Visualization

- matplotlib
- seaborn
- plotly
- bokeh
- folium

Ploting

- Univariate (single variable)
- Bivariate (for two variables)
- ctrl+MP (to save notebook in pdf format)

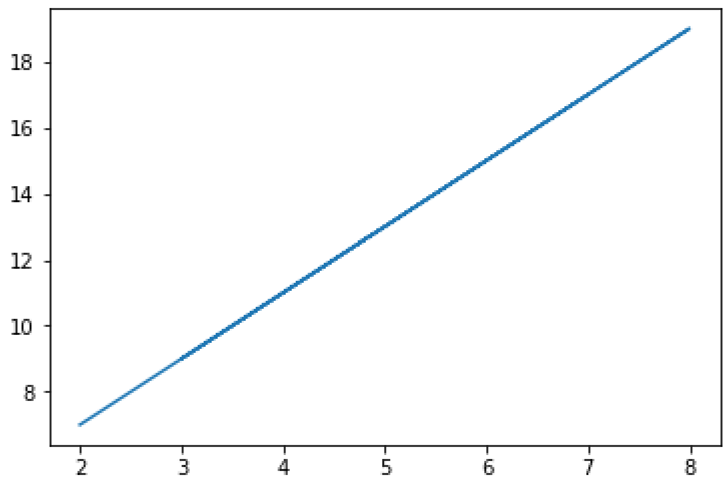
The following color abbreviations are supported:

```
=============    ===============================
character        color
=============    ===============================
``'b'``          blue
``'g'``          green
``'r'``          red
``'c'``          cyan
``'m'``          magenta
``'y'``          yellow
``'k'``          black
``'w'``          white
```

In [5]:
```
%matplotlib inline

import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
```

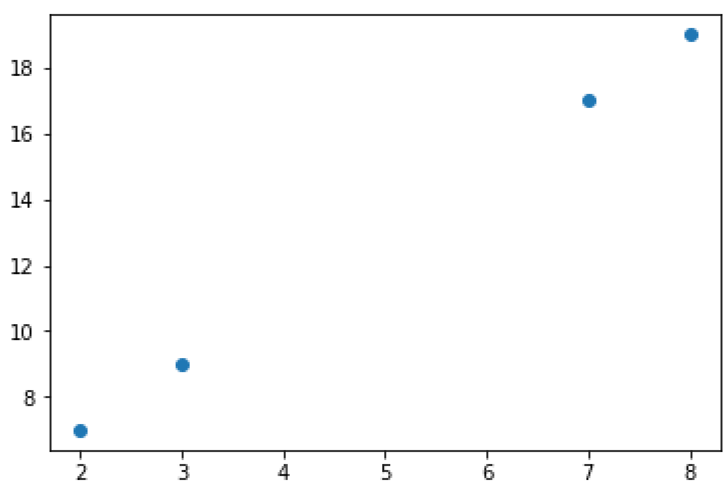
In [6]:
```
x=np.array([2,8,3,7])
y=x*2+3
#(2,8,3,7)(7,19,9,17)
#(2,7)(8,19)(3,9)(7,17)
plt.plot(x,y)  #(bydefault size is blue)
```

Out[6]: [<matplotlib.lines.Line2D at 0x26a15088908>]



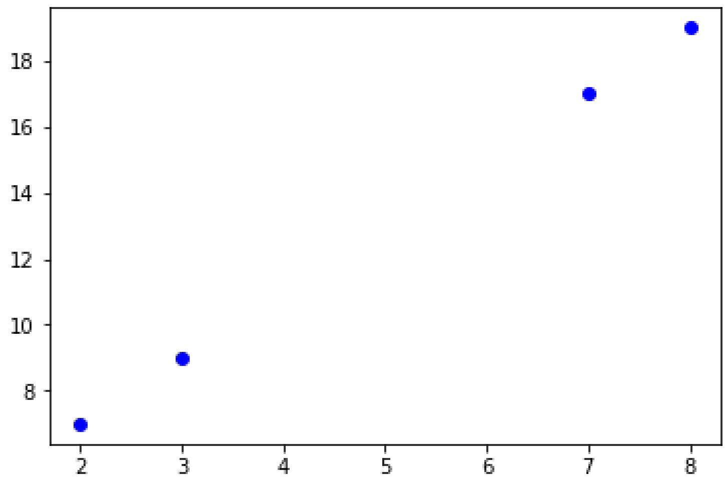
In [7]:
```
plt.plot(x,y,'o')
```

Out[7]: [<matplotlib.lines.Line2D at 0x26a15129f60>]



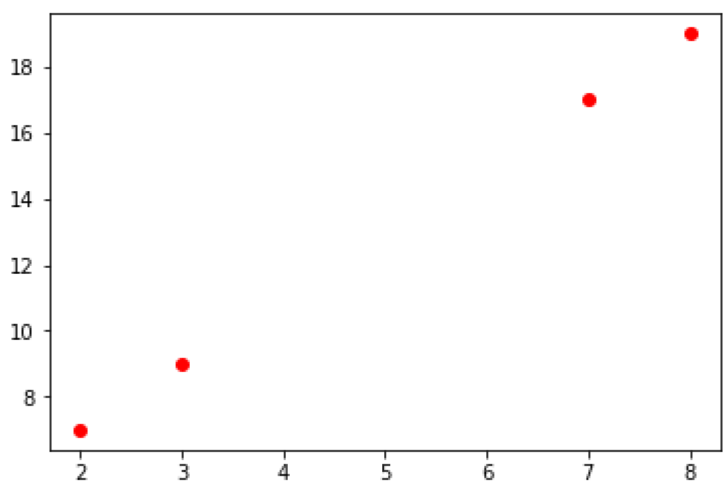
In [8]:
```
plt.plot(x,y,'bo')
```

Out[8]: [<matplotlib.lines.Line2D at 0x26a161696a0>]



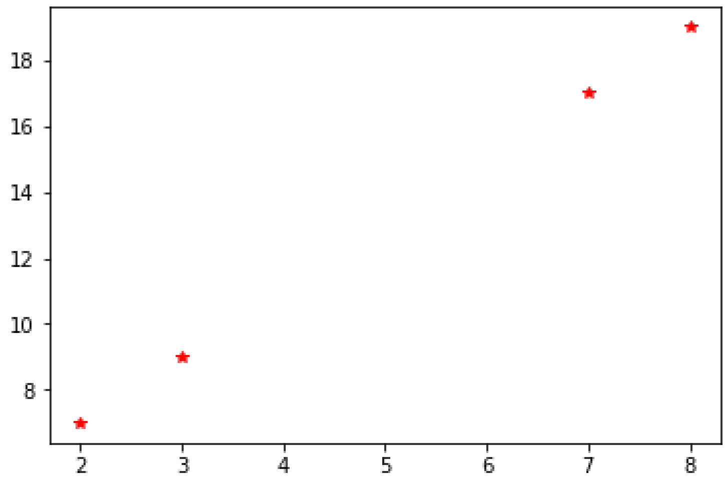
In [9]:
```
plt.plot(x,y,'ro')
```

Out[9]: [<matplotlib.lines.Line2D at 0x26a161c59e8>]



In [10]:
```
plt.plot(x,y,'r*')   # *,+,o shapes of marker,, r,g,b,--red,green,blue,, c-cyan,r
```

Out[10]:
```
[<matplotlib.lines.Line2D at 0x26a16225d30>]
```

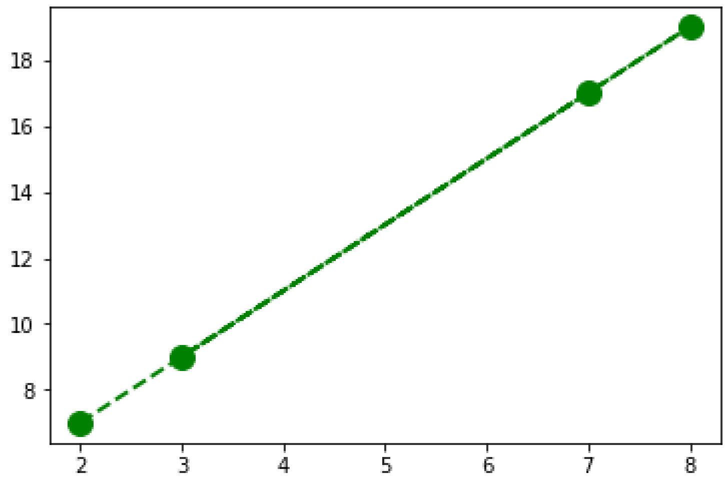


In [11]:
```
help(plt.plot)
```

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

plot(*args, scalex=True, scaley=True, data=None, **kwargs)
    Plot y versus x as lines and/or markers.

    Call signatures::

        plot([x], y, [fmt], data=None, **kwargs)
        plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs)

    The coordinates of the points or line nodes are given by *x*, *y*.

    The optional parameter *fmt* is a convenient way for defining basic
    formatting like color, marker and linestyle. It's a shortcut string
    notation described in the *Notes* section below.

