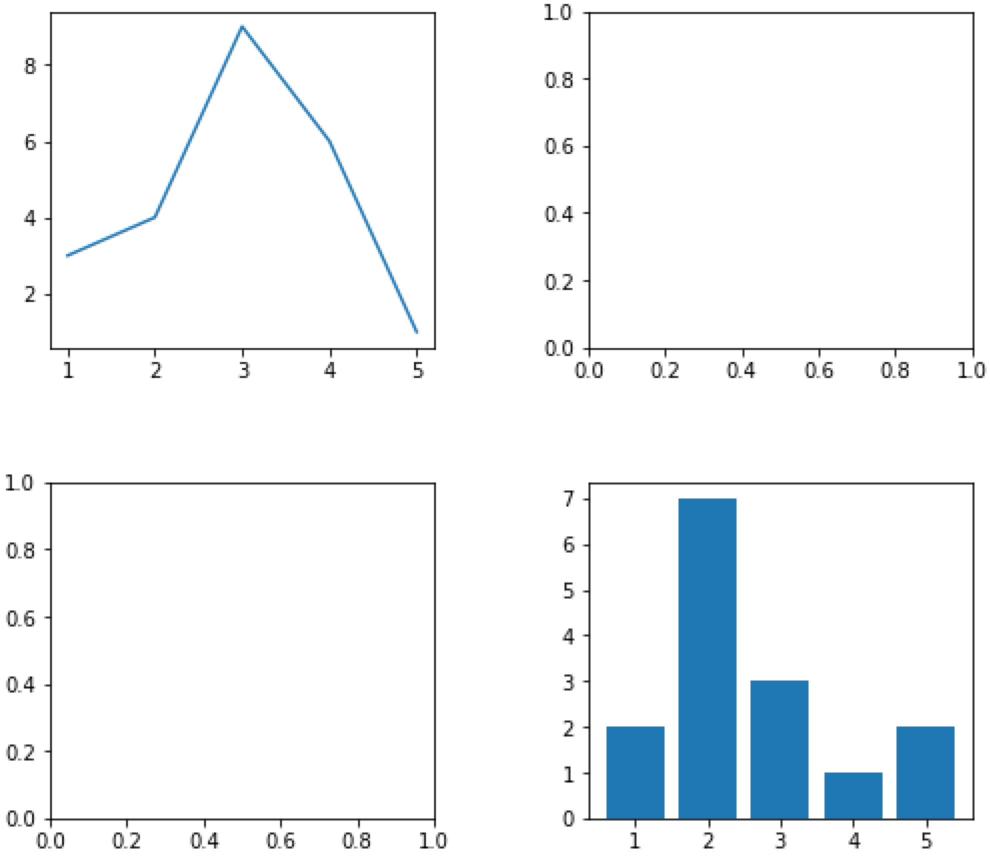
In [274]:

```
x = range(1,6)
y1 = np.random.randint(1,10,5)
y2 = np.random.randint(1,10,5)

fig, ax = plt.subplots(1,2, sharey = True)   # creating empty subplots area

ax[0].plot(x,y1)
ax[1].bar(x,y2)
plt.show()
```



In [275]:

```
x = range(1,6)
y1 = np.random.randint(1,10,5)
y2 = np.random.randint(1,10,5)

fig, ax = plt.subplots(2,2, figsize = (8,7))   # creating empty subplots are

ax[0,0].plot(x,y1)
ax[1,1].bar(x,y2)

# adjusting the subplots
plt.subplots_adjust(left=0.1,
                    bottom=0.1,
                    right=0.9,
                    top=0.9,
                    wspace=0.4,
                    hspace=0.4)
plt.show()
```



In [272]:

```python
help(plt.subplots_adjust)
```

```
Help on function subplots_adjust in module matplotlib.pyplot:

subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspa
ce=None)
    Adjust the subplot layout parameters.
    
    Unset parameters are left unmodified; initial values are given by
    :rc:`figure.subplot.[name]`.
    
    Parameters
    ----------
    left : float, optional
        The position of the left edge of the subplots,
        as a fraction of the figure width.
    right : float, optional
        The position of the right edge of the subplots,
        as a fraction of the figure width.
    bottom : float, optional
        The position of the bottom edge of the subplots,
        as a fraction of the figure height.
    top : float, optional
        The position of the top edge of the subplots,
        as a fraction of the figure height.
    wspace : float, optional
        The width of the padding between subplots,
        as a fraction of the average axes width.
    hspace : float, optional
        The height of the padding between subplots,
        as a fraction of the average axes height.
```

Explore more charts from here :

https://matplotlib.org/stable/gallery/index.html (https://matplotlib.org/stable/gallery/index.html)