    >>> plot(x, y)        # plot x and y using default line style and color
    >>> plot(x, y, 'bo')  # plot x and y using blue circle markers
    >>> plot(y)           # plot y using x as index array 0..N-1
```

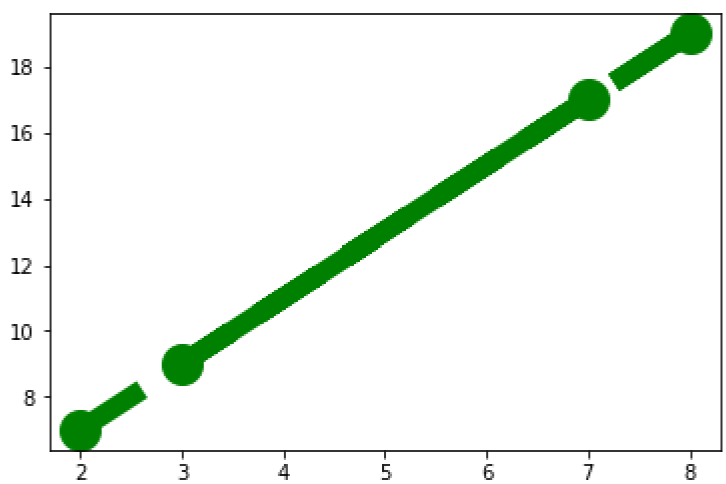
In [12]:
```
plt.plot(x, y, color='green', marker='o', linestyle='dashed',linewidth=2, marker
```

Out[12]: [<matplotlib.lines.Line2D at 0x26a16290198>]



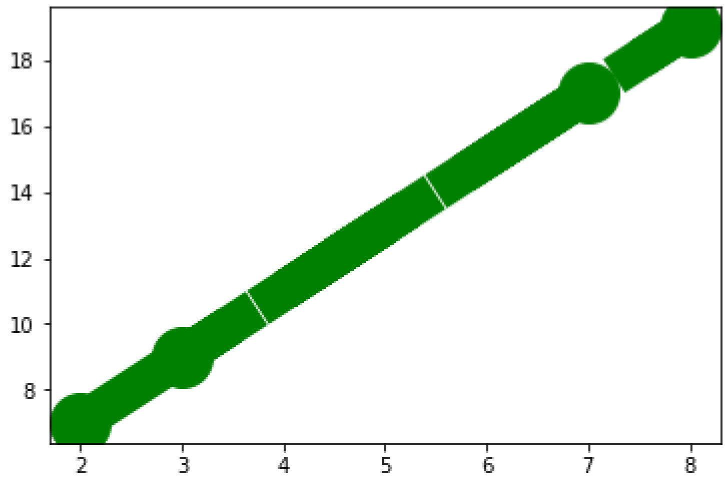
In [13]:
```
plt.plot(x, y, 'go--', linewidth=10, markersize=20)  #markersize-->circle size,l
```

Out[13]: [<matplotlib.lines.Line2D at 0x26a162e9c88>]



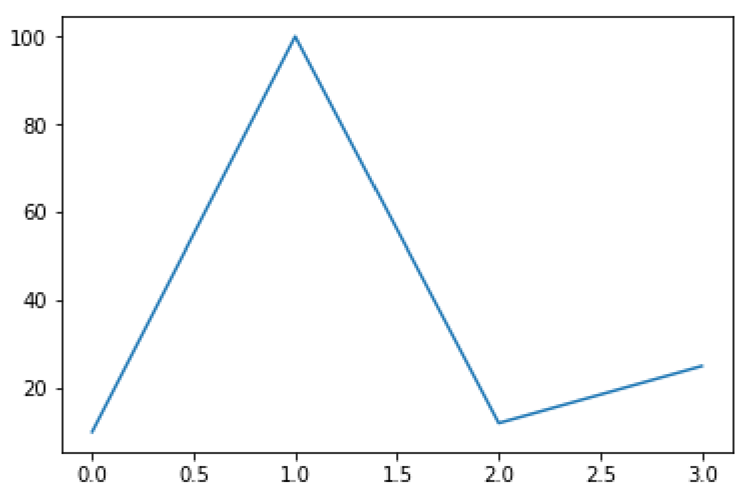
In [14]:
```
plt.plot(x,y,'go--',linewidth=20,markersize=30)
```

Out[14]: [<matplotlib.lines.Line2D at 0x26a1634c0b8>]



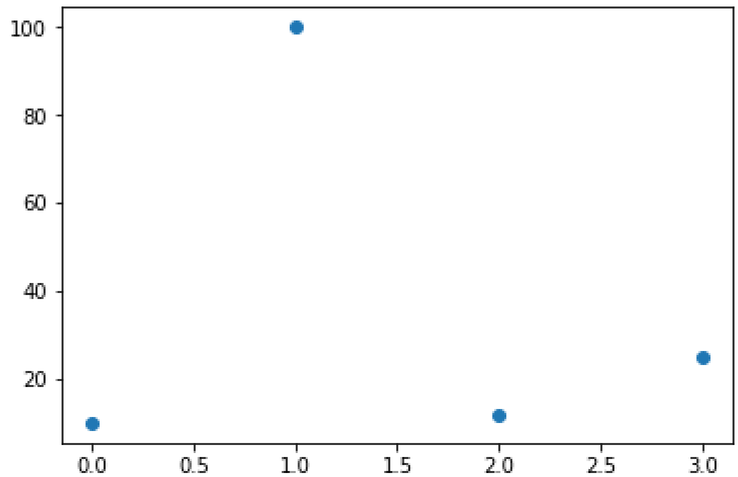
In [15]:
```
plt.plot([10,100,12,25])  #considering this values as y, indices as x values
```

Out[15]: [<matplotlib.lines.Line2D at 0x26a163ab400>]



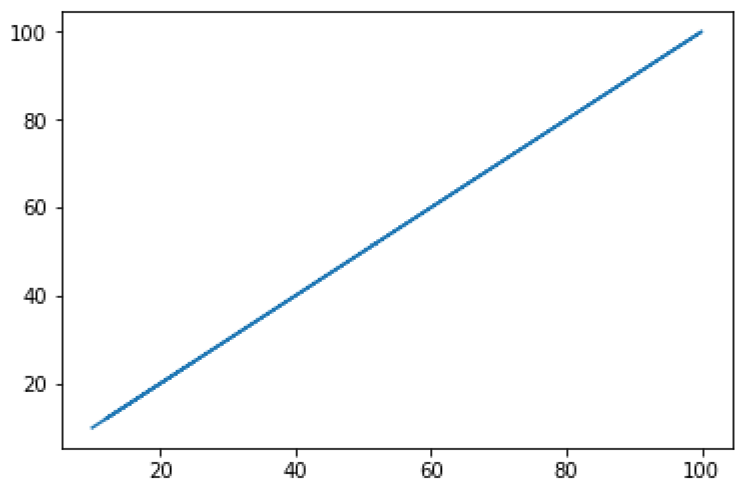
In [16]:
```
plt.plot([10,100,12,25],'o')
```

Out[16]: [<matplotlib.lines.Line2D at 0x26a16405320>]



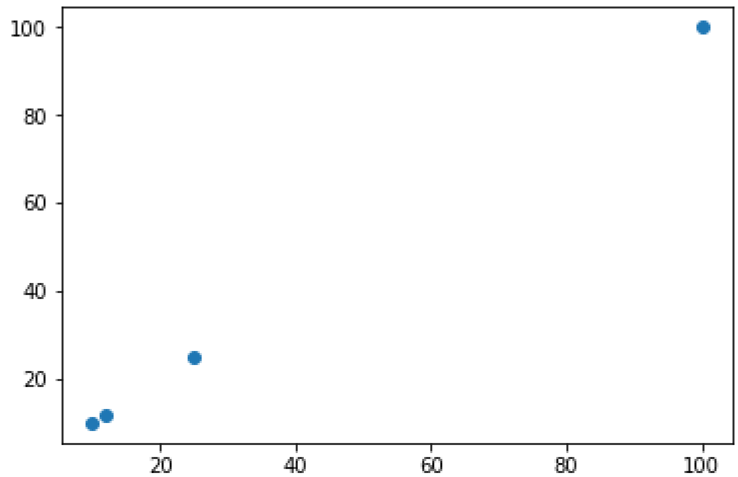
In [17]:
```
plt.plot([10,100,12,25],[10,100,12,25])
```

Out[17]: [<matplotlib.lines.Line2D at 0x26a16461470>]



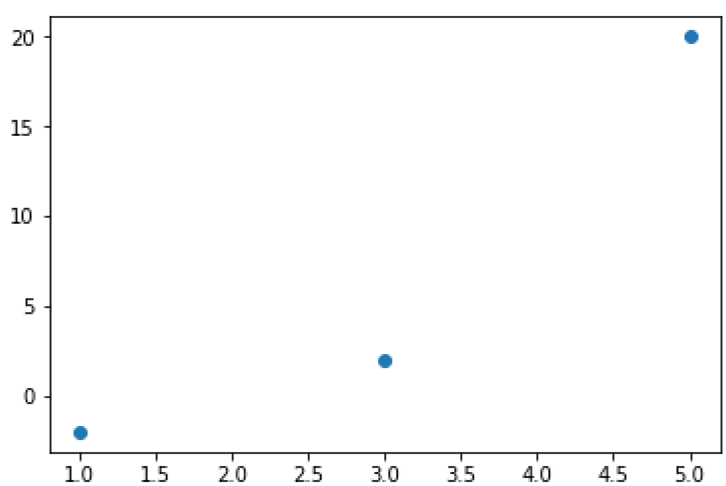
In [18]:
```
plt.plot([10,100,12,25],[10,100,12,25],'o')
```

Out[18]: [<matplotlib.lines.Line2D at 0x26a164b4b38>]



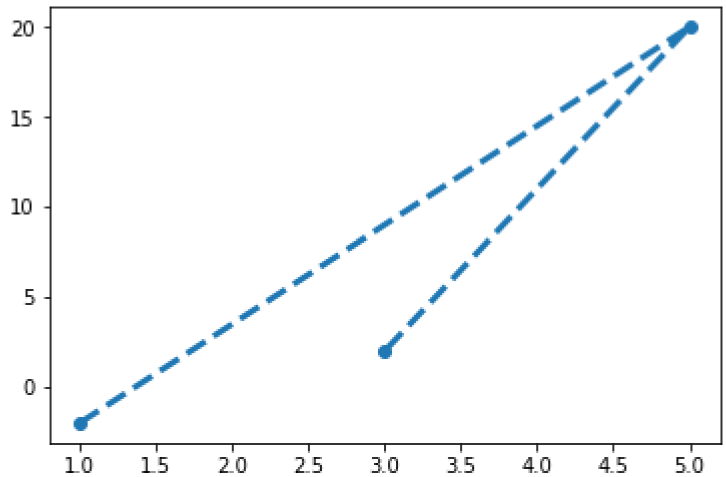
In [19]:
```
plt.plot([1,5,3],[-2,20,2],'o')
```

Out[19]: [<matplotlib.lines.Line2D at 0x26a1650d1d0>]



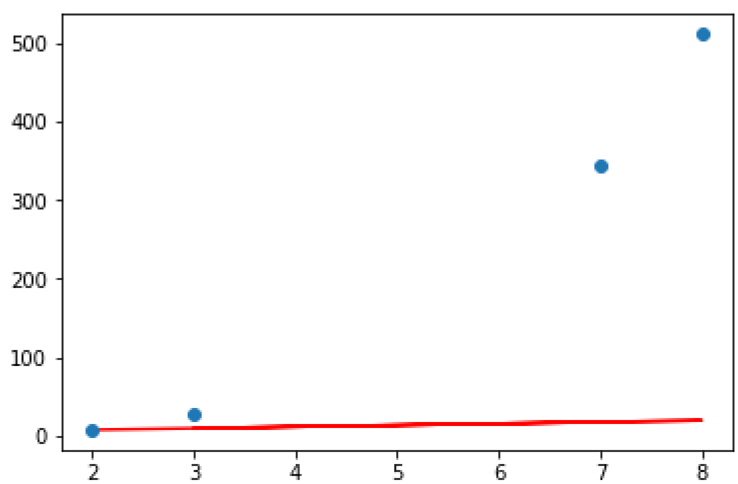
In [20]:
```
plt.plot([1,5,3],[-2,20,2],'o--',linewidth=3)  #graph is drawn based on given ord
```

Out[20]: [<matplotlib.lines.Line2D at 0x26a1656acc0>]



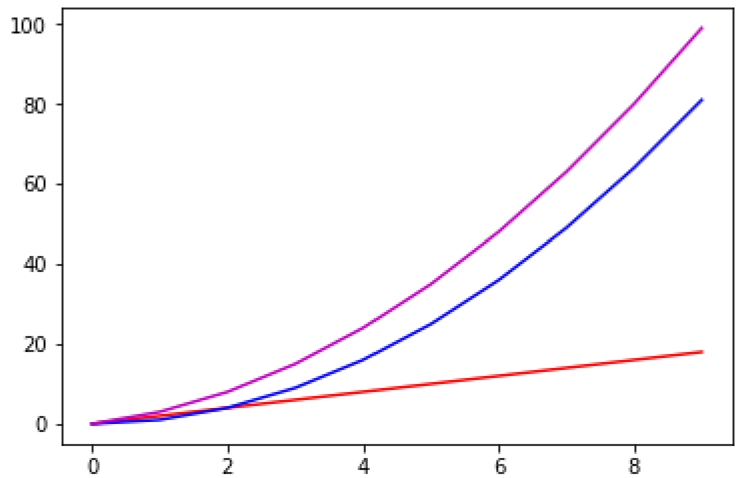
In [21]:
```
plt.plot(x,y,'r')
plt.plot(x,x**3,'o')
```

Out[21]: [<matplotlib.lines.Line2D at 0x26a165a8748>]



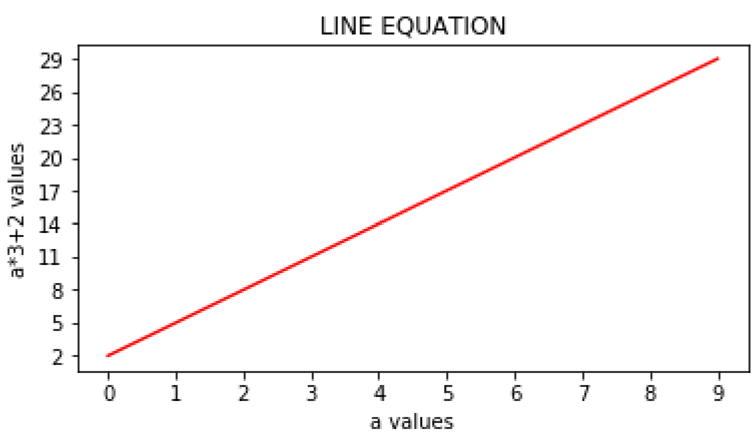
In [22]:
```
a=np.arange(10)
plt.plot(a,a*2,'r')
plt.plot(a,a**2,'b')
plt.plot(a,a*a+2*a,'m')
```

Out[22]:
```
[<matplotlib.lines.Line2D at 0x26a166325c0>]
```



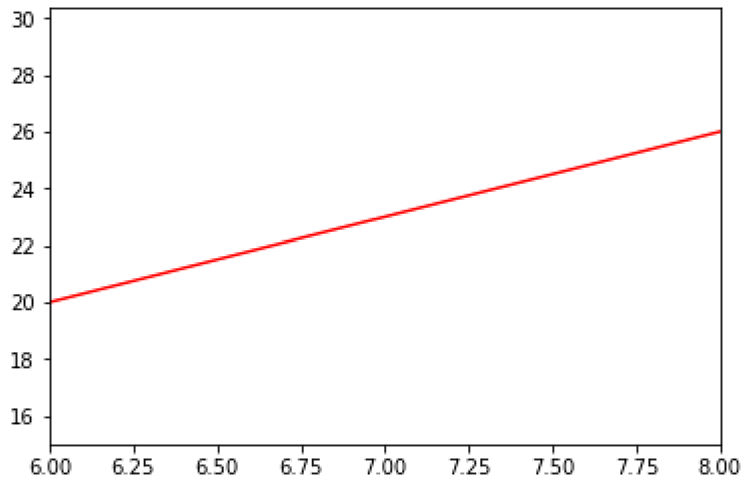
In [23]:
```
plt.figure(figsize=(6,3))  #size of graph  #figsize-->width,height
plt.plot(a,a*3+2,'r')
plt.title('LINE EQUATION')  #Title of graph
plt.xlabel('a values')      # X axis label
plt.ylabel('a*3+2 values')  # y axis label
plt.xticks(a)               #to take values on x-axis,a values 0-9 will take on x
plt.yticks(a*3+2)           #to take values on y-axis
```

Out[23]:
```
([<matplotlib.axis.YTick at 0x26a1626f898>,
  <matplotlib.axis.YTick at 0x26a1626f358>,
  <matplotlib.axis.YTick at 0x26a166568d0>,
  <matplotlib.axis.YTick at 0x26a1669d7f0>,
  <matplotlib.axis.YTick at 0x26a16696b38>,
  <matplotlib.axis.YTick at 0x26a166a6550>,
  <matplotlib.axis.YTick at 0x26a166a6b00>,
  <matplotlib.axis.YTick at 0x26a166af0b8>,
  <matplotlib.axis.YTick at 0x26a166af550>,
  <matplotlib.axis.YTick at 0x26a166afa58>],
 <a list of 10 Text yticklabel objects>)
```



In [24]:
```
plt.plot(a,a*3+2,'r')
plt.xlim(6,8)   #to see graph with in specific limit
plt.ylim(15)
```

Out[24]: (15, 30.35)

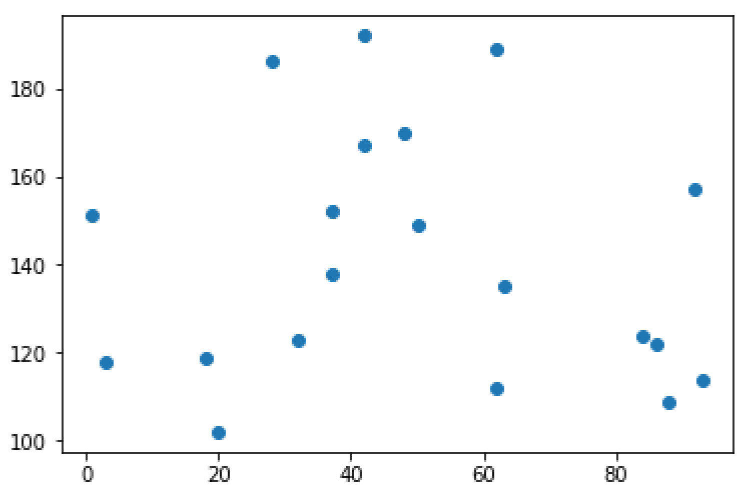


Scatterplot (circle representation) (For Bivariant)

- a graph in which the values of two variables are plotted along two axes, the pattern of the resulting points revealing any correlation present.
- x axis - feature 1 -c
- y axis - feature 2 -d

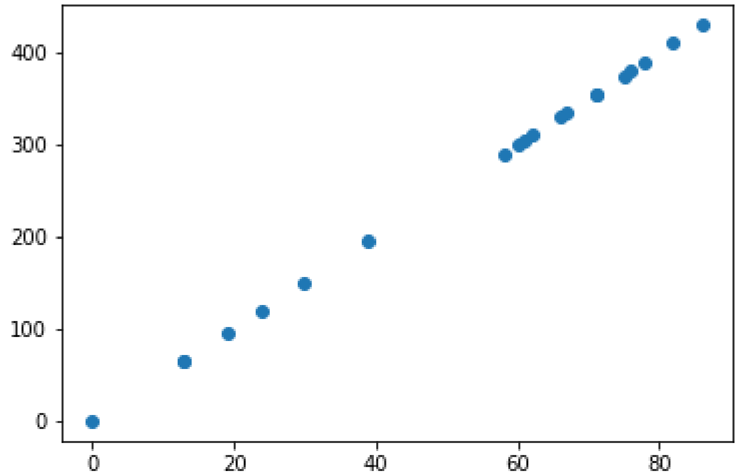
In [25]:
```
c=np.random.randint(100,size=20)
d=np.random.randint(100,200,(20))
plt.scatter(c,d)                    #not correlated, randomly distributed
```

Out[25]: <matplotlib.collections.PathCollection at 0x26a1674ad30>



In [26]:
```
c=np.random.randint(100,size=20)
d1=c*5                              #correlated , d1 depends on c
plt.scatter(c,d1)
```

Out[26]: <matplotlib.collections.PathCollection at 0x26a167a49e8>



Histogram (univariant - numerical data,not possible for categorical data)

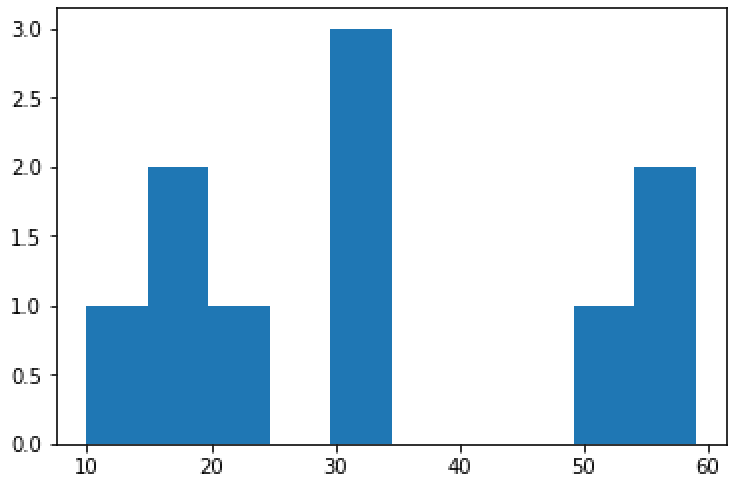
- x axis - bins (bin values are counted based on min,max values
- y axis- count of variable

In [27]:
```
np.linspace(10,59,11) # 11 parts, 10 bins  ,min,max range
```

Out[27]:
```
array([10. , 14.9, 19.8, 24.7, 29.6, 34.5, 39.4, 44.3, 49.2, 54.1, 59. ])
```

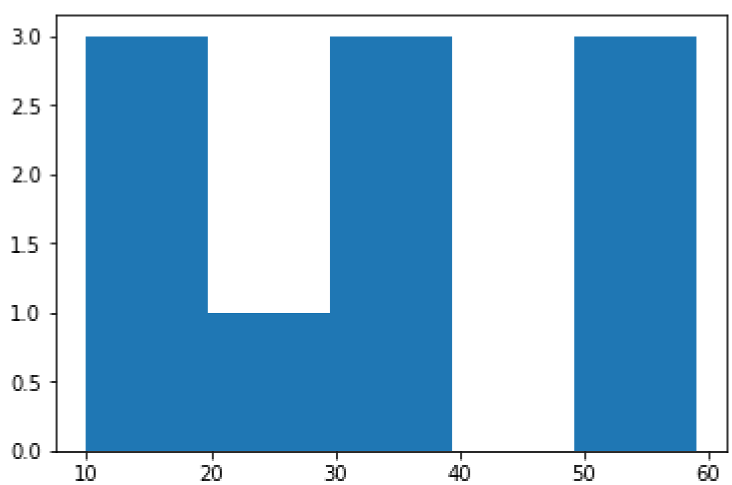
In [28]:
```
marks=[10,15,20,15,30,30,56,52,32,59]
plt.hist(marks)
```

Out[28]:
```
(array([1., 2., 1., 0., 3., 0., 0., 0., 1., 2.]),
 array([10. , 14.9, 19.8, 24.7, 29.6, 34.5, 39.4, 44.3, 49.2, 54.1, 59. ]),
 <a list of 10 Patch objects>)
```



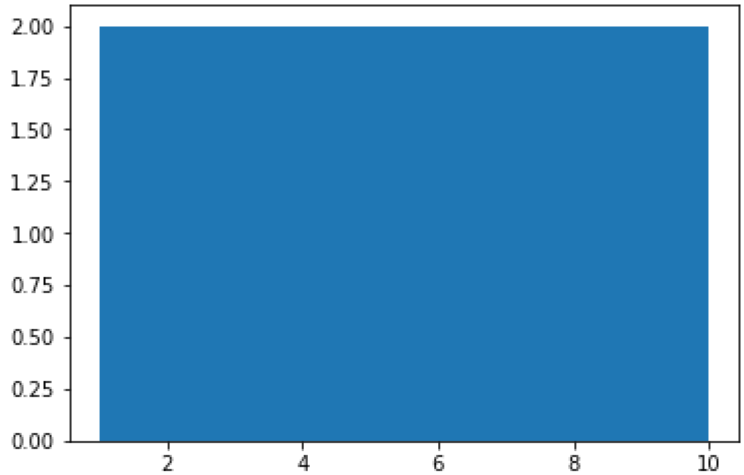
In [29]:
```
plt.hist(marks,bins=5)
```

Out[29]:
```
(array([3., 1., 3., 0., 3.]),
 array([10. , 19.8, 29.6, 39.4, 49.2, 59. ]),
 <a list of 5 Patch objects>)
```



In [30]:
```
plt.hist(range(1,11),bins=5)  #uniform distribution, histogram  5 bins, 6 points
```

Out[30]:
```
(array([2., 2., 2., 2., 2.]),
 array([ 1. ,  2.8,  4.6,  6.4,  8.2, 10. ]),
 <a list of 5 Patch objects>)
```

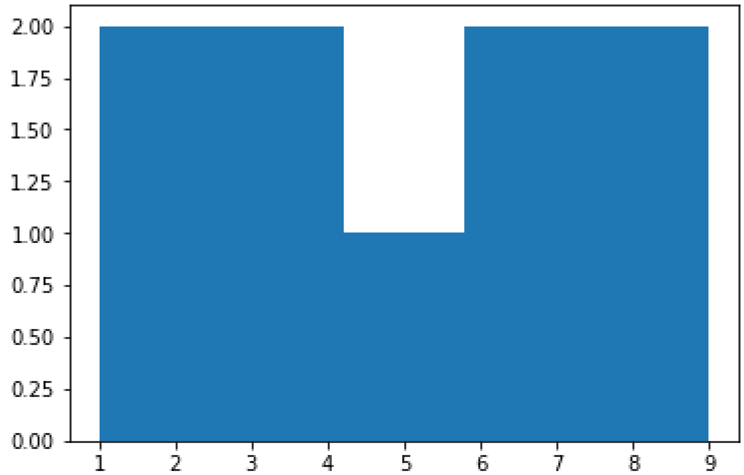


In [31]:
```
np.linspace(1,10,6)
```

Out[31]:
```
array([ 1. ,  2.8,  4.6,  6.4,  8.2, 10. ])
```

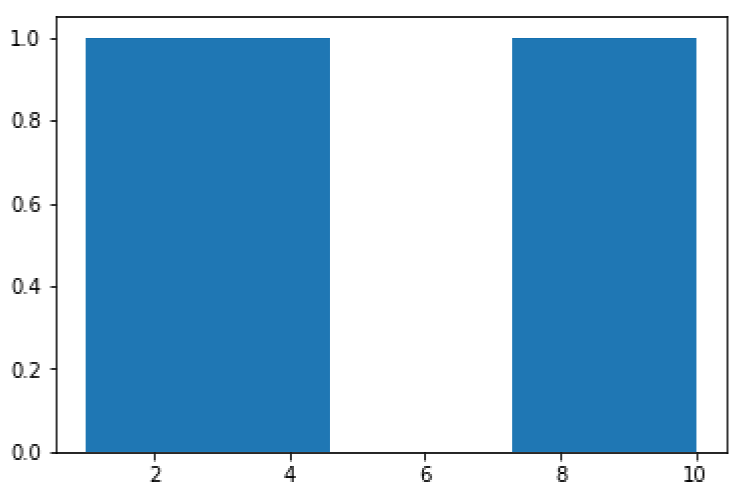
In [32]:
```
plt.hist(range(1,10),bins=5)
```

Out[32]:
```
(array([2., 2., 1., 2., 2.]),
 array([1. , 2.6, 4.2, 5.8, 7.4, 9. ]),
 <a list of 5 Patch objects>)
```



In [33]:
```
plt.hist([1,2,3,4,8,9,10])
```

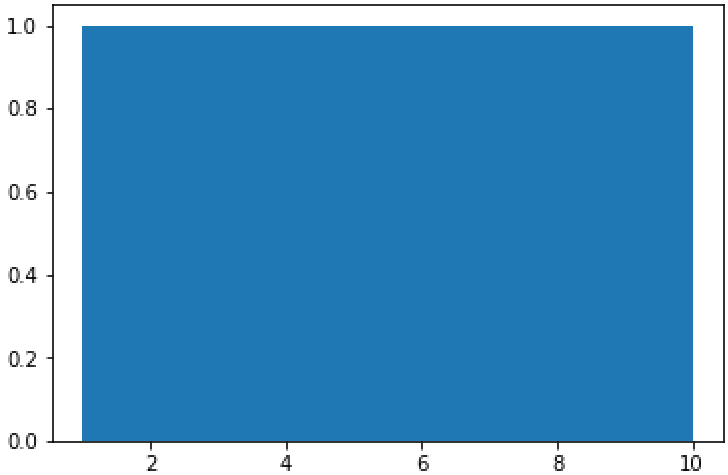
Out[33]:
```
(array([1., 1., 1., 1., 0., 0., 0., 1., 1., 1.]),
 array([ 1. ,  1.9,  2.8,  3.7,  4.6,  5.5,  6.4,  7.3,  8.2,  9.1, 10. ]),
 <a list of 10 Patch objects>)
```



Uniform distribution

In [34]:
```
plt.hist(range(1,11))
```

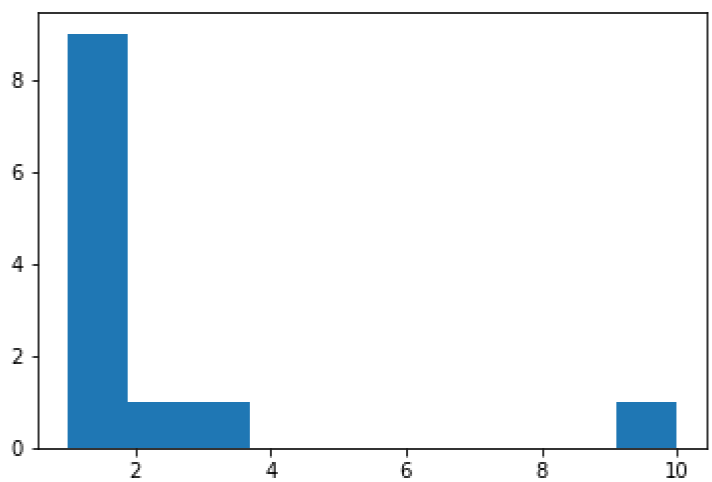
Out[34]:
```
(array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.]),
 array([ 1. ,  1.9,  2.8,  3.7,  4.6,  5.5,  6.4,  7.3,  8.2,  9.1, 10. ]),
 <a list of 10 Patch objects>)
```



positively skewed data

In [35]:
```
plt.hist([1,1,1,1,1,1,1,1,1,2,3,10])
```

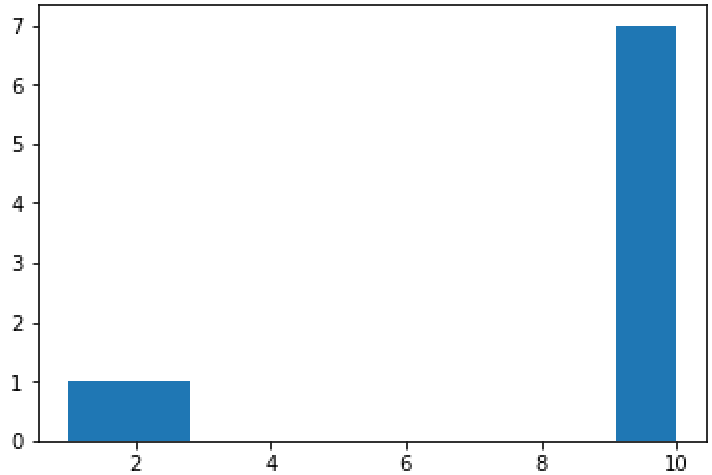
Out[35]:
```
(array([9., 1., 1., 0., 0., 0., 0., 0., 0., 1.]),
 array([ 1. ,  1.9,  2.8,  3.7,  4.6,  5.5,  6.4,  7.3,  8.2,  9.1, 10. ]),
 <a list of 10 Patch objects>)
```



negatively skewed data

In [36]:
```
plt.hist([10,10,10,10,10,10,10,2,1])
```

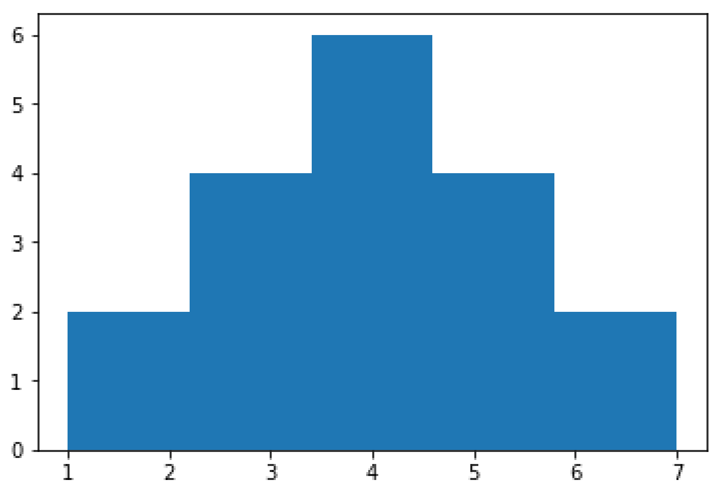
Out[36]:
```
(array([1., 1., 0., 0., 0., 0., 0., 0., 0., 7.]),
 array([ 1. ,  1.9,  2.8,  3.7,  4.6,  5.5,  6.4,  7.3,  8.2,  9.1, 10. ]),
 <a list of 10 Patch objects>)
```



Normal distribution

In [37]:
```
plt.hist([1,2,3,3,3,3,4,4,4,4,4,4,5,5,5,5,6,7],bins=5)
```

Out[37]:
```
(array([2., 4., 6., 4., 2.]),
 array([1. , 2.2, 3.4, 4.6, 5.8, 7. ]),
 <a list of 5 Patch objects>)
```



Bargraph(Univariate- categorical)

In [38]:
```
colleges=pd.Series(['IIIT','SVCE','IIIT','SVCE','RVRJC','IIIT','VRSEC'])
colleges
```

Out[38]:
```
0     IIIT
1     SVCE
2     IIIT
3     SVCE
4    RVRJC
5     IIIT
6    VRSEC
dtype: object
```

In [39]:
```
colleges.nunique()
```

Out[39]:
```
4
```

In [40]:
```
colleges.value_counts()
```

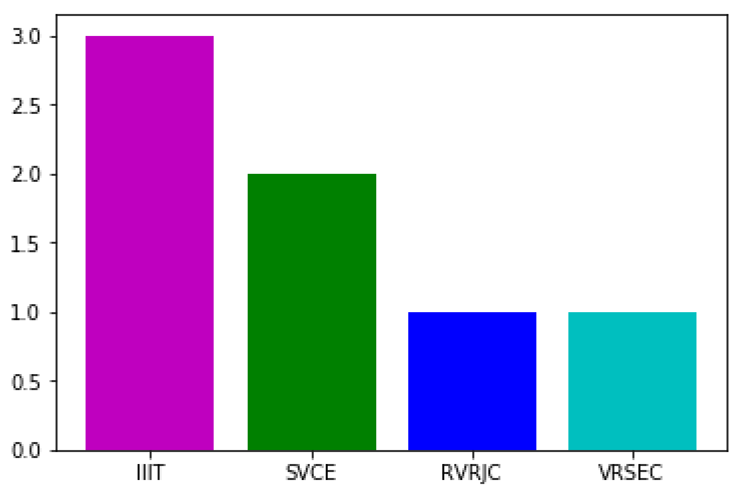
Out[40]:
```
IIIT     3
SVCE     2
RVRJC    1
VRSEC    1
dtype: int64
```

In [41]:
```
names=colleges.value_counts().index
names
```

Out[41]:
```
Index(['IIIT', 'SVCE', 'RVRJC', 'VRSEC'], dtype='object')
```

In [42]:
```
plt.bar(names,colleges.value_counts(),color='mgbc')
```

Out[42]:
```
<BarContainer object of 4 artists>
```



Seaborn

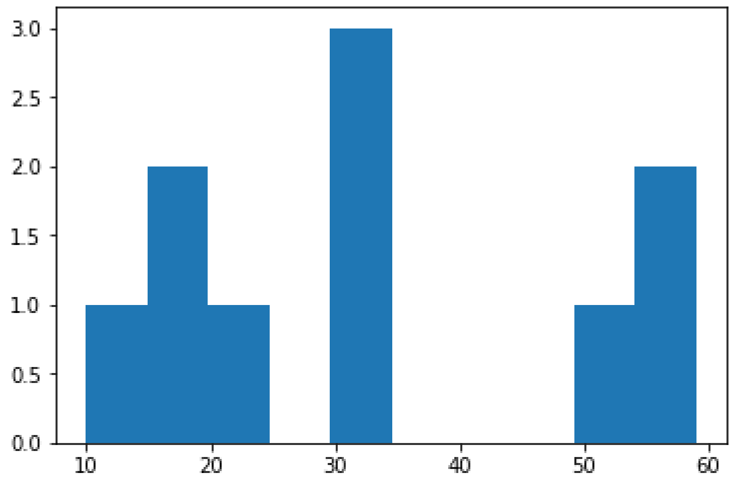
- Distplot(The distplot figure factory displays a combination of statistical representations of numerical data, such as histogram, kernel density estimation or normal curve, and rug plot.)
- y axis-proportion

In [43]:
```
marks
```

Out[43]:
```
[10, 15, 20, 15, 30, 30, 56, 52, 32, 59]
```

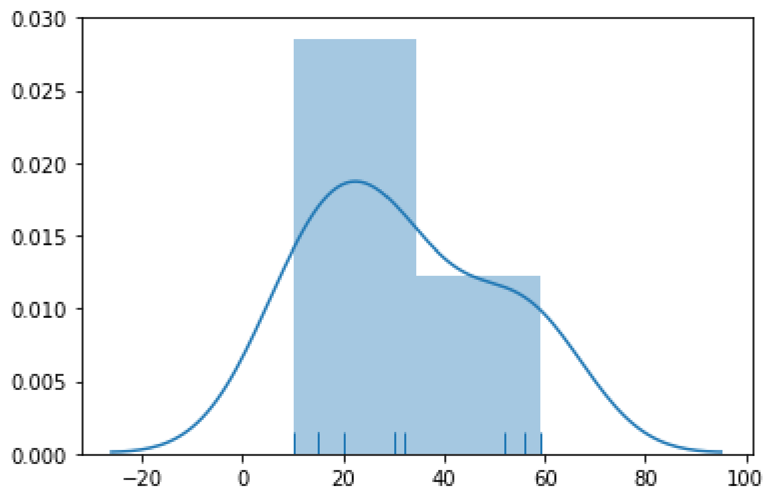
In [44]:
```
plt.hist(marks)
```

Out[44]:
```
(array([1., 2., 1., 0., 3., 0., 0., 0., 1., 2.]),
 array([10. , 14.9, 19.8, 24.7, 29.6, 34.5, 39.4, 44.3, 49.2, 54.1, 59. ]),
 <a list of 10 Patch objects>)
```



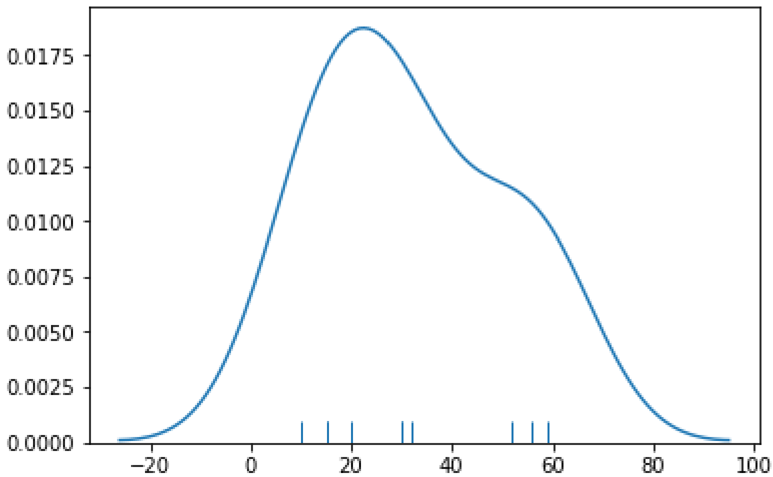
In [45]:
```
sns.distplot(marks,rug=True)
```

Out[45]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x26a17bea390>
```



In [46]:
```
sns.distplot(marks,hist=False,rug=True)   #rug=True, for datasets values indicat
```

Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x26a17c54ef0>

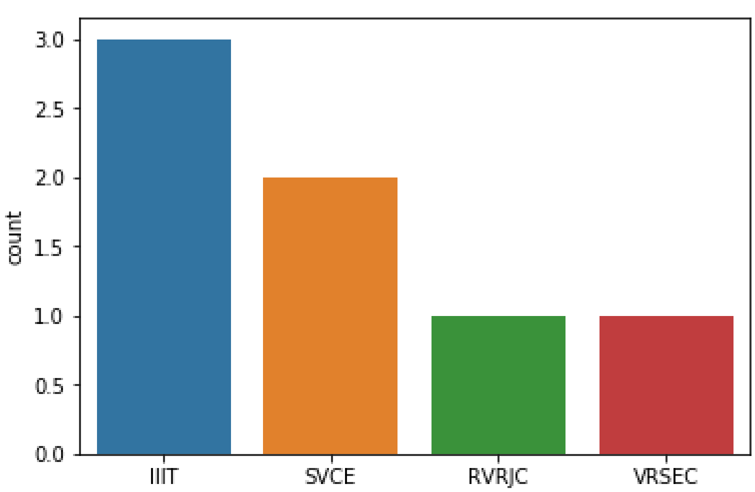


Countplot

- Categorical columns

In [47]:
```
sns.countplot(colleges)
```

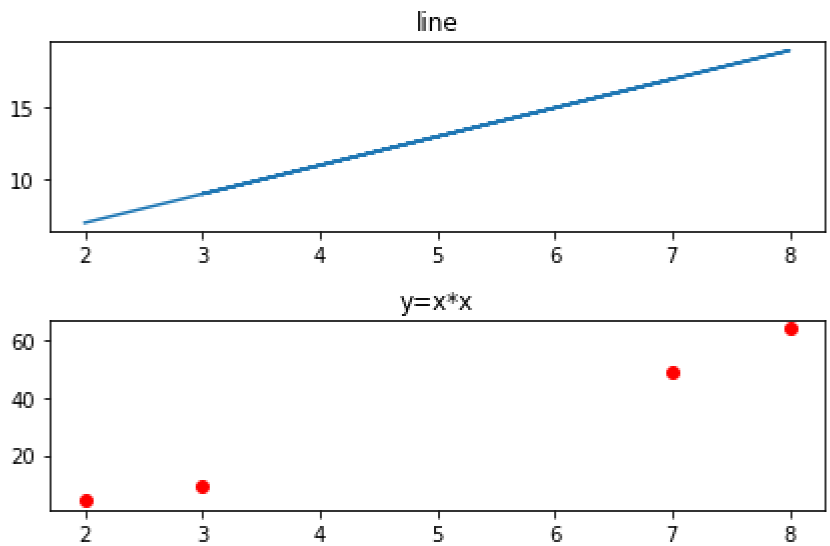
Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x26a17be24e0>



Subplots

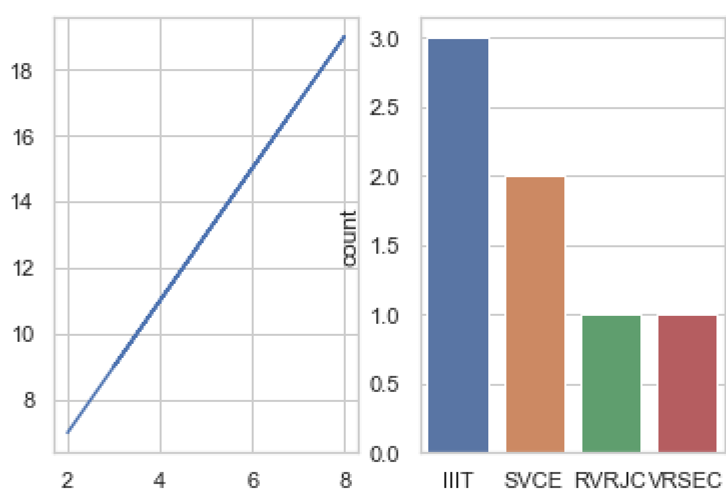
- only plt functions
- combination of sns and plt

In [48]:
```
plt.subplot(211)
plt.plot(x,y)
plt.title('line')
plt.subplot(212)
plt.plot(x,x*x,'ro')
plt.title('y=x*x')
plt.tight_layout()
```



In [82]:
```
plt.subplot(1,2,1)
plt.plot(x,y)
plt.subplot(1,2,2)
sns.countplot(colleges)
```

Out[82]: <matplotlib.axes._subplots.AxesSubplot at 0x26a24414e48>



In [50]:
```
np.median([10,20,60,40,50])
```
Out[50]:
```
40.0
```

In [51]:
```
np.median([10,20,500,40,50])
```
Out[51]:
```
40.0
```

In [52]:
```
l=[10,20,60,40,50]
l1=[10,20,500,40,50]
print(np.mean(l),np.mean(l1))
```
```
36.0 124.0
```

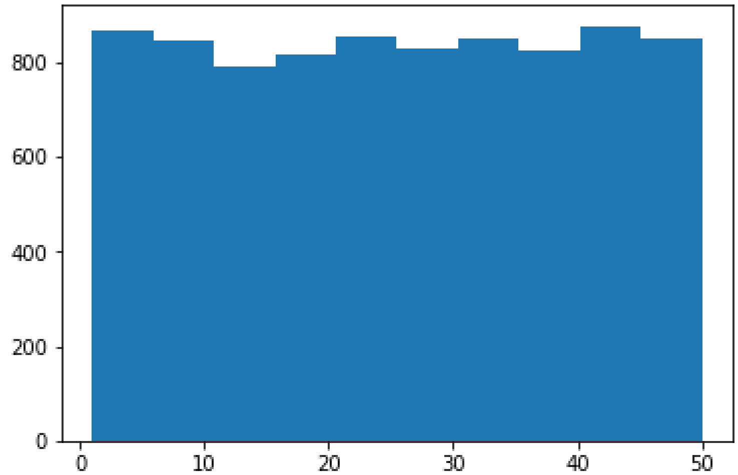
Boxplot (to findout outliers)

- Observe Histograms for numerical columns
- Bar Plots for categorical columns
- Scatter plots between numerical columns

In [53]:
```
df=pd.read_csv("market_fact.csv")
df.dtypes
```
Out[53]:
```
Ord_id                  object
Prod_id                 object
Ship_id                 object
Cust_id                 object
Sales                  float64
Discount               float64
Order_Quantity           int64
Profit                 float64
Shipping_Cost          float64
Product_Base_Margin    float64
dtype: object
```

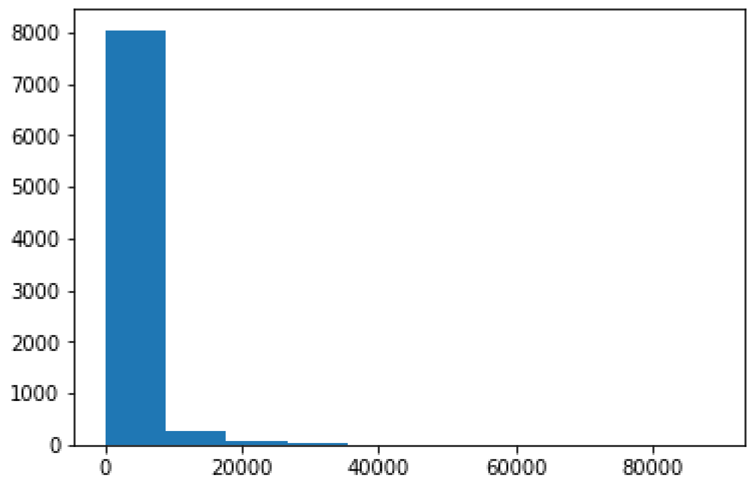
In [54]:
```
plt.hist(df['Order_Quantity'])
```

Out[54]:
```
(array([868., 847., 789., 816., 854., 827., 848., 825., 876., 849.]),
 array([ 1. ,  5.9, 10.8, 15.7, 20.6, 25.5, 30.4, 35.3, 40.2, 45.1, 50. ]),
 <a list of 10 Patch objects>)
```



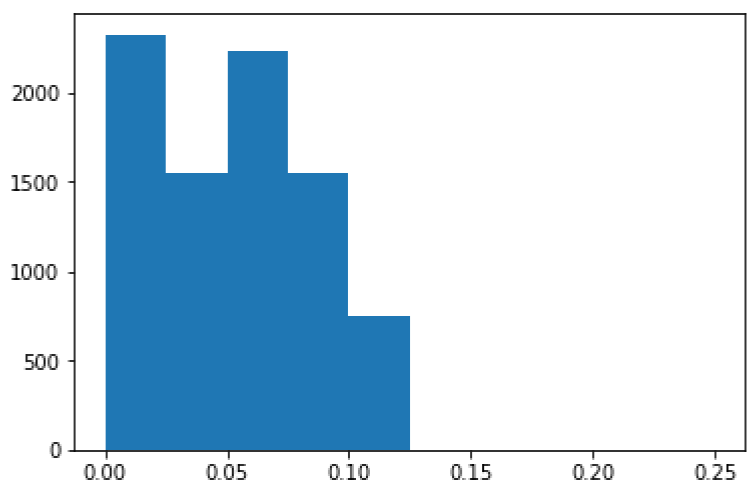
In [55]:
```
plt.hist(df['Sales'])
```

Out[55]:
```
(array([8.047e+03, 2.580e+02, 7.800e+01, 1.300e+01, 1.000e+00, 1.000e+00,
        0.000e+00, 0.000e+00, 0.000e+00, 1.000e+00]),
 array([2.2400000e+00, 8.9081210e+03, 1.7814002e+04, 2.6719883e+04,
        3.5625764e+04, 4.4531645e+04, 5.3437526e+04, 6.2343407e+04,
        7.1249288e+04, 8.0155169e+04, 8.9061050e+04]),
 <a list of 10 Patch objects>)
```



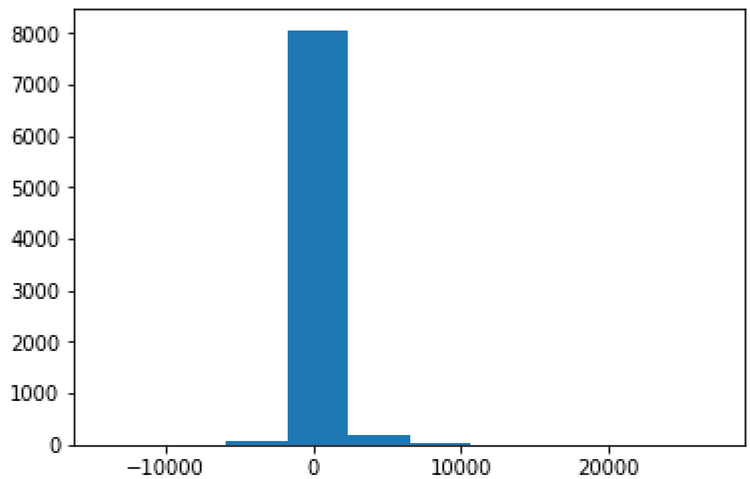
In [56]:
```
plt.hist(df['Discount'])
```

Out[56]:
```
(array([2.327e+03, 1.549e+03, 2.230e+03, 1.543e+03, 7.460e+02, 0.000e+00,
        2.000e+00, 0.000e+00, 1.000e+00, 1.000e+00]),
 array([0.   , 0.025, 0.05 , 0.075, 0.1  , 0.125, 0.15 , 0.175, 0.2  ,
        0.225, 0.25 ]),
 <a list of 10 Patch objects>)
```



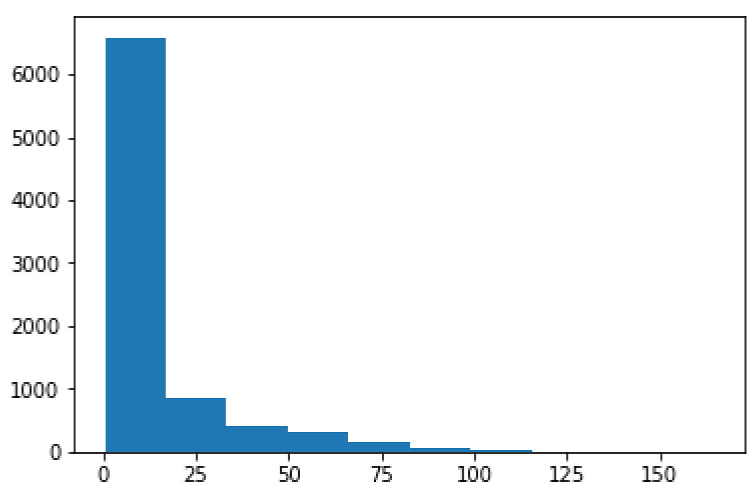
In [57]:
```
plt.hist(df['Profit'])
```

Out[57]:
```
(array([7.000e+00, 9.000e+00, 7.300e+01, 8.066e+03, 1.880e+02, 4.600e+01,
        9.000e+00, 0.000e+00, 0.000e+00, 1.000e+00]),
 array([-14140.7  , -10004.561,  -5868.422,  -1732.283,   2403.856,
          6539.995,  10676.134,  14812.273,  18948.412,  23084.551,
         27220.69 ]),
 <a list of 10 Patch objects>)
```



In [58]:
```
plt.hist(df['Shipping_Cost'])
```

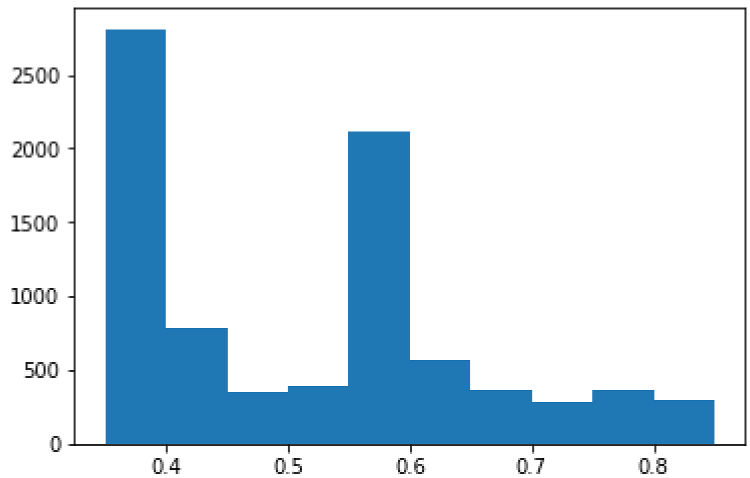
Out[58]:
```
(array([6.583e+03, 8.600e+02, 4.180e+02, 3.260e+02, 1.530e+02, 4.200e+01,
        1.000e+01, 2.000e+00, 3.000e+00, 2.000e+00]),
 array([  0.49 ,  16.914,  33.338,  49.762,  66.186,  82.61 ,  99.034,
        115.458, 131.882, 148.306, 164.73 ]),
 <a list of 10 Patch objects>)
```



In [59]:
```
plt.hist(df['Product_Base_Margin'])
```

```
C:\Users\APSSDC\Anaconda3\lib\site-packages\numpy\lib\histograms.py:824: Runtim
eWarning: invalid value encountered in greater_equal
  keep = (tmp_a >= first_edge)
C:\Users\APSSDC\Anaconda3\lib\site-packages\numpy\lib\histograms.py:825: Runtim
eWarning: invalid value encountered in less_equal
  keep &= (tmp_a <= last_edge)
```

Out[59]:
```
(array([2811.,  779.,  348.,  396., 2116.,  574.,  365.,  285.,  365.,
         297.]),
 array([0.35, 0.4 , 0.45, 0.5 , 0.55, 0.6 , 0.65, 0.7 , 0.75, 0.8 , 0.85]),
 <a list of 10 Patch objects>)
```



In [60]:
```
i=df['Ord_id'].value_counts().index
```

In [61]:
```
df['Ord_id']
```

Out[61]:
```
0        Ord_5446
1        Ord_5406
2        Ord_5446
3        Ord_5456
4        Ord_5485
5        Ord_5446
6          Ord_31
7        Ord_4725
8        Ord_4725
9        Ord_4725
10       Ord_4743
11       Ord_1925
12       Ord_2978
13       Ord_2207
14       Ord_2207
15       Ord_2280
16       Ord_2282
17       Ord_4471
18       Ord_4427
19        Ord_996
20        Ord_996
21        Ord_996
22        Ord_996
23        Ord_996
24       Ord_2573
25       Ord_2335
26       Ord_2456
27       Ord_2405
28       Ord_2573
29       Ord_2478
           ...
8369     Ord_3633
8370     Ord_2696
8371     Ord_2624
8372     Ord_2772
8373     Ord_2600
8374     Ord_2658
8375     Ord_2772
8376     Ord_2624
8377     Ord_2722
8378     Ord_2706
8379     Ord_2722
8380     Ord_2772
8381     Ord_2696
8382     Ord_2658
8383     Ord_2722
8384     Ord_4620
8385     Ord_1833
8386     Ord_2324
8387     Ord_2220
8388     Ord_4424
8389     Ord_4444
8390     Ord_5435
8391     Ord_5435
8392     Ord_5384
```

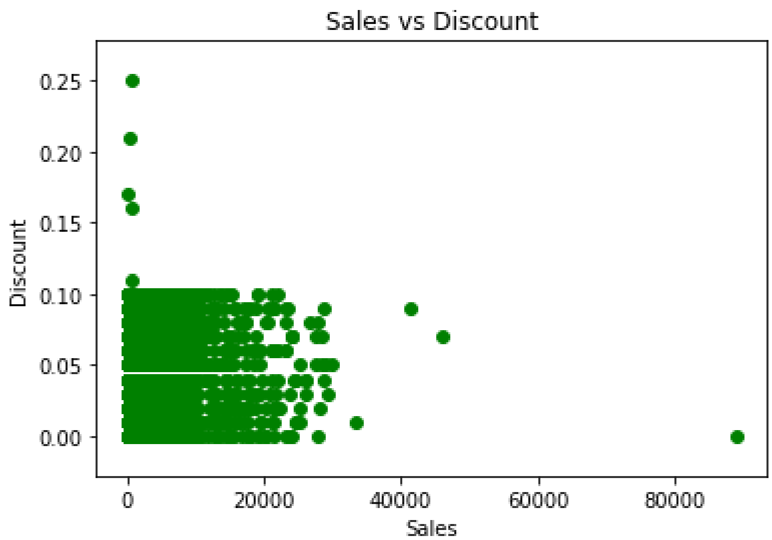
```
8393    Ord_5348
8394    Ord_5353
8395    Ord_5411
8396    Ord_5388
8397    Ord_5348
8398    Ord_5459
Name: Ord_id, Length: 8399, dtype: object
```

In [62]:
```
plt.bar(i,df['Ord_id'].value_counts())
```

...

In [63]:
```
plt.scatter(df['Sales'],df['Discount'],color='g')
plt.xlabel('Sales')
plt.ylabel('Discount')
plt.title('Sales vs Discount')
```

Out[63]:
```
Text(0.5, 1.0, 'Sales vs Discount')
```

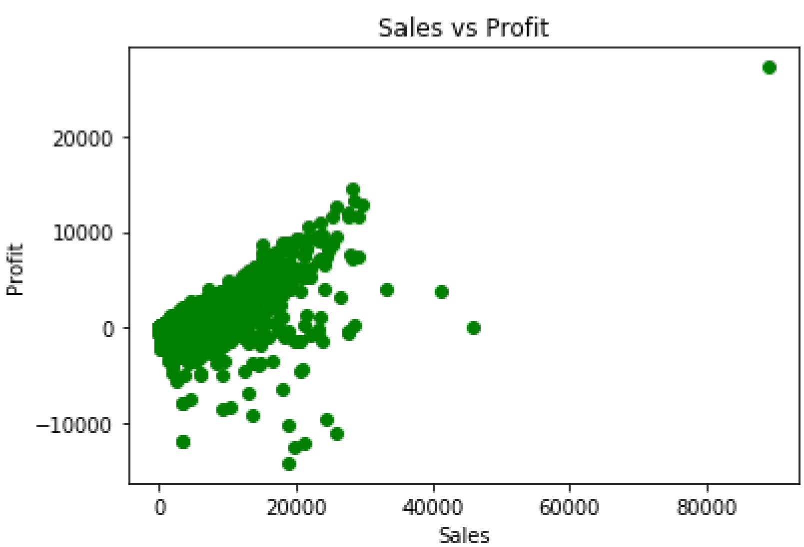


In [64]:
```
plt.scatter(df['Sales'],df['Order_Quantity'],color='g')
plt.xlabel('Sales')
plt.ylabel('Order_Quantity')
plt.title('Sales vs Order_Quantity')
```

Out[64]: Text(0.5, 1.0, 'Sales vs Order_Quantity')

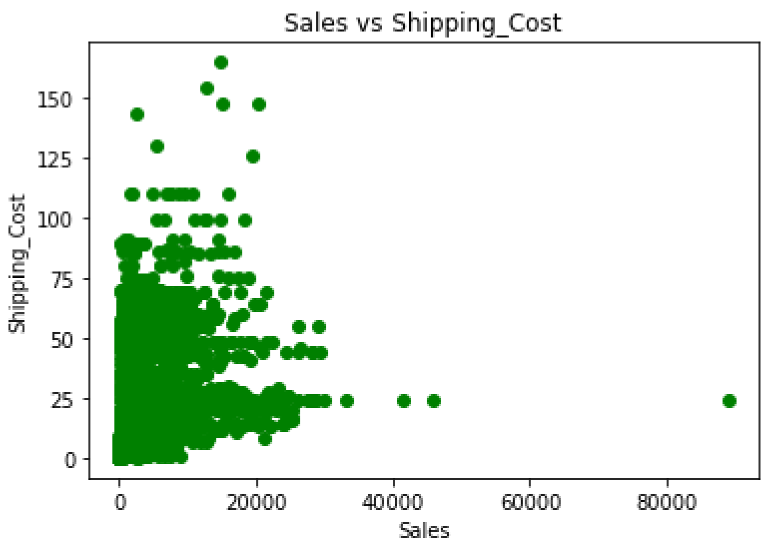
In [65]:
```
plt.scatter(df['Sales'],df['Profit'],color='g')
plt.xlabel('Sales')
plt.ylabel('Profit')
plt.title('Sales vs Profit')
```

Out[65]: Text(0.5, 1.0, 'Sales vs Profit')



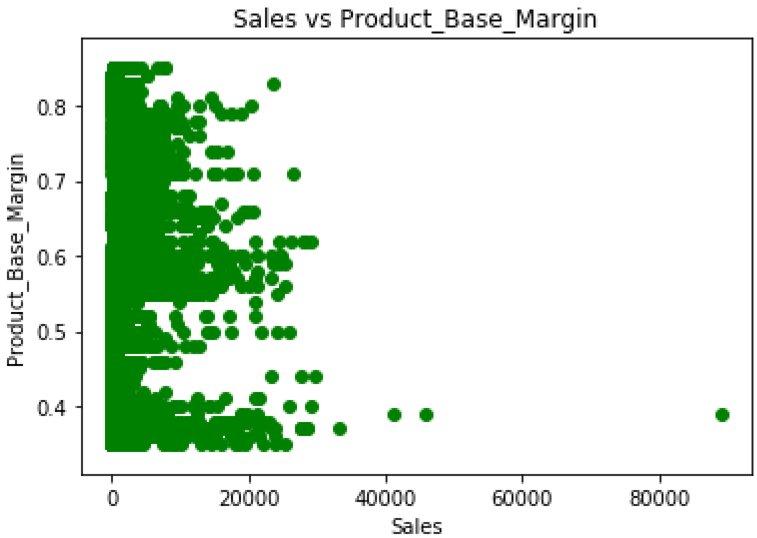
In [66]:
```
plt.scatter(df['Sales'],df['Shipping_Cost'],color='g')
plt.xlabel('Sales')
plt.ylabel('Shipping_Cost')
plt.title('Sales vs Shipping_Cost')
```

Out[66]: Text(0.5, 1.0, 'Sales vs Shipping_Cost')



In [67]:
```
plt.scatter(df['Sales'],df['Product_Base_Margin'],color='g')
plt.xlabel('Sales')
plt.ylabel('Product_Base_Margin')
plt.title('Sales vs Product_Base_Margin')
```

Out[67]: Text(0.5, 1.0, 'Sales vs Product_Base_Margin')

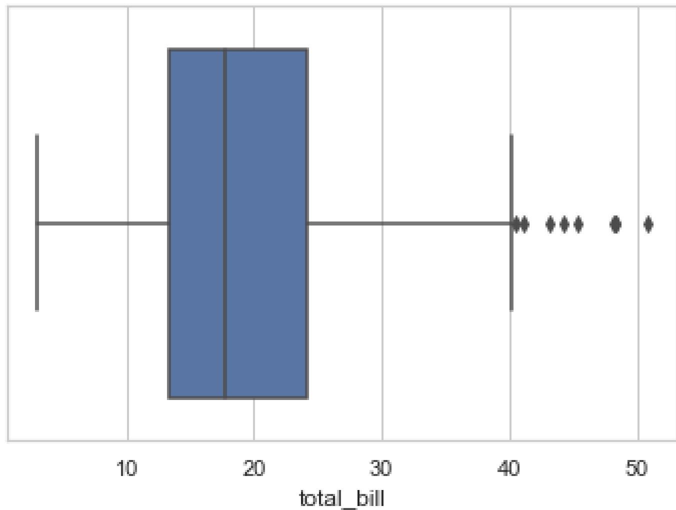


- check what is bocplot and observe the outliers in each numerical column using boxplot
- for boxplot calculations [https://www.mathsisfun.com/data/quartiles.html (https://www.mathsisfun.com/data/quartiles.html)](https://www.mathsisfun.com/data/quartiles.html)

In [68]:
```
help(sns.boxplot)
```

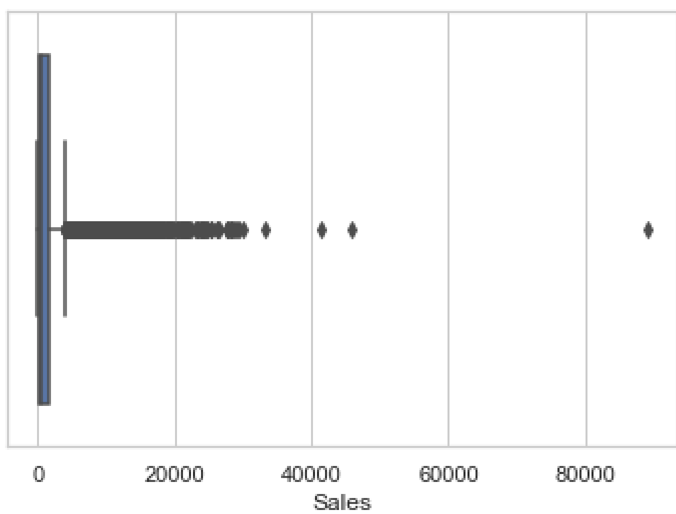
```
Help on function boxplot in module seaborn.categorical:

boxplot(x=None, y=None, hue=None, data=None, order=None, hue_order=None, orie
nt=None, color=None, palette=None, saturation=0.75, width=0.8, dodge=True, fl
iersize=5, linewidth=None, whis=1.5, notch=False, ax=None, **kwargs)
    Draw a box plot to show distributions with respect to categories.

    A box plot (or box-and-whisker plot) shows the distribution of quantitati
ve
    data in a way that facilitates comparisons between variables or across
    levels of a categorical variable. The box shows the quartiles of the
    dataset while the whiskers extend to show the rest of the distribution,
    except for points that are determined to be "outliers" using a method
    that is a function of the inter-quartile range.


    Input data can be passed in a variety of formats, including:

    - Vectors of data represented as lists, numpy arrays, or pandas Series
```

In [69]:
```
import seaborn as sns
sns.set(style="whitegrid")
tips = sns.load_dataset("tips")
ax = sns.boxplot(x=tips["total_bill"])
```



In [71]:
```
sns.boxplot(df['Sales'])
```

Out[71]: <matplotlib.axes._subplots.AxesSubplot at 0x26a2412ce80>



In [74]:
```
import numpy as np
q1=np.quantile(df['Sales'],0.25)
q2=np.quantile(df['Sales'],0.5)
q3=np.quantile(df['Sales'],0.75)
```

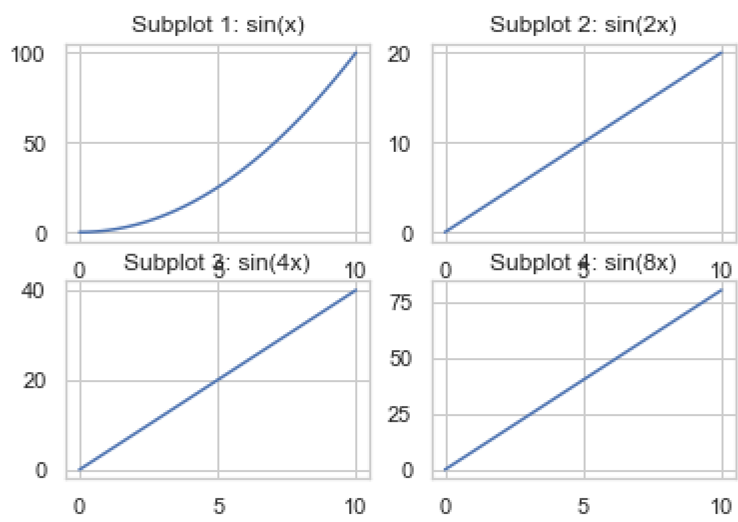
In [76]:
```
iqr=q3-q1
print(q1-iqr,q3+iqr)  #outside these range are outliers
```

```
-1422.9300000000003 3275.4450000000006
```

Subplot - to represent in grid format

In [91]:
```
import math
plt.subplot(2,2,1)
x = np.linspace(0,10);
y1 = x*x
plt.plot(x,y1)
plt.title('Subplot 1: sin(x)')

plt.subplot(2,2,2)
y2 = 2*x;
plt.plot(x,y2)
plt.title('Subplot 2: sin(2x)')

plt.subplot(2,2,3)
y3 = 4*x;
plt.plot(x,y3)
plt.title('Subplot 3: sin(4x)')

plt.subplot(2,2,4)
y4 = 8*x;
plt.plot(x,y4)
plt.title('Subplot 4: sin(8x)')
```

Out[91]:
```
Text(0.5, 1.0, 'Subplot 4: sin(8x)')
```



Google,Kaggle,UCI datasets

In [ ]: