Todays topics

- Matploptlib
- Seaborn -data visualitation modules
  - in the form of plots/graphs

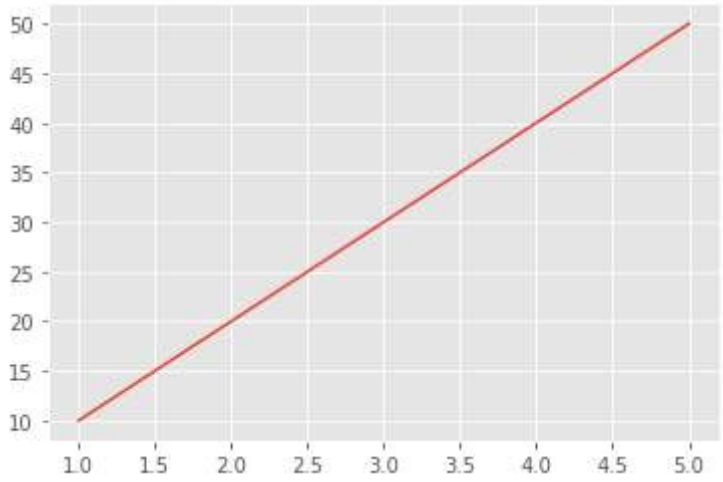
In [ ]:

In [20]:
```
import matplotlib.pyplot as plt
```

In [21]:
```
#ploting the graphs between list of variables
dir(plt)
```

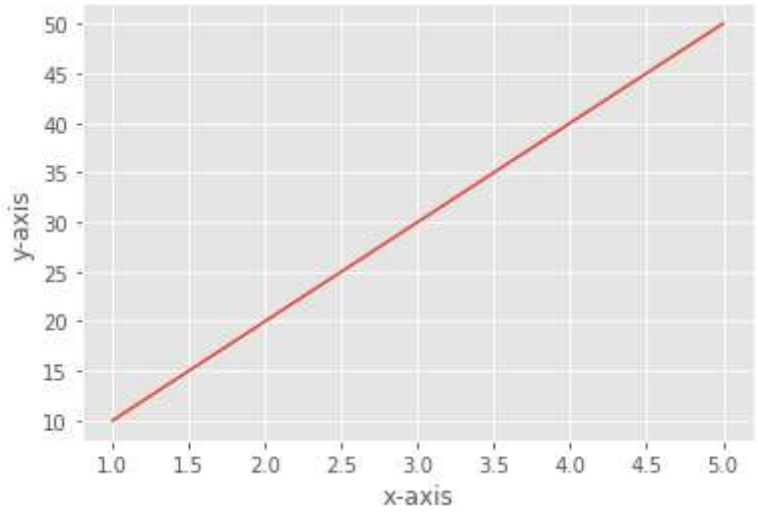
Out[21]:
```
['Annotation',
 'Arrow',
 'Artist',
 'AutoLocator',
 'Axes',
 'Button',
 'Circle',
 'Figure',
 'FigureCanvasBase',
 'FixedFormatter',
 'FixedLocator',
 'FormatStrFormatter',
 'Formatter',
 'FuncFormatter',
 'GridSpec',
 'IndexLocator',
 'Line2D',
 'LinearLocator',
 'Locator',
```

In [22]:
```
x=[1,2,3,4,5]
y=[10,20,30,40,50]
z=[34.5,67.8,56.7,34.5,23.6]
plt.plot(x,y)
plt.show()
```



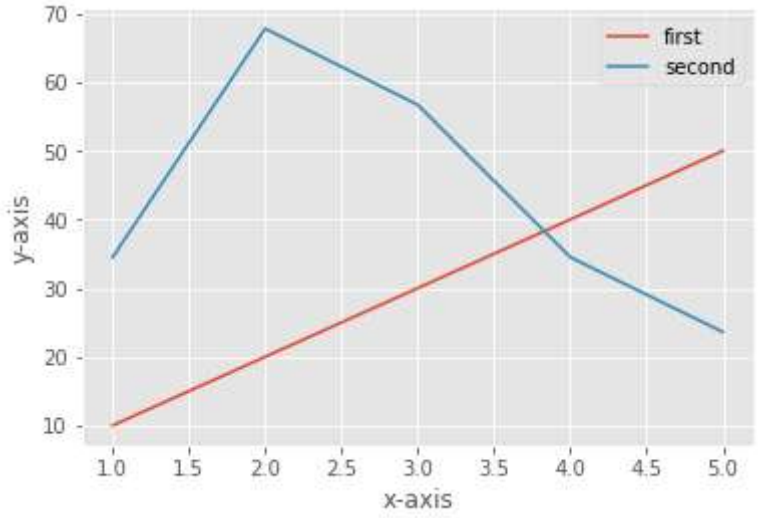
Sample Plot

In [23]:
```
x=[1,2,3,4,5]
y=[10,20,30,40,50]
z=[34.5,67.8,56.7,34.5,23.6]
plt.plot(x,y)
plt.xlabel('x-axis')
plt.ylabel('y-axis')                                    #plotting sample line graph b
plt.show()
```



In [ ]:

In [24]:
```
#adding x,y lables
plt.plot(x,y,label='first')
plt.plot(x,z,label='second')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.legend()
plt.show()
```

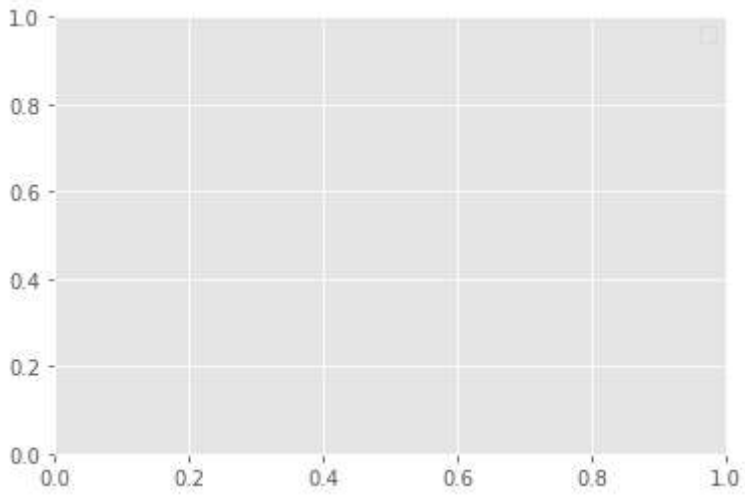


In [ ]:
```

```

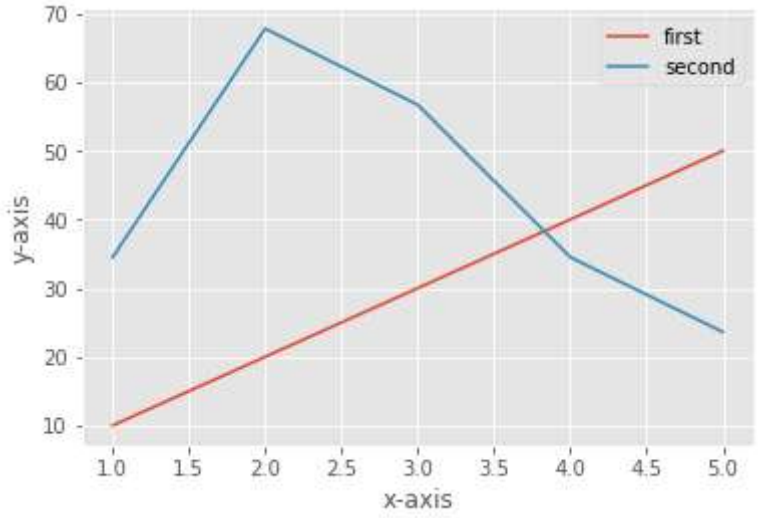
In [25]:
```
plt.legend()
```

```
No handles with labels found to put in legend.
```

Out[25]:
```
<matplotlib.legend.Legend at 0x2912b568340>
```

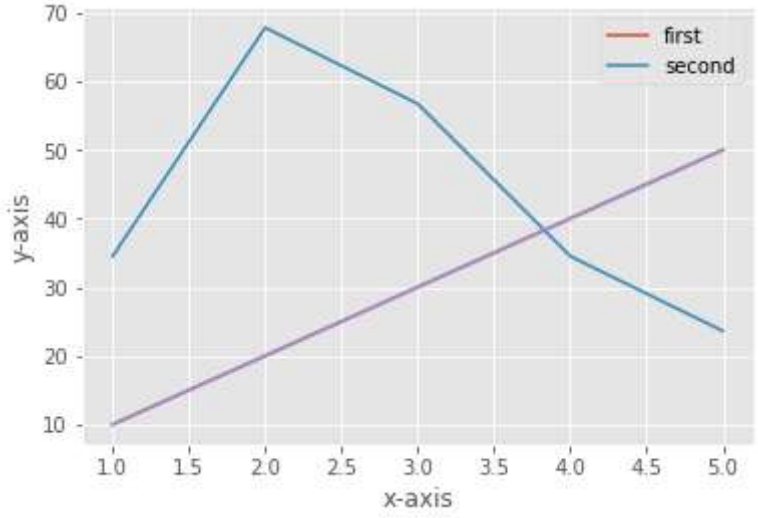


In [26]:
```
#adding x,y lables
plt.plot(x,y,label='first')
plt.plot(x,z,label='second')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.legend()
plt.show()
```

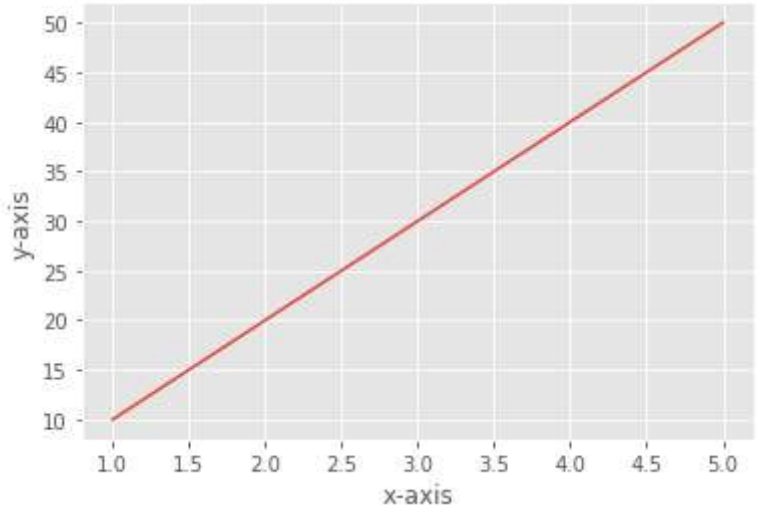


In [ ]:
```
```

In [27]:
```
plt.plot(x,y,label='first')
plt.plot(x,z,label='second')
plt.plot(x,y)
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.legend()
plt.show()
```



In [28]:
```
plt.plot(x,y)
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.show()
```



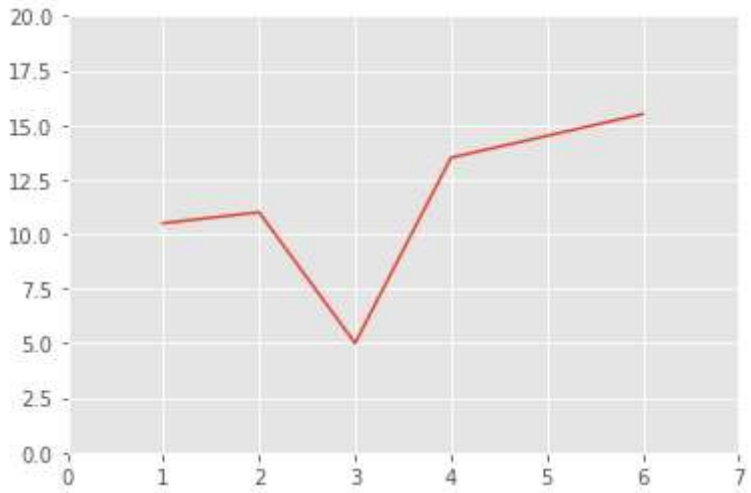
In [ ]:
```
```

In [29]:
```
import matplotlib as mlt
```

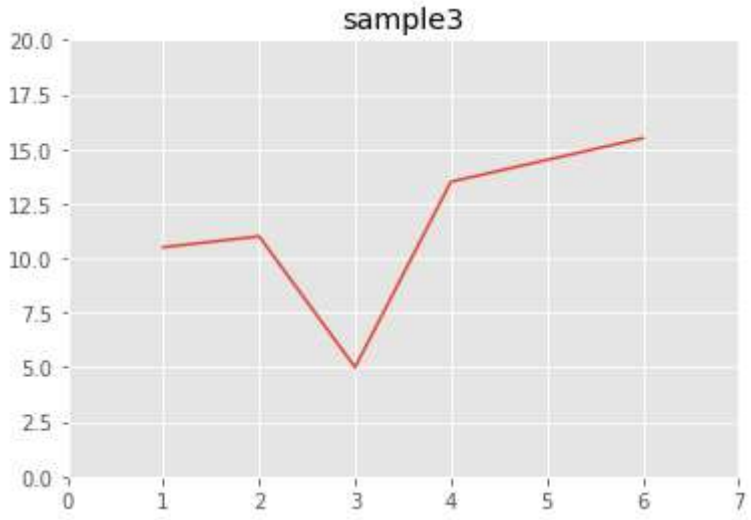
In [30]:
```
mlt.__version__
```

Out[30]:
```
'3.3.4'
```

In [31]:
```
#take 2 list of elements of diff length
li=[1,2,3,4,5,6] #empty char
li2=[10.5,11,5,13.5,14.5,15.5]
plt.plot(li,li2)
plt.axis([0,7,0,20])              #axis range
plt.show()
```

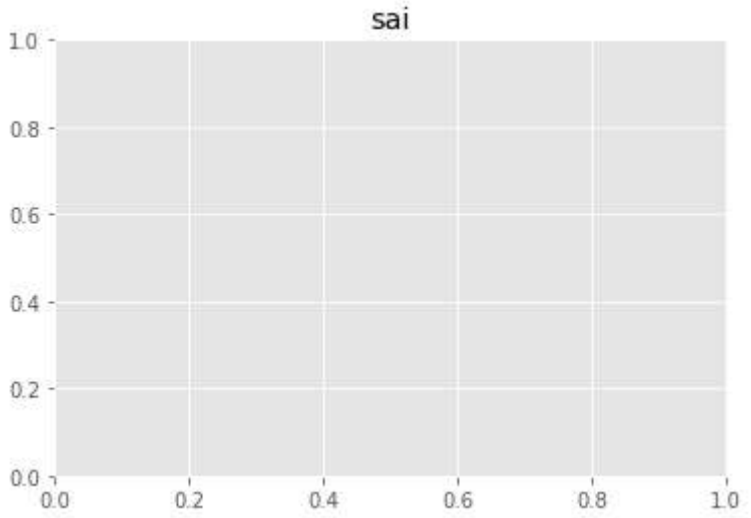


In [32]:
```
li=[1,2,3,4,5,6] #empty char
li2=[10.5,11,5,13.5,14.5,15.5]
plt.plot(li,li2)
plt.axis([0,7,0,20])              #axis range
plt.title('sample3')
plt.show()
```

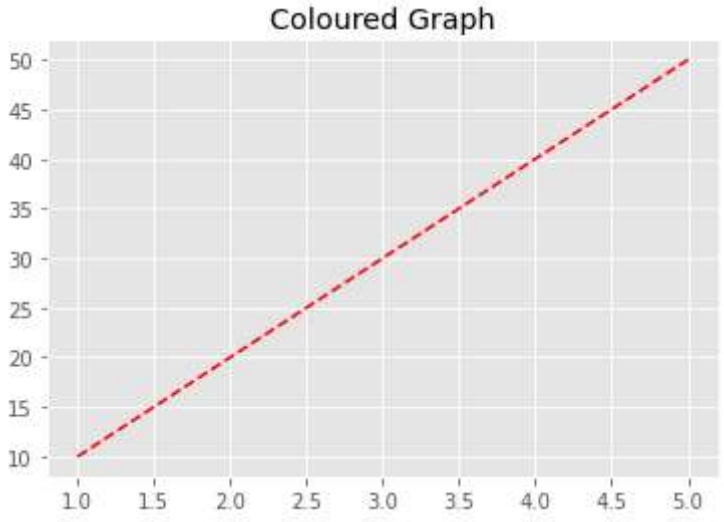


In [33]:
```
plt.title('sai')            #empty graph plot b/w 0 to 10 by default
```

Out[33]:
```
Text(0.5, 1.0, 'sai')
```

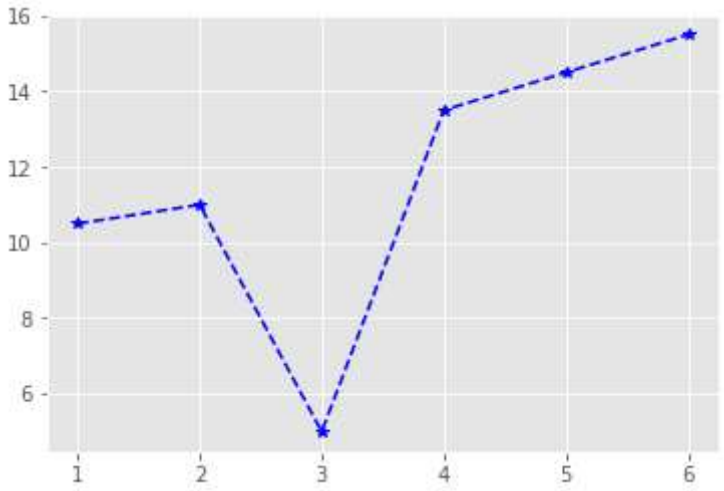


In [34]:
```
plt.plot(x,y,color='r',linestyle='--')
plt.title('Coloured Graph')
plt.show()
```



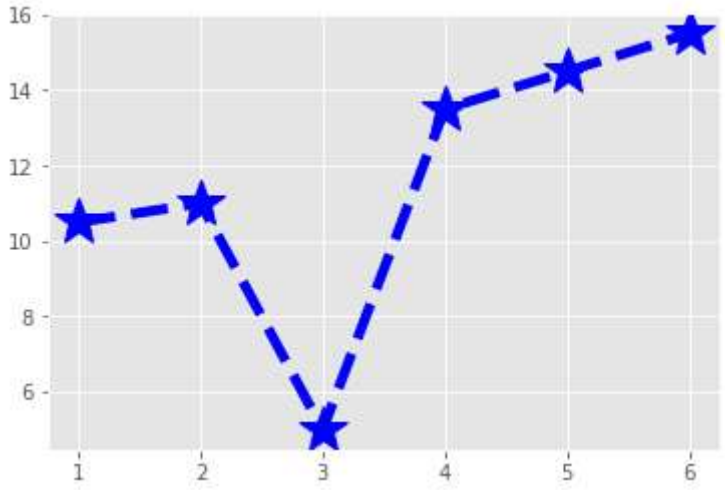
In [35]:
```
plt.plot(li,li2,color='b',linestyle='dashed',marker='*',)
```

Out[35]:
```
[<matplotlib.lines.Line2D at 0x2912aea9e80>]
```



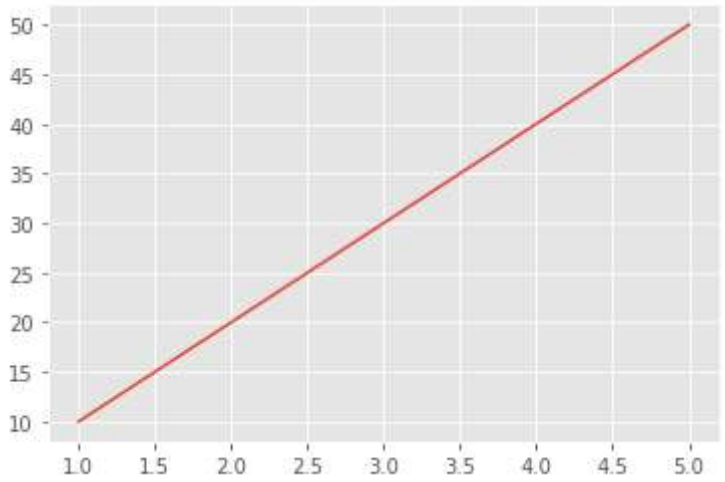
In [36]:
```
plt.plot(li,li2,color='b',linestyle='dashed',marker='*',linewidth=5,markersize=25
```

Out[36]:
```
[<matplotlib.lines.Line2D at 0x2912af57790>]
```

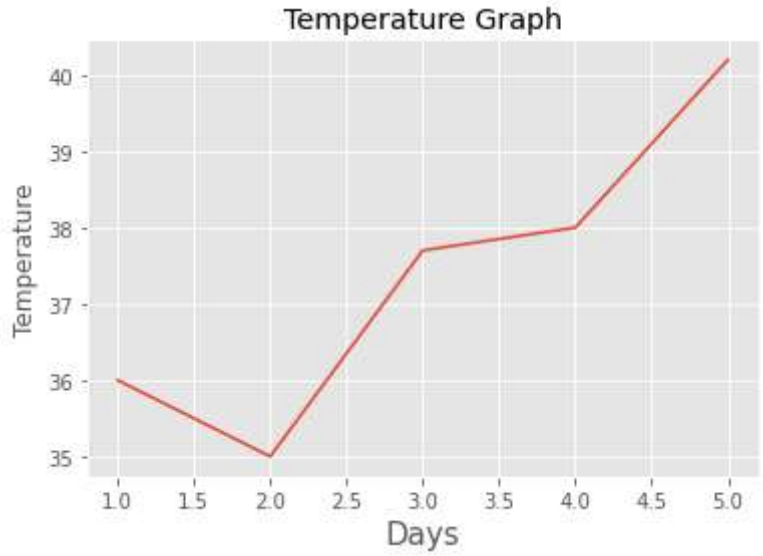


In [37]:
```
#ggplot
from matplotlib import style
style.use('ggplot')
plt.plot(x,y)
```

Out[37]:
```
[<matplotlib.lines.Line2D at 0x2912ad33190>]
```

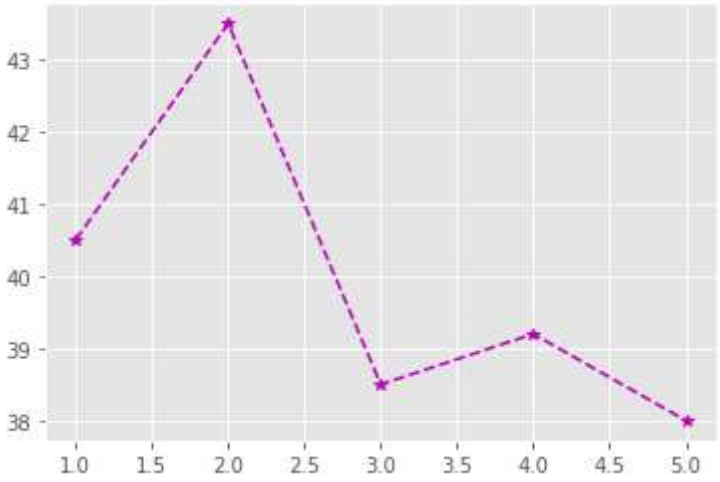


In [38]:
```
#ggplot
from matplotlib import style
style.use('ggplot')
#plt.plot(x,y)
days=[1,2,3,4,5]
temp=[36,35,37.7,38,40.2]
plt.plot(days,temp)
plt.xlabel('Days',fontsize=15)
plt.ylabel('Temperature')
plt.title("Temperature Graph")
plt.show()
```



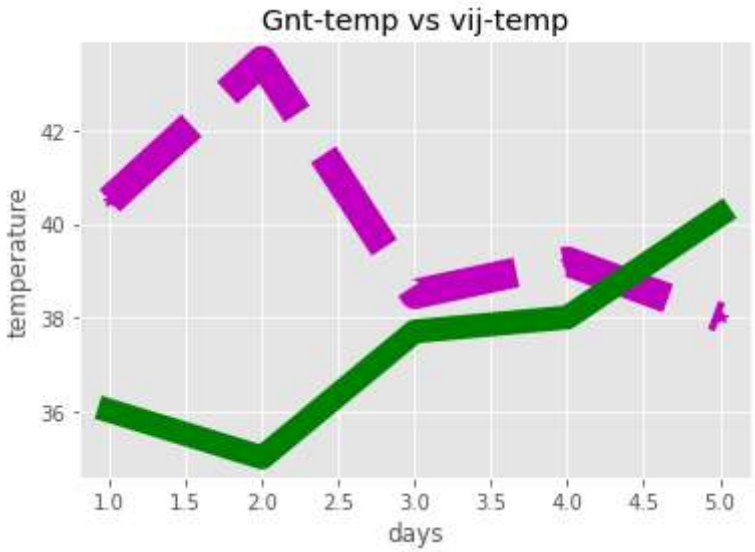
In [39]:
```
vij_temp=[40.5,43.5,38.5,39.2,38]
plt.plot(days,vij_temp,'m*--')
```

Out[39]:
```
[<matplotlib.lines.Line2D at 0x2912b5b11c0>]
```

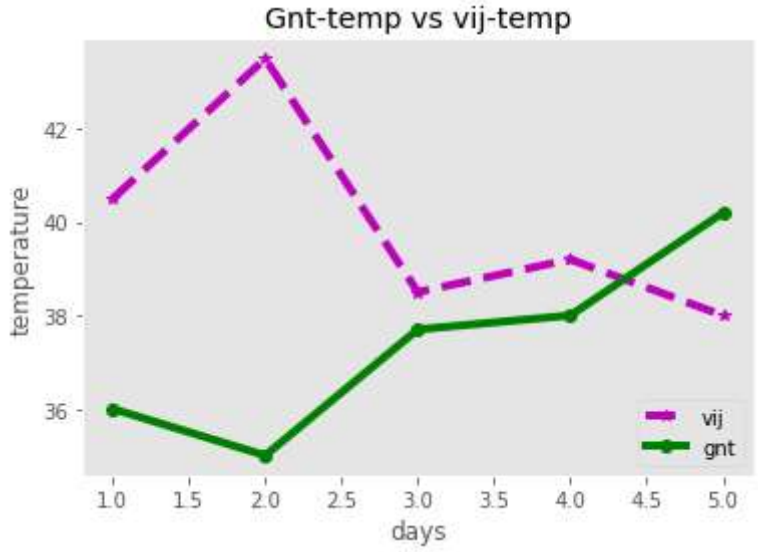


In [40]:
```
vij_temp=[40.5,43.5,38.5,39.2,38]
plt.plot(days,vij_temp,'m*--',linewidth=15)
plt.plot(days,temp,color='g',marker='o',linewidth=12)
plt.title('Gnt-temp vs vij-temp')
plt.xlabel('days')
plt.ylabel('temperature')
```

Out[40]:
```
Text(0, 0.5, 'temperature')
```



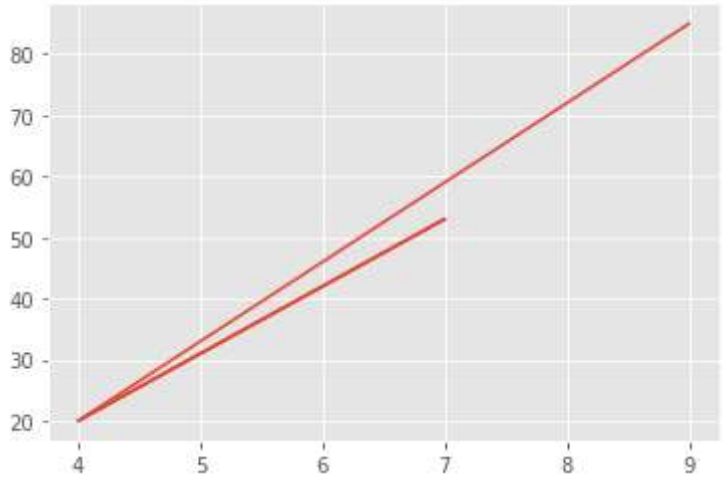
In [41]:
```
vij_temp=[40.5,43.5,38.5,39.2,38]
plt.plot(days,vij_temp,'m*--',linewidth=4,label='vij')
plt.plot(days,temp,color='g',marker='o',linewidth=4,label='gnt')
plt.title('Gnt-temp vs vij-temp')
plt.xlabel('days')
plt.ylabel('temperature')
plt.legend(loc=4)
plt.grid(color='b',linestyle='-',linewidth=2)
plt.show()
```



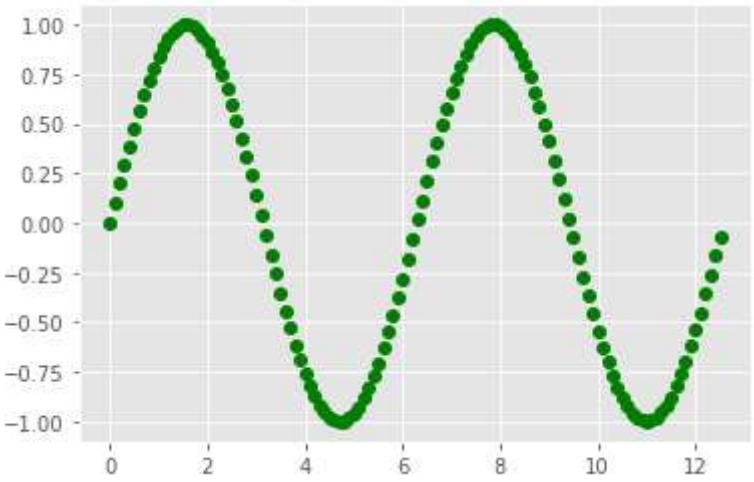
plotting equations

In [42]:
```
import numpy as np
```

In [43]:
```
x = np.random.randint(1,15,4)
y = x**2+4

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



In [44]:
```
# plotting sine funtion/wave

x = np.arange(0,4*np.pi,0.1)
y = np.sin(x)

plt.plot(x,y,'go') # g = green color; o = circle
plt.show()
```

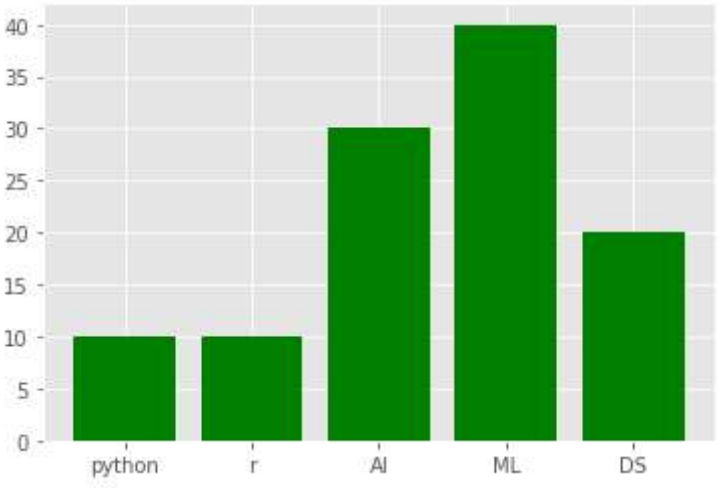


Bar Graphs

- bar graph
- histogram/hist

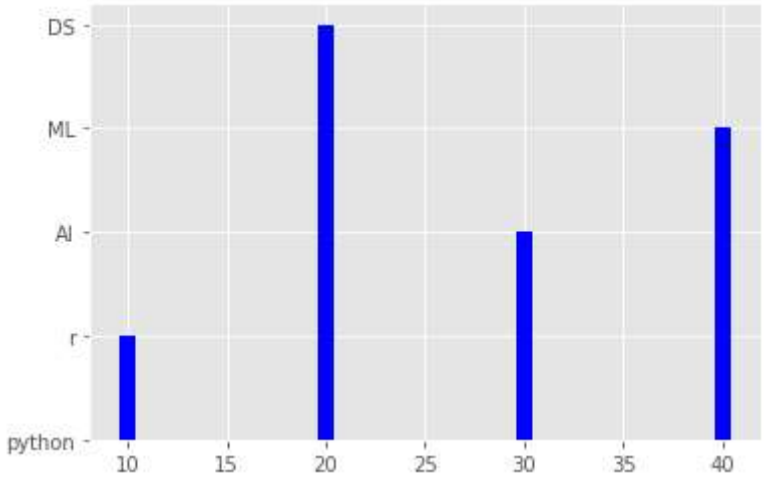
In [45]:
```
classes = ['python','r','AI','ML','DS']
st = [10,10,30,40,20]
st2 = [50,30,27,15,60]
st3 = [20,15,5,30,23]

plt.bar(classes,st,color='g')
```

Out[45]:
```
<BarContainer object of 5 artists>
```

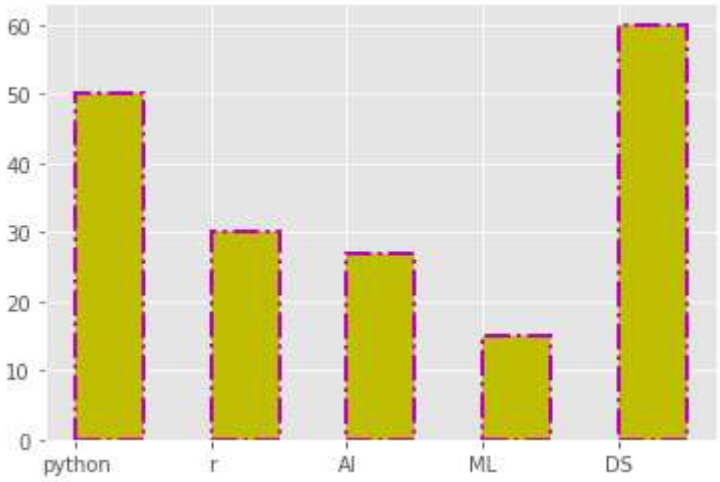


In [46]:
```
plt.bar(st,classes,color='b')
```

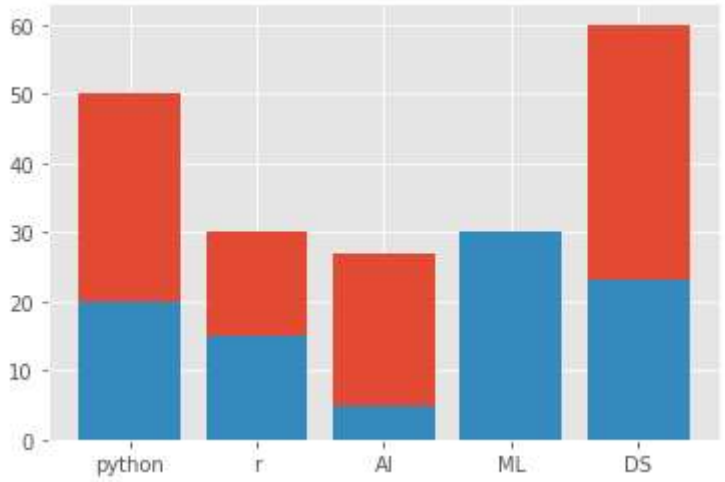
Out[46]: <BarContainer object of 5 artists>



In [47]:
```
plt.bar(classes,st2,width=0.5, color='y',align='edge',edgecolor='m',linewidth=2,l
plt.show()
```

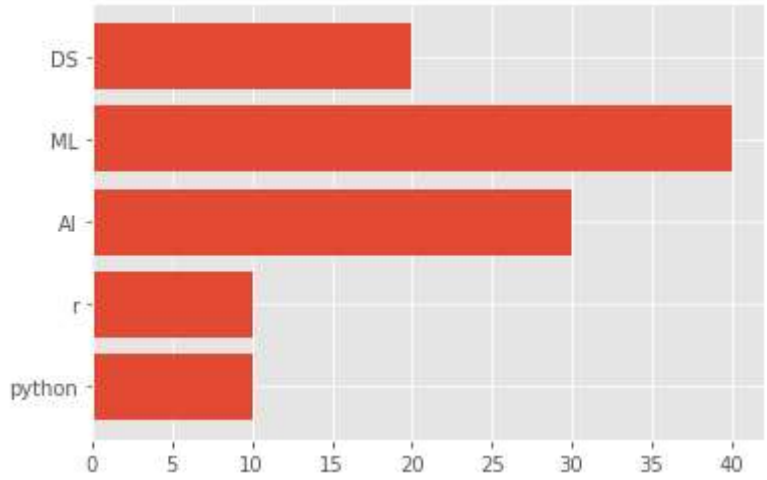


In [48]:
```
plt.bar(classes,st2)
plt.bar(classes,st3)
plt.show()
```



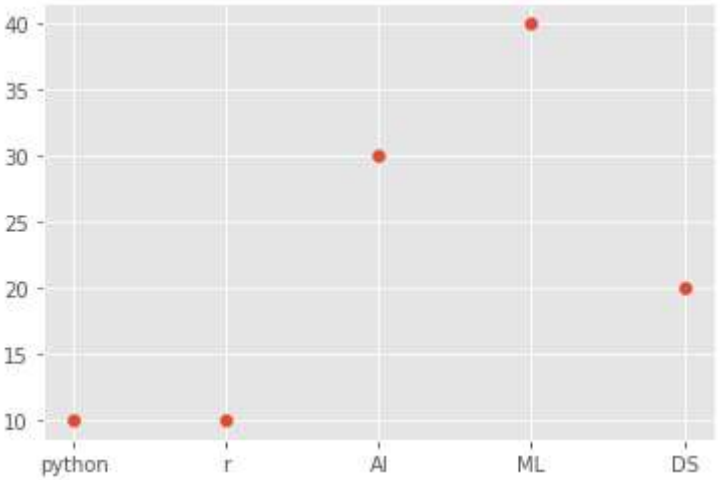
In [49]:
```
# horizantal bargraph

plt.barh(classes,st)
```

Out[49]: <BarContainer object of 5 artists>



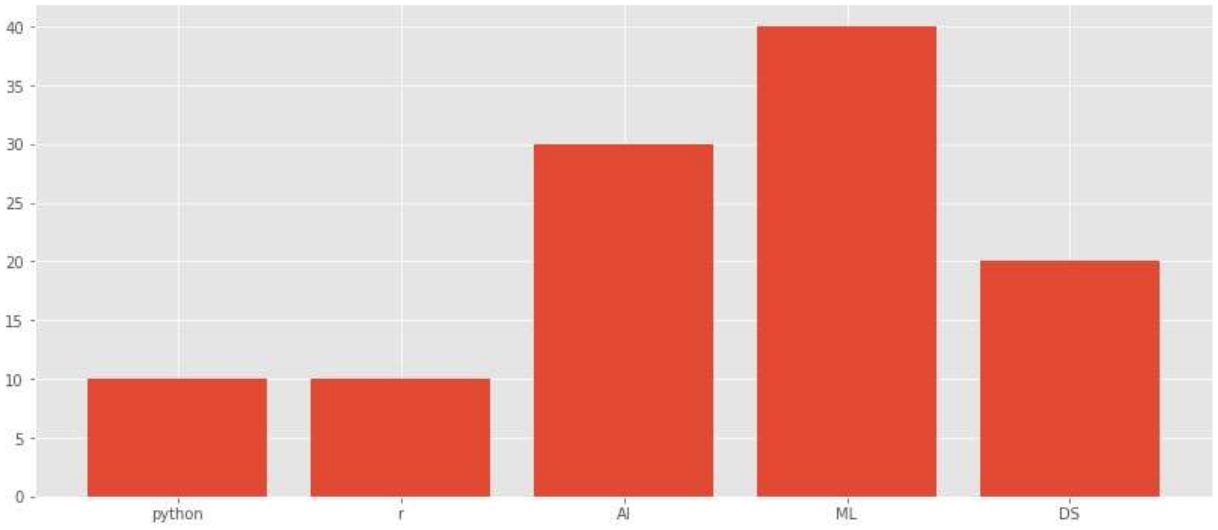
In [50]:
```
plt.scatter(classes,st)
```

Out[50]:
```
<matplotlib.collections.PathCollection at 0x2912b5f4670>
```



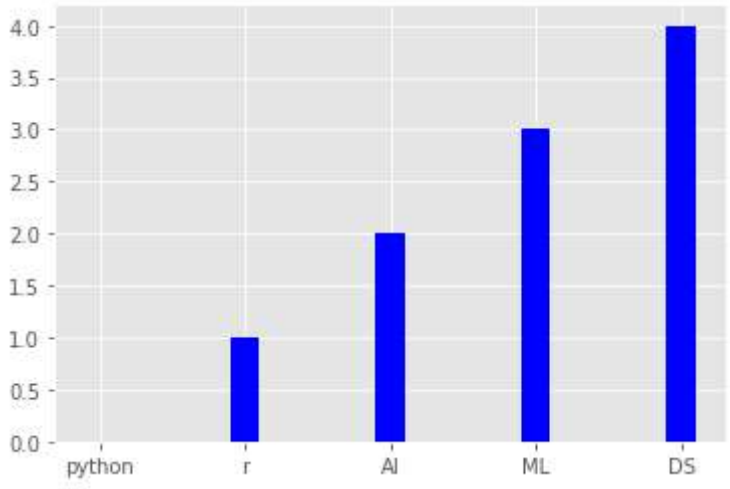
In [51]:
```
plt.figure(figsize=(14,6))
plt.bar(classes,st)
```

Out[51]:
```
<BarContainer object of 5 artists>
```



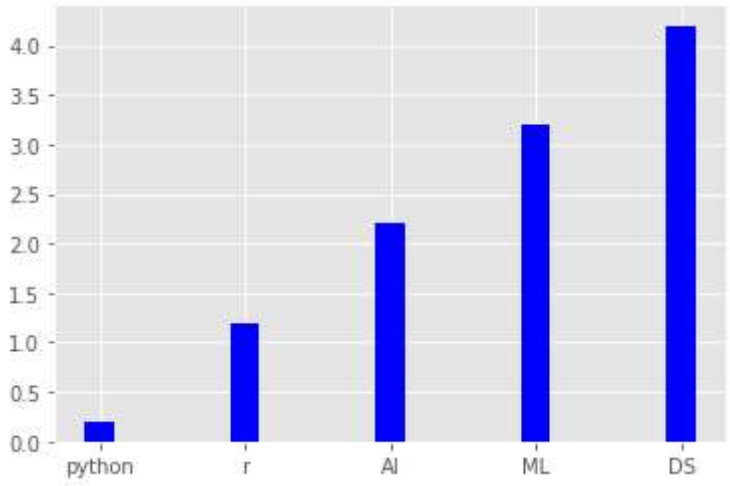
In [52]:
```
cls_index = np.arange(len(classes))
width = 0.2
plt.bar(classes,cls_index,width,color='b')
```

Out[52]: <BarContainer object of 5 artists>



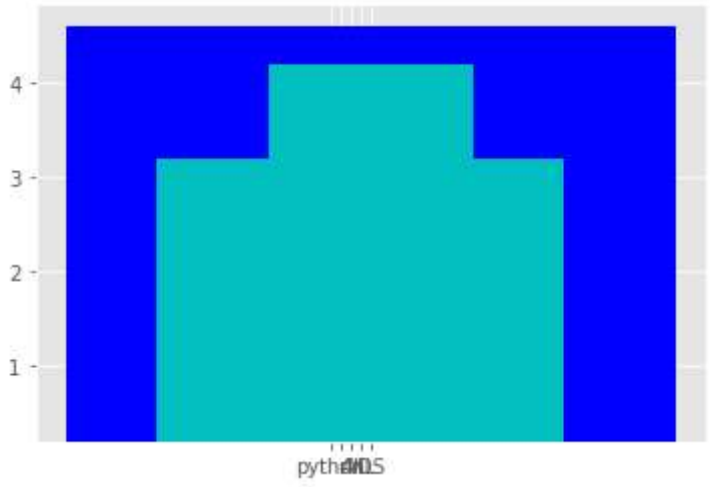
In [53]:
```
plt.bar(classes,cls_index+width,width,color='b')
```

Out[53]: <BarContainer object of 5 artists>



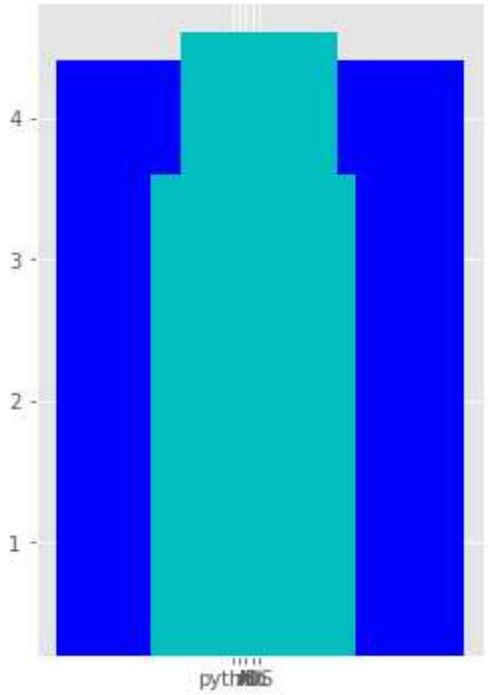
In [54]:
```
plt.bar(classes,cls_index,st,width,color='g')
plt.bar(classes,cls_index+width*2,st2,width,color='b')
plt.bar(classes,cls_index,st,width,color='c')
```

Out[54]: <BarContainer object of 5 artists>



In [55]:
```
plt.figure(figsize=(4,6))

plt.bar(classes,cls_index,st,width,color='g')
plt.bar(classes,cls_index+width,st2,width,color='b')
plt.bar(classes,cls_index+width+width,st3,width,color='c')

plt.xticks(cls_index+width,width,classes,rotation=40)
plt.yticks(np.arange(0,60,15))
```

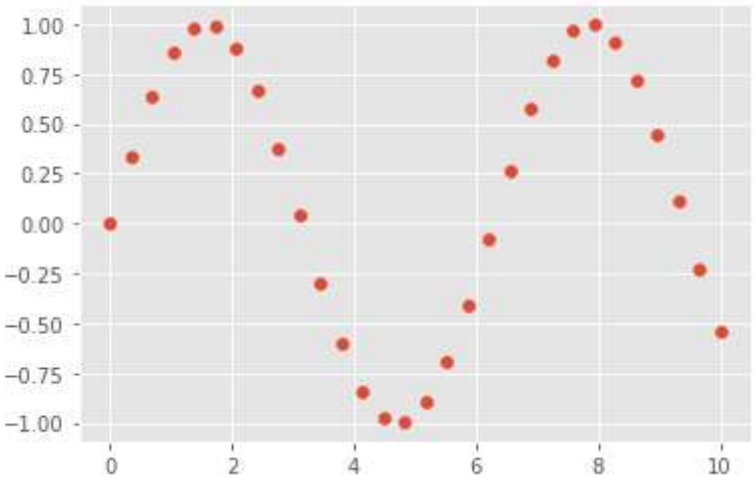
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-55-d52009129663> in <module>
      5 plt.bar(classes,cls_index+width+width,st3,width,color='c')
      6
----> 7 plt.xticks(cls_index+width,width,classes,rotation=40)
      8 plt.yticks(np.arange(0,60,15))

TypeError: xticks() takes from 0 to 2 positional arguments but 3 were given
```



In [56]:
```
# sine function

import matplotlib.pyplot as plt
import numpy as np
```

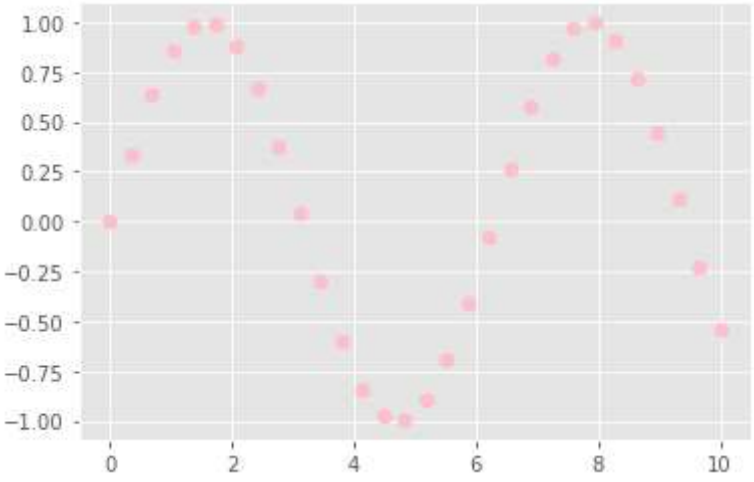
In [57]:
```
x = np.linspace(0,10,30)
y = np.sin(x)
plt.scatter(x,y)
```

Out[57]: <matplotlib.collections.PathCollection at 0x2912b125880>



In [58]:
```
plt.scatter(x,y,color='pink',s=50)
```

Out[58]: <matplotlib.collections.PathCollection at 0x2912b14dca0>



In [59]:
```
from sklearn.datasets import load_iris
```

In [60]:
```
iris = load_iris()
print(iris)
iris
```

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
```

In [61]:
```
print(iris)
```

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
```

In [62]:
```
print(iris.DESCR)
```

```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. topic:: References

   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
```

```
    Structure and Classification Rule for Recognition in Partially Exposed
    Environments".  IEEE Transactions on Pattern Analysis and Machine
    Intelligence, Vol. PAMI-2, No. 1, 67-71.
- Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
    on Information Theory, May 1972, 431-433.
- See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
    conceptual clustering system finds 3 classes in the data.
- Many, many more ...
```

In [63]:
```
type(iris)
```

Out[63]:
```
sklearn.utils.Bunch
```

In [64]:
```
iris['data']
```

Out[64]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
```

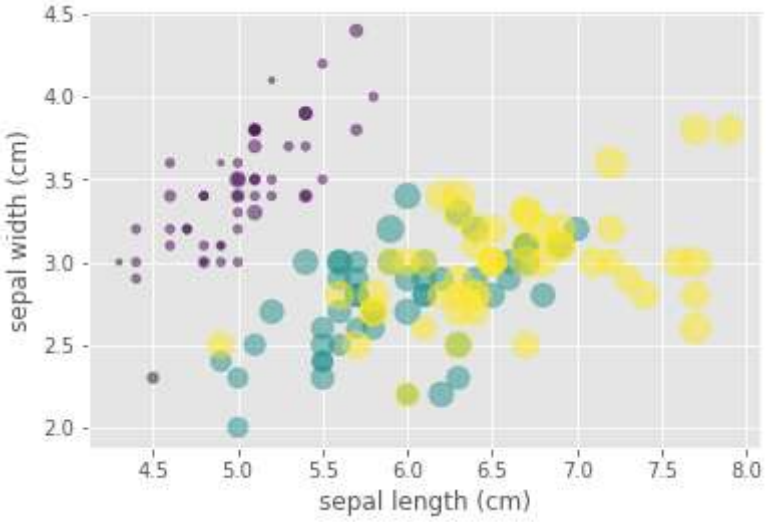
In [65]:
```
import matplotlib.pyplot as plt
```

In [66]:
```
features = iris.data.T
```

In [67]:
```
features[0]
```

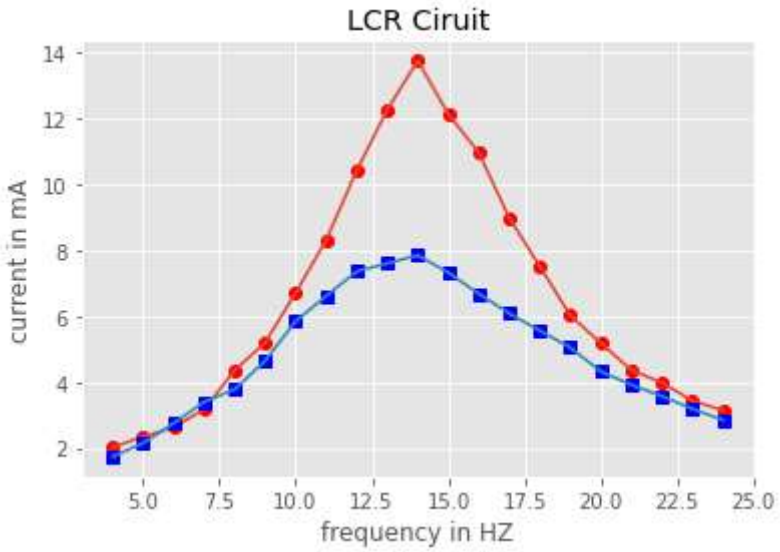
Out[67]:
```
array([5.1, 4.9, 4.7, 4.6, 5. , 5.4, 4.6, 5. , 4.4, 4.9, 5.4, 4.8, 4.8,
       4.3, 5.8, 5.7, 5.4, 5.1, 5.7, 5.1, 5.4, 5.1, 4.6, 5.1, 4.8, 5. ,
       5. , 5.2, 5.2, 4.7, 4.8, 5.4, 5.2, 5.5, 4.9, 5. , 5.5, 4.9, 4.4,
       5.1, 5. , 4.5, 4.4, 5. , 5.1, 4.8, 5.1, 4.6, 5.3, 5. , 7. , 6.4,
       6.9, 5.5, 6.5, 5.7, 6.3, 4.9, 6.6, 5.2, 5. , 5.9, 6. , 6.1, 5.6,
       6.7, 5.6, 5.8, 6.2, 5.6, 5.9, 6.1, 6.3, 6.1, 6.4, 6.6, 6.8, 6.7,
       6. , 5.7, 5.5, 5.5, 5.8, 6. , 5.4, 6. , 6.7, 6.3, 5.6, 5.5, 5.5,
       6.1, 5.8, 5. , 5.6, 5.7, 5.7, 6.2, 5.1, 5.7, 6.3, 5.8, 7.1, 6.3,
       6.5, 7.6, 4.9, 7.3, 6.7, 7.2, 6.5, 6.4, 6.8, 5.7, 5.8, 6.4, 6.5,
       7.7, 7.7, 6. , 6.9, 5.6, 7.7, 6.3, 6.7, 7.2, 6.2, 6.1, 6.4, 7.2,
       7.4, 7.9, 6.4, 6.3, 6.1, 7.7, 6.3, 6.4, 6. , 6.9, 6.7, 6.9, 5.8,
       6.8, 6.7, 6.7, 6.3, 6.5, 6.2, 5.9])
```

In [69]:
```
plt.scatter(features[0],features[1], s=100*features[3], c=iris.target,alpha=0.5)

plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])

plt.show()
```



LCR Circuit Graph

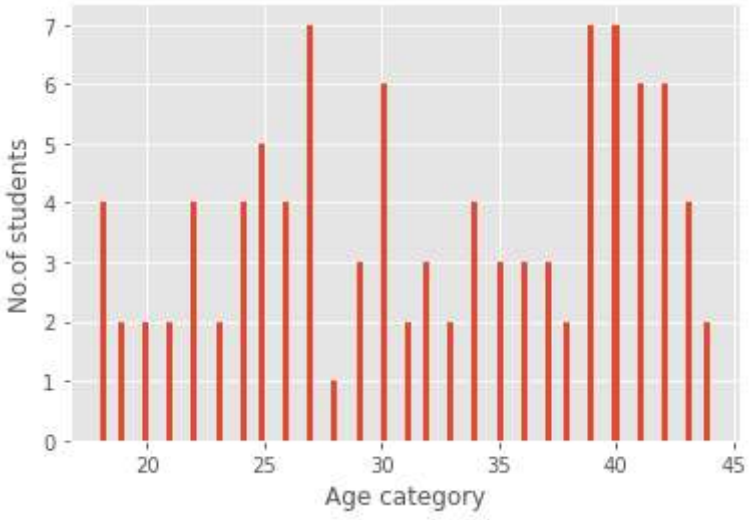
In [111]:
```
# to draw the graph of the LCR circuit
import matplotlib.pyplot as plt

f = [s for s in range(4,25,1)]
# here values of 'f' are frequencies

r200 = [2.02,2.35,2.62,3.18,4.36,5.20,6.71,8.31,10.42,12.26,13.75,12.13,10.96,8.9
# r200 is the resistance of 200 ohms  for current

r500 = [1.74,2.15,2.75,3.40,3.78,4.65,5.85,6.61,7.36,7.60,7.85,7.32,6.68,6.08,5.5
# r500 is the resistance of 500 ohms for current

plt.plot(f,r200,"ro")
plt.plot(f,r200)

plt.plot(f,r500,'bs')
plt.plot(f,r500)

plt.xlabel('frequency in HZ')
plt.ylabel('current in mA')

plt.title('LCR Ciruit')
plt.show()
```



Histogram

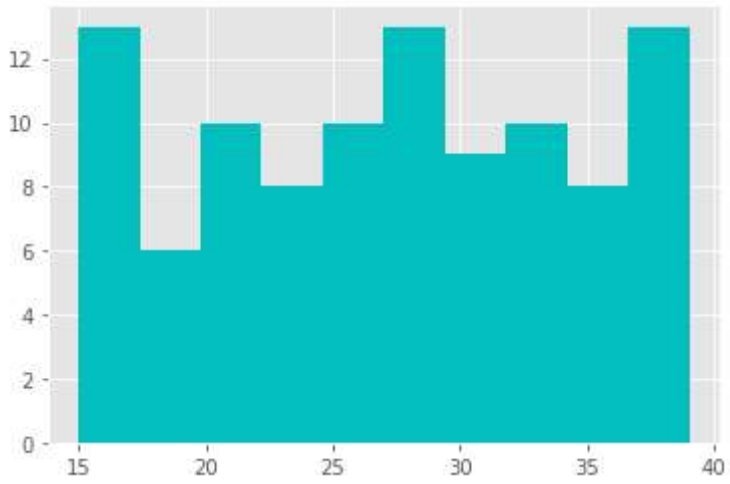
- represented as bins
- bins represets the data points in range
- plt.hist(ele,bins = count) == > syntax

In [71]:
```
import numpy as np

std1 = np.random.randint(18,45,100)
std2 = np.random.randint(15,40,100)

plt.hist(std1,bins=100)
plt.xlabel('Age category')
plt.ylabel('No.of students')
plt.show()
```



In [72]:
```
plt.hist(std2,bins=10,color='c')
```

Out[72]:
```
(array([13.,  6., 10.,  8., 10., 13.,  9., 10.,  8., 13.]),
 array([15. , 17.4, 19.8, 22.2, 24.6, 27. , 29.4, 31.8, 34.2, 36.6, 39. ]),
 <BarContainer object of 10 artists>)
```



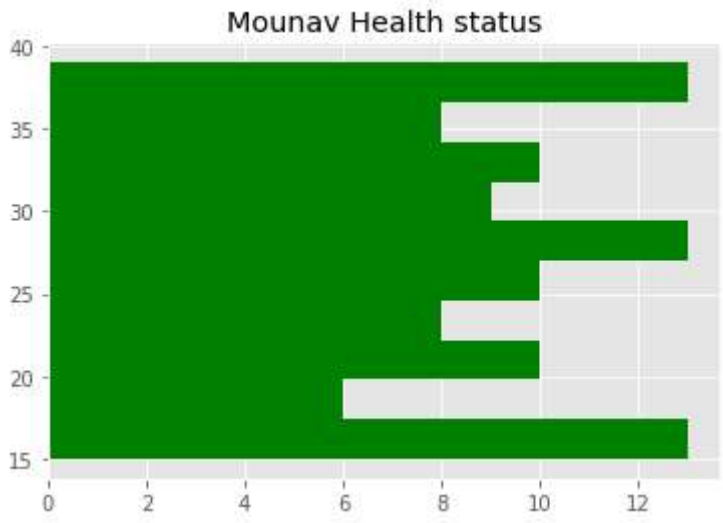
In [73]:
```
std1
```

Out[73]:
```
array([40, 27, 21, 43, 30, 39, 41, 34, 31, 19, 28, 41, 25, 27, 23, 22, 29,
       43, 20, 30, 26, 41, 40, 22, 44, 18, 43, 40, 26, 42, 41, 18, 41, 41,
       35, 36, 34, 44, 39, 32, 19, 20, 37, 29, 39, 32, 43, 39, 30, 18, 35,
       39, 34, 25, 32, 27, 24, 30, 24, 23, 30, 25, 42, 37, 39, 38, 34, 36,
       30, 40, 27, 18, 24, 31, 35, 40, 42, 40, 21, 22, 26, 42, 38, 26, 24,
       22, 27, 27, 40, 25, 27, 42, 36, 37, 25, 42, 39, 33, 33, 29])
```

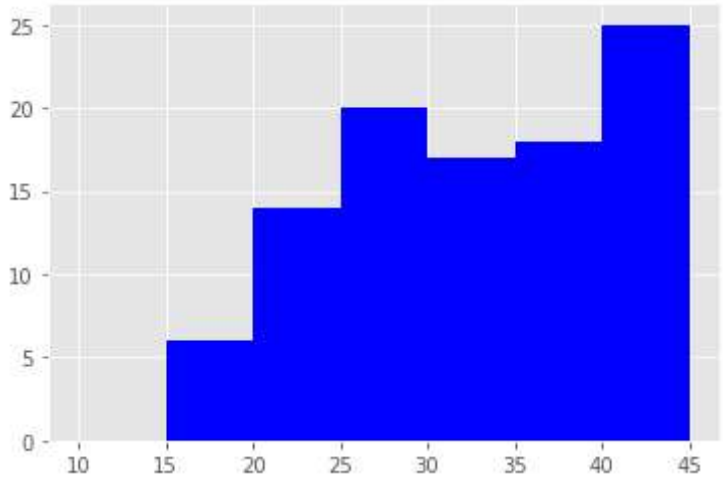
In [74]:
```
std2
```

Out[74]:
```
array([29, 35, 33, 16, 21, 39, 38, 30, 35, 19, 16, 28, 31, 25, 35, 26, 37,
       26, 35, 21, 24, 26, 33, 18, 24, 37, 39, 31, 15, 27, 27, 37, 23, 31,
       26, 24, 39, 17, 25, 31, 24, 22, 21, 21, 38, 39, 30, 29, 20, 36, 34,
       37, 33, 17, 24, 20, 26, 34, 25, 28, 38, 27, 23, 20, 34, 15, 26, 20,
       16, 36, 18, 29, 15, 28, 29, 17, 36, 39, 18, 16, 18, 17, 15, 26, 32,
       29, 27, 30, 31, 39, 32, 36, 33, 22, 24, 28, 16, 19, 31, 32])
```

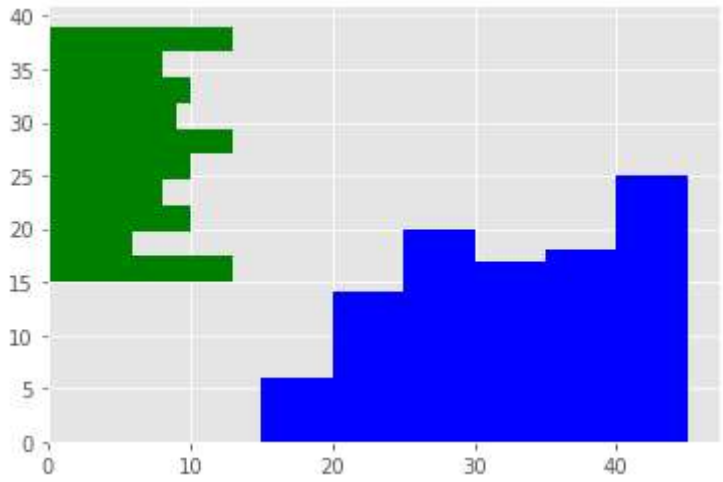
In [75]:
```
plt.hist(std2,bins=10,color='g',orientation='horizontal')
plt.title('Mounav Health status')
plt.show()
```



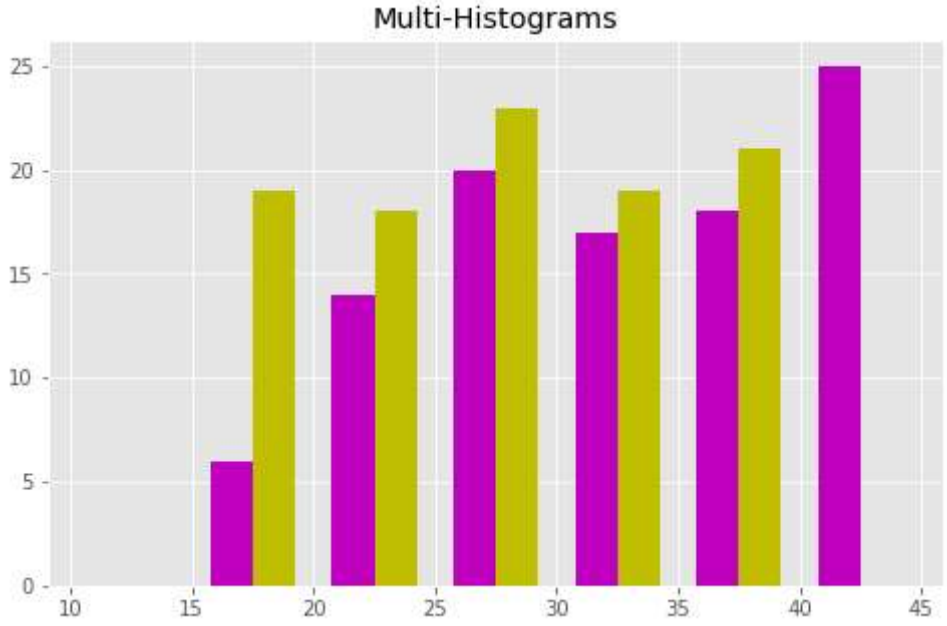
In [76]:
```
vals = [num for num in range(10,50,5)]

plt.hist(std1,vals, histtype='bar',orientation='vertical',color='b')

plt.show()
```



In [77]:
```
vals = [num for num in range(10,50,5)]

plt.hist(std1,vals, histtype='bar',orientation='vertical',color='b')
plt.hist(std2,bins=10,color='g',orientation='horizontal')
plt.show()
```

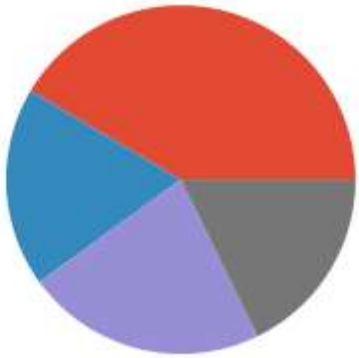


In [80]:
```
from matplotlib import style

style.use('ggplot')

plt.figure(figsize=(8,5))
plt.hist([std1,std2],vals, rwidth=0.7,histtype='bar',orientation='vertical',color
plt.title('Multi-Histograms')
plt.show()
```

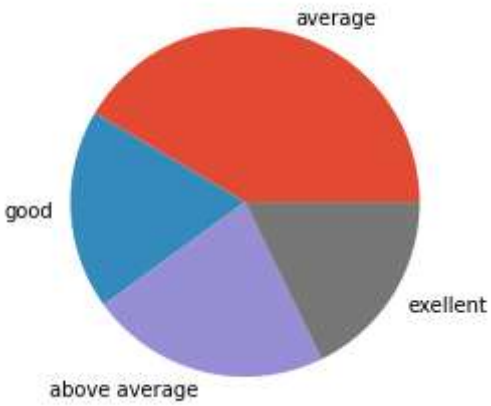


Pie chart plot (pie-plot)

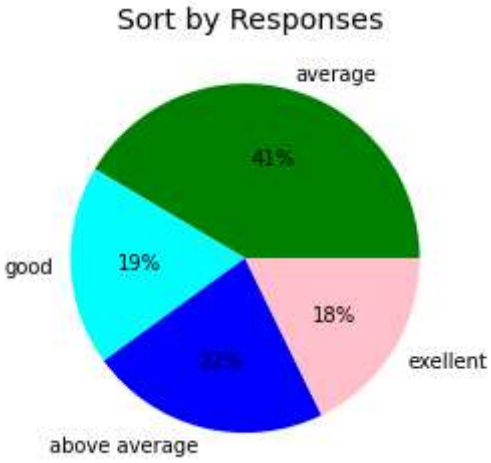
In [81]:
```
x = [56,25,30,24] # making the regions

plt.pie(x)
plt.show()
```



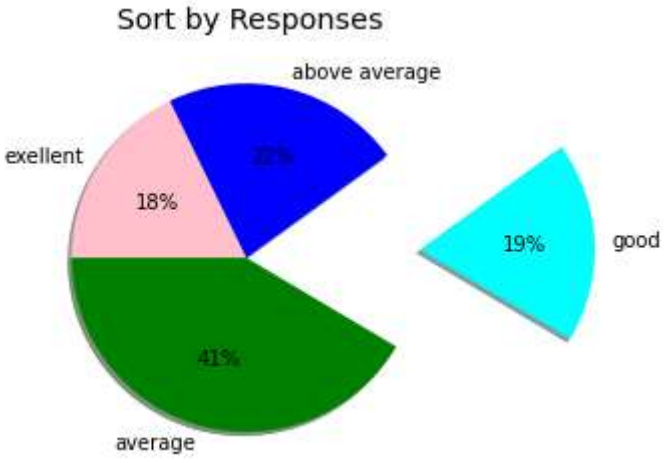
In [87]:
```
# adding labels

fb = ['average','good','above average','exellent']
plt.pie(x,labels=fb) # assingnig labels of fb to x
plt.show()
```



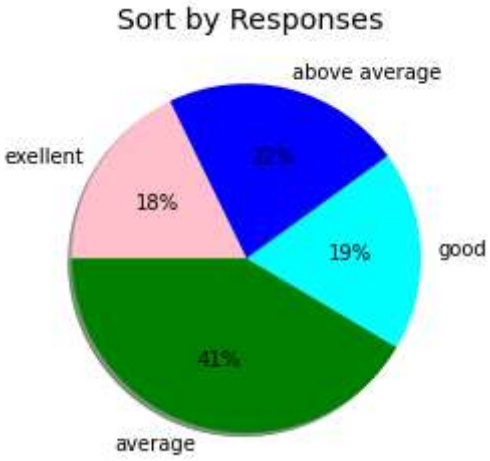
In [94]:
```
# addings colors to regions of pie
plt.pie(x,labels=fb, colors=['green','cyan','blue','pink'], autopct="%.f%%")

plt.title(' Sort by Responses')
plt.show()
```



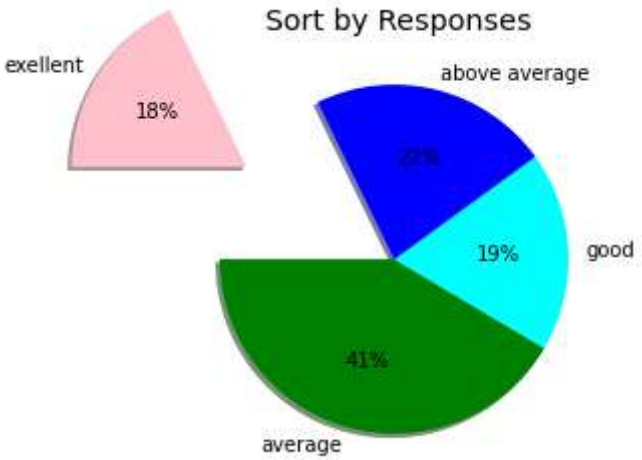
In [96]:
```
# addings shadow to regions of pie
plt.pie(x,labels=fb, colors=['green','cyan','blue','pink'], autopct="%.f%%", shad
# explode means exluding

plt.title(' Sort by Responses')
plt.show()
```



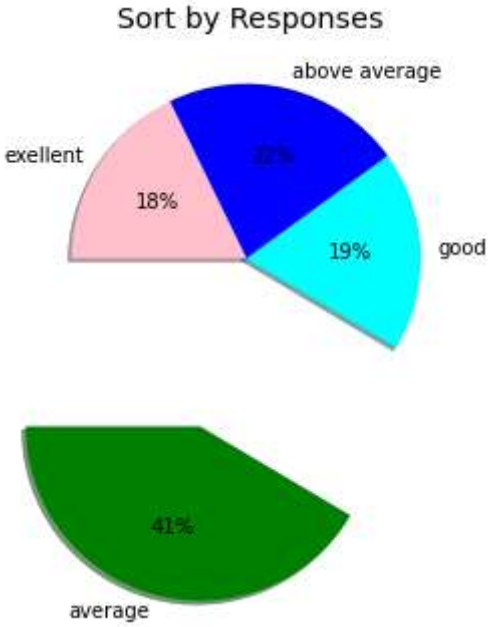
In [97]:
```
# addings shadow to regions of pie
plt.pie(x,labels=fb, colors=['green','cyan','blue','pink'], autopct="%.f%%", shad

plt.title(' Sort by Responses')
plt.show()
```



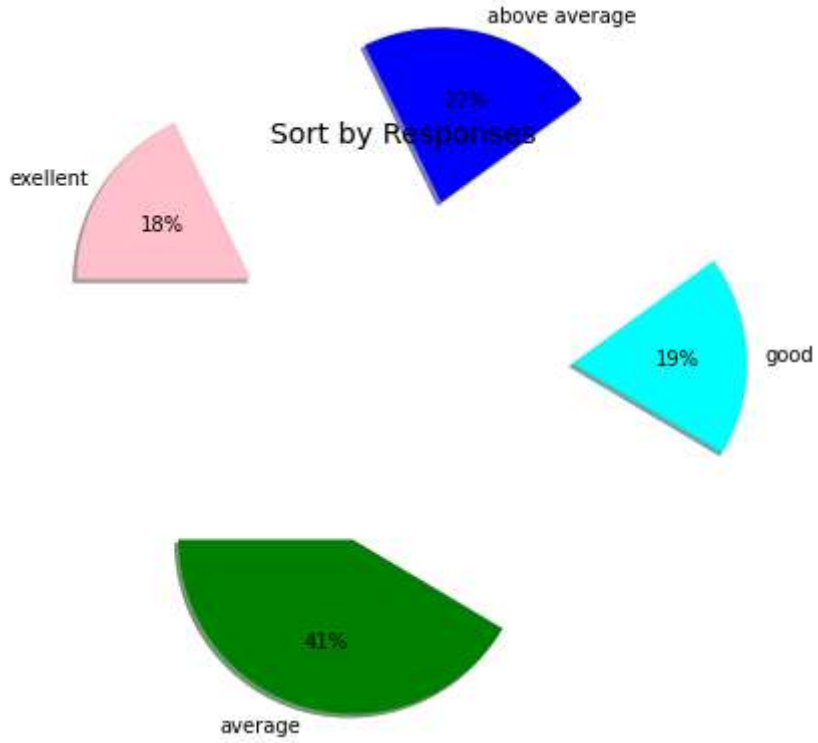
In [98]:
```
# addings shadow to regions of pie
plt.pie(x,labels=fb, colors=['green','cyan','blue','pink'], autopct="%.f%%", shad
# explode means exluding

plt.title(' Sort by Responses')
plt.show()
```



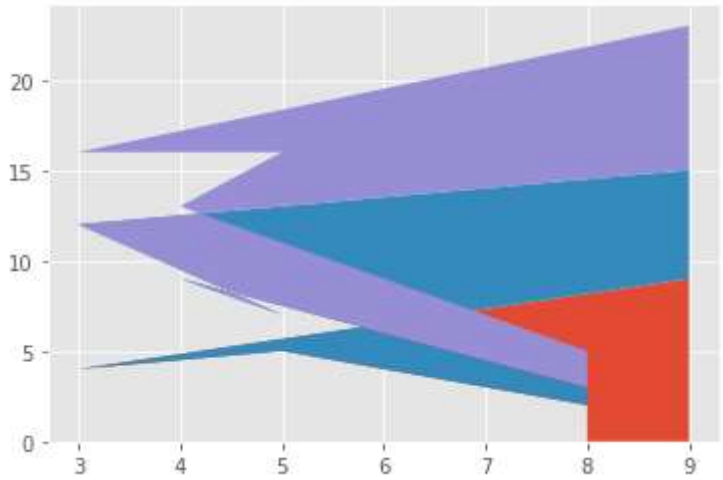
In [99]:
```
# addings shadow to regions of pie
plt.pie(x,labels=fb, colors=['green','cyan','blue','pink'], autopct="%.f%%", shad
# explode means exluding

plt.title(' Sort by Responses')
plt.show()
```



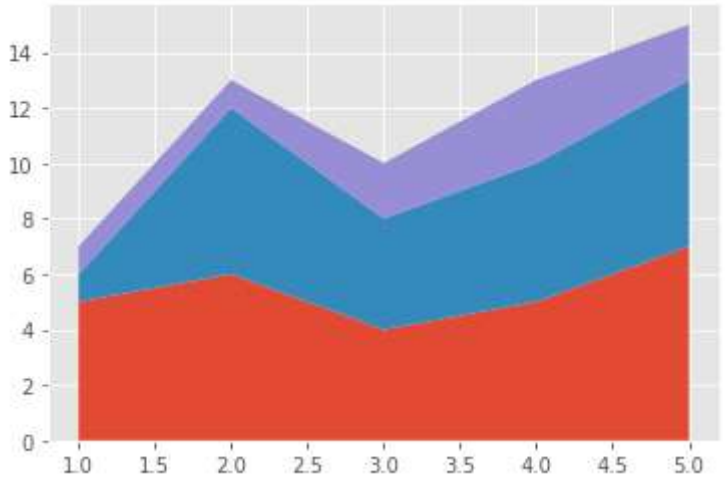
In [100]:
```
# addings shadow to regions of pie
plt.pie(x,labels=fb, colors=['green','cyan','blue','pink'], autopct="%.f%%", shad
# explode means exluding

plt.title(' Sort by Responses')
plt.show()
```



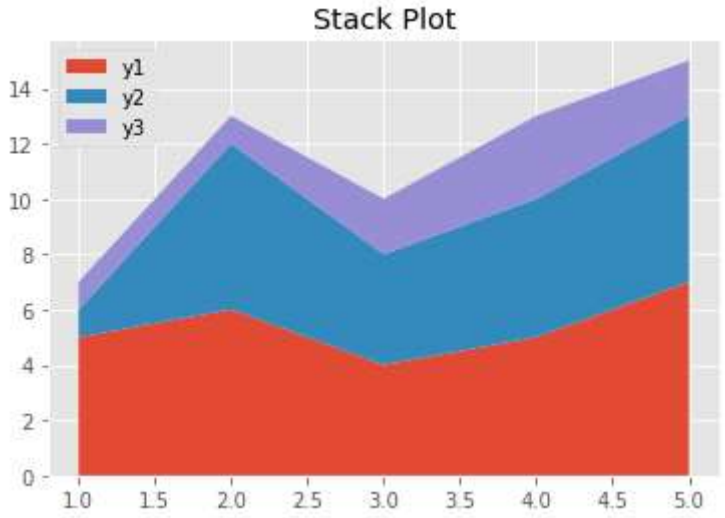
In [106]:
```
## stak plot
x1 = [9,3,5,4,8]
a = [9,4,5,6,2]
b = [6,8,2,3,1]
c = [8,4,9,4,2]
plt.stackplot(x1,a,b,c)

plt.show()
```



In [107]:
```
f = [1,2,3,4,5]
s = {'y1':[5,6,4,5,7], 'y2':[1,6,4,5,6], 'y3':[1,1,2,3,2]}

plt.stackplot(f,s.values())
plt.show()
```



In [108]:
```
# adding labels

plt.stackplot(f,s.values(),labels = s.keys())
plt.legend(loc = 'upper left')
plt.title('Stack Plot')
plt.show()
```



In [129]:
```
df
```

Out[129]:

	x	y
0	-0.669343	-0.259170
1	-0.006814	-0.830724
2	-0.476029	1.060543
3	1.248954	1.233699
4	-1.062834	0.695946
5	3.401332	-0.607716
6	-0.407058	-1.575471
7	-2.764721	-0.751533
8	-2.330415	0.104062
9	-0.832254	-2.380565

In [130]:
```
# box plot

b = pd.read_csv('wn.csv')
b
```

Out[130]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total	sulfur dioxide	density	pH	sulphates	alc
0	0	7.4	0.70	0.001.9	0.076	11.000	34.0	0.9978	3.5100	0.56	9.40	
1	1	7.8	0.88	0.00	2.600	0.098	25.0	67.0000	0.9968	3.20	0.68	
2	2	7.8	0.76	0.04	2.300	0.092	15.0	54.0000	0.9970	3.26	0.65	
3	3	11.2	0.28	0.56	1.900	0.075	17.0	60.0000	0.9980	3.16	0.58	
4	4	7.4	0.70	0.00	1.900	0.076	11.0	34.0000	0.9978	3.51	0.56	

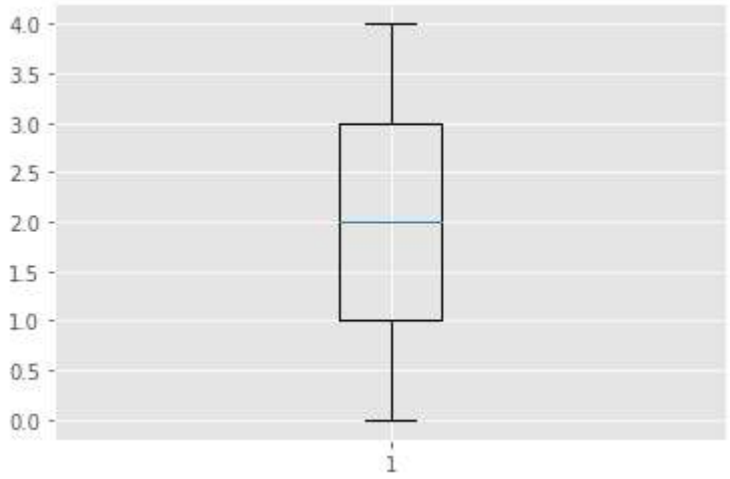
In [131]:
```
b.columns
```

Out[131]:
```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total', 'sulfur dioxide',
       'density', 'pH', 'sulphates', 'alcohol', ' quality'],
      dtype='object')
```

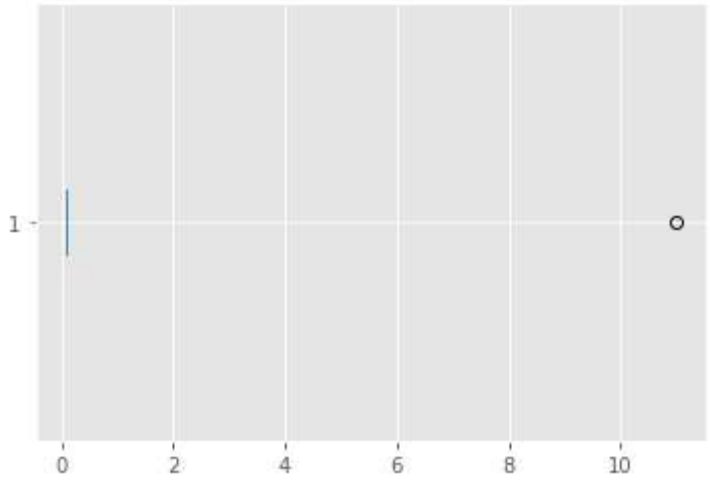
In [133]:
```
b.isnull().any()
```

Out[133]:
```
fixed acidity          False
volatile acidity       False
citric acid            False
residual sugar         False
chlorides              False
free sulfur dioxide    False
total                  False
sulfur dioxide         False
density                False
pH                     False
sulphates              False
alcohol                False
 quality                True
dtype: bool
```

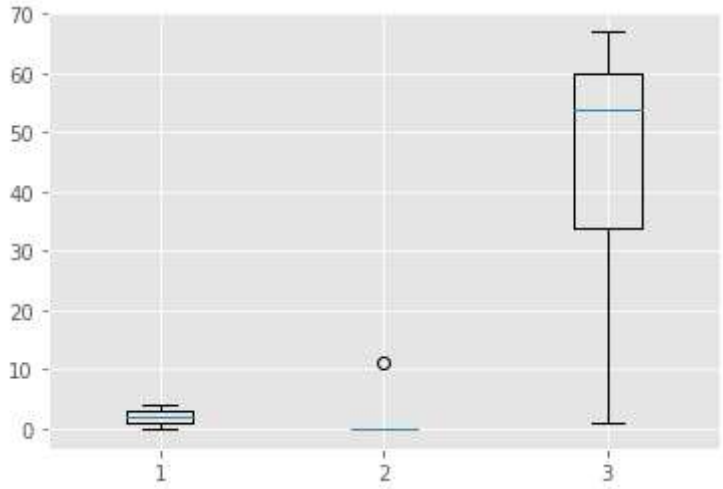
In [136]:
```
fixed_acid = b['fixed acidity']
free_sulpher = b['free sulfur dioxide']
total_sulpher = b['sulfur dioxide']

plt.boxplot(fixed_acid)
plt.show()
```



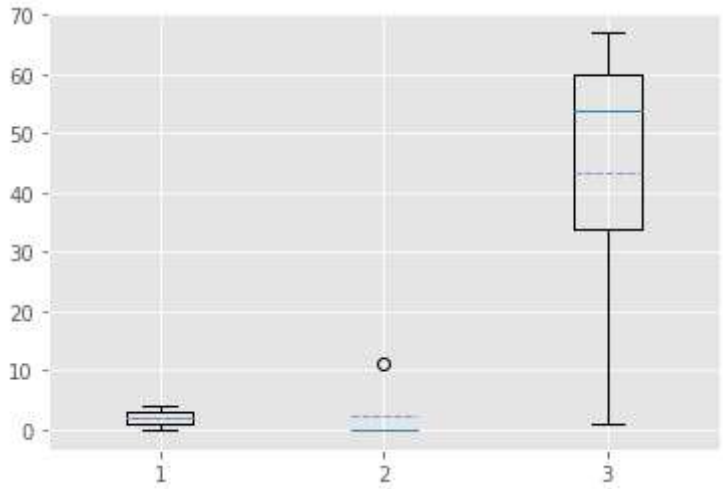
In [138]:
```
plt.boxplot(free_sulpher, vert=0)
plt.show()
```



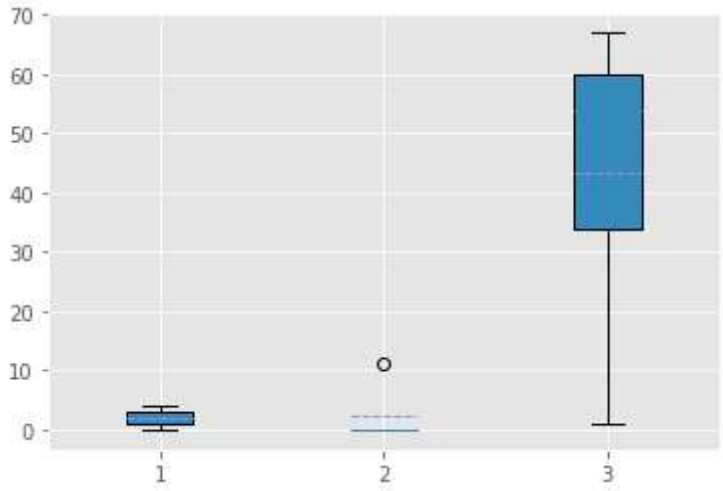
In [139]:
```
plt.boxplot([fixed_acid,free_sulpher,total_sulpher])
plt.show()
```



In [140]:
```
plt.boxplot([fixed_acid,free_sulpher,total_sulpher], meanline = True, showmeans =
plt.show()
```



In [141]:
```
plt.boxplot([fixed_acid,free_sulpher,total_sulpher], meanline = True, patch_artis
plt.show()
```



Course Overview

- Data is of different types
  - devices,statical,scalar,vector..
  - numpy:
    - used to create and deal with array type of data.
      - min(1) and max(32) dimensions
      - scientific computation
      - vectorize the normal function
  - pandas:
    - data creation
    - data manipulation
    - data processing/analysis
  - matplotlib:
    - plotting
    - visulizing the data
- scikit learn:
  - sklearn to load datasets

In [112]:
```
import numpy as np

dir(np)
```

Out[112]:
```
['ALLOW_THREADS',
 'AxisError',
 'BUFSIZE',
 'Bytes0',
 'CLIP',
 'ComplexWarning',
 'DataSource',
 'Datetime64',
 'ERR_CALL',
 'ERR_DEFAULT',
 'ERR_IGNORE',
 'ERR_LOG',
 'ERR_PRINT',
 'ERR_RAISE',
 'ERR_WARN',
 'FLOATING_POINT_SUPPORT',
 'FPE_DIVIDEBYZERO',
 'FPE_INVALID',
 'FPE_OVERFLOW',
```

In [119]:
```
import pandas as pd

df = pd.read_csv('wn.csv')
print('csv file: ',df)
df
```

```
csv file:      fixed acidity  volatile acidity  citric acid residual sugar  chlo
rides  \
0                0              7.4         0.70        0.001.9      0.076
1                1              7.8         0.88           0.00      2.600
2                2              7.8         0.76           0.04      2.300
3                3             11.2         0.28           0.56      1.900
4                4              7.4         0.70           0.00      1.900

   free sulfur dioxide  total  sulfur dioxide  density    pH  sulphates  \
0               11.000   34.0          0.9978   3.5100  0.56       9.40
1                0.098   25.0         67.0000   0.9968  3.20       0.68
2                0.092   15.0         54.0000   0.9970  3.26       0.65
3                0.075   17.0         60.0000   0.9980  3.16       0.58
4                0.076   11.0         34.0000   0.9978  3.51       0.56

   alcohol   quality
0      5.0       NaN
1      9.8       5.0
2      9.8       5.0
3      9.8       6.0
4      9.4       5.0
```

Out[119]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total	sulfur dioxide	density	pH	sulphates	alc
0	0	7.4	0.70	0.001.9	0.076	11.000	34.0	0.9978	3.5100	0.56	9.40	
1	1	7.8	0.88	0.00	2.600	0.098	25.0	67.0000	0.9968	3.20	0.68	
2	2	7.8	0.76	0.04	2.300	0.092	15.0	54.0000	0.9970	3.26	0.65	
3	3	11.2	0.28	0.56	1.900	0.075	17.0	60.0000	0.9980	3.16	0.58	
4	4	7.4	0.70	0.00	1.900	0.076	11.0	34.0000	0.9978	3.51	0.56	

In [113]:
```
import seaborn as sns
```

In [142]:
```
data = np.random.multivariate_normal([0,0],[[5,2],[2,2]],size = 100)

df = pd.DataFrame(data,columns=['x','y'])
df
```

Out[142]:

	x	y
0	-2.384315	-1.352998
1	3.904605	2.492092
2	3.379499	-0.115659
3	5.509544	0.847455
4	1.210252	2.920340
...	...	...
95	-0.351398	-1.369710
96	-2.589043	0.492035
97	1.154548	0.784164
98	0.010075	0.038262
99	-1.232003	0.405225

100 rows × 2 columns

In [143]:
```
df=pd.DataFrame(data,columns=['x','y'])
df
```

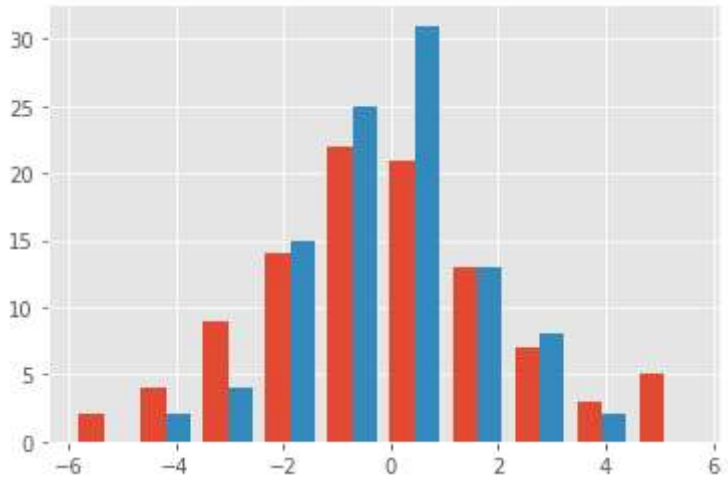
Out[143]:

	x	y
0	-2.384315	-1.352998
1	3.904605	2.492092
2	3.379499	-0.115659
3	5.509544	0.847455
4	1.210252	2.920340
...	...	...
95	-0.351398	-1.369710
96	-2.589043	0.492035
97	1.154548	0.784164
98	0.010075	0.038262
99	-1.232003	0.405225

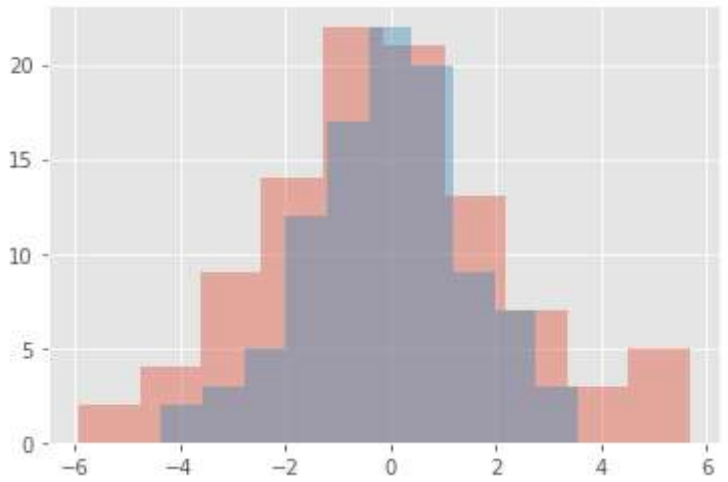
100 rows × 2 columns

In [144]:
```
plt.hist(df)
```

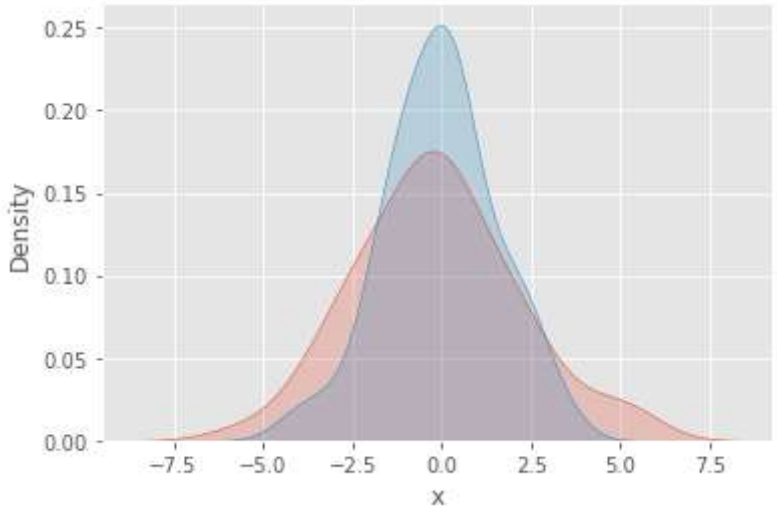
Out[144]:
```
(array([[ 2.,  4.,  9., 14., 22., 21., 13.,  7.,  3.,  5.],
        [ 0.,  2.,  4., 15., 25., 31., 13.,  8.,  2.,  0.]]),
 array([-5.92665084, -4.76715767, -3.6076645 , -2.44817133, -1.28867816,
        -0.12918499,  1.03030818,  2.18980135,  3.34929452,  4.50878769,
         5.66828086]),
 <a list of 2 BarContainer objects>)
```



In [145]:
```
for col in 'xy':
    plt.hist(df[col],alpha=0.4)
```

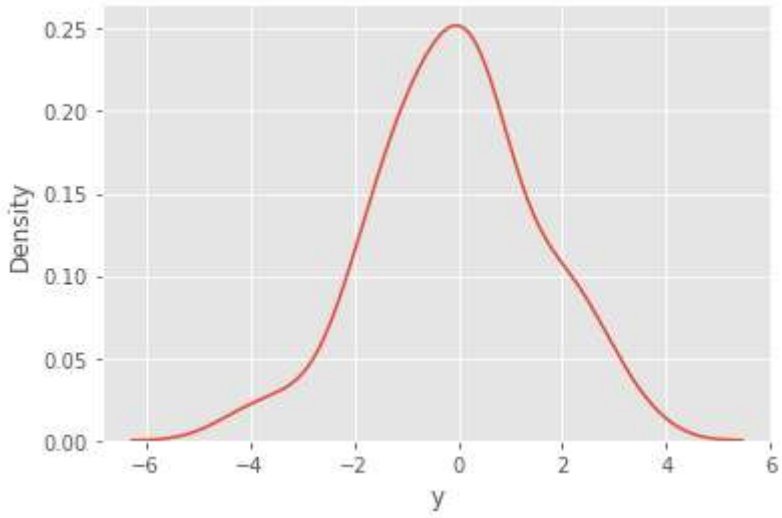


In [146]:
```
#kde means kernal density estimattion
#represents the probability distribution of data points
for col in 'xy':
    sns.kdeplot(df[col],shade=True)
```



In [147]:
```
sns.kdeplot(df[col])
```

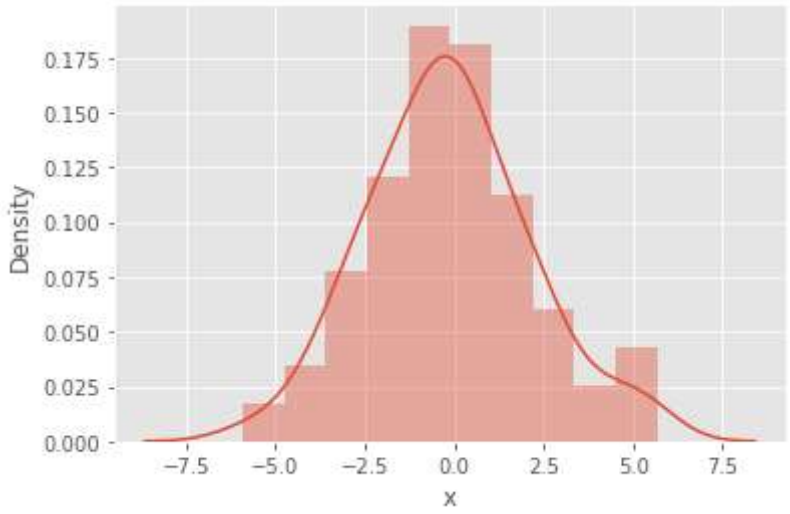
Out[147]:
```
<AxesSubplot:xlabel='y', ylabel='Density'>
```



In [148]:
```
sns.distplot(df['x'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

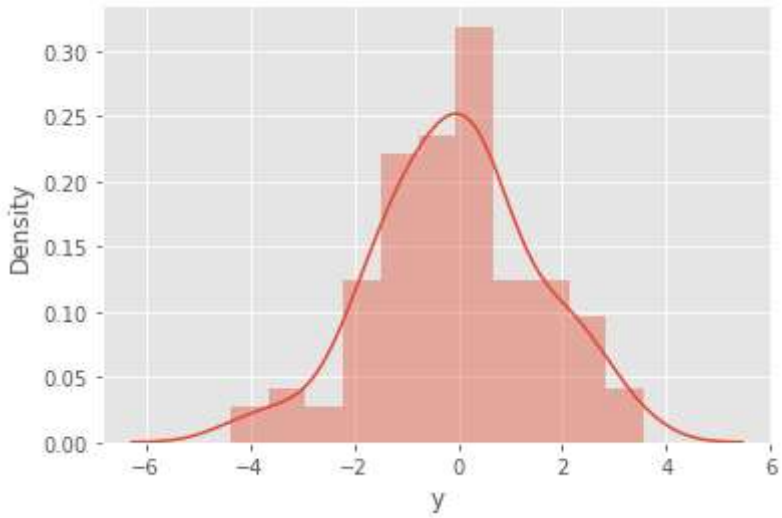
Out[148]:
```
<AxesSubplot:xlabel='x', ylabel='Density'>
```



In [150]:
```
sns.distplot(df['y'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[150]:
```
<AxesSubplot:xlabel='y', ylabel='Density'>
```



In [122]:
```
iris = sns.load_dataset('iris')

iris
```

Out[122]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

pair plots:

- visulize the correlation among the data points having large dimensions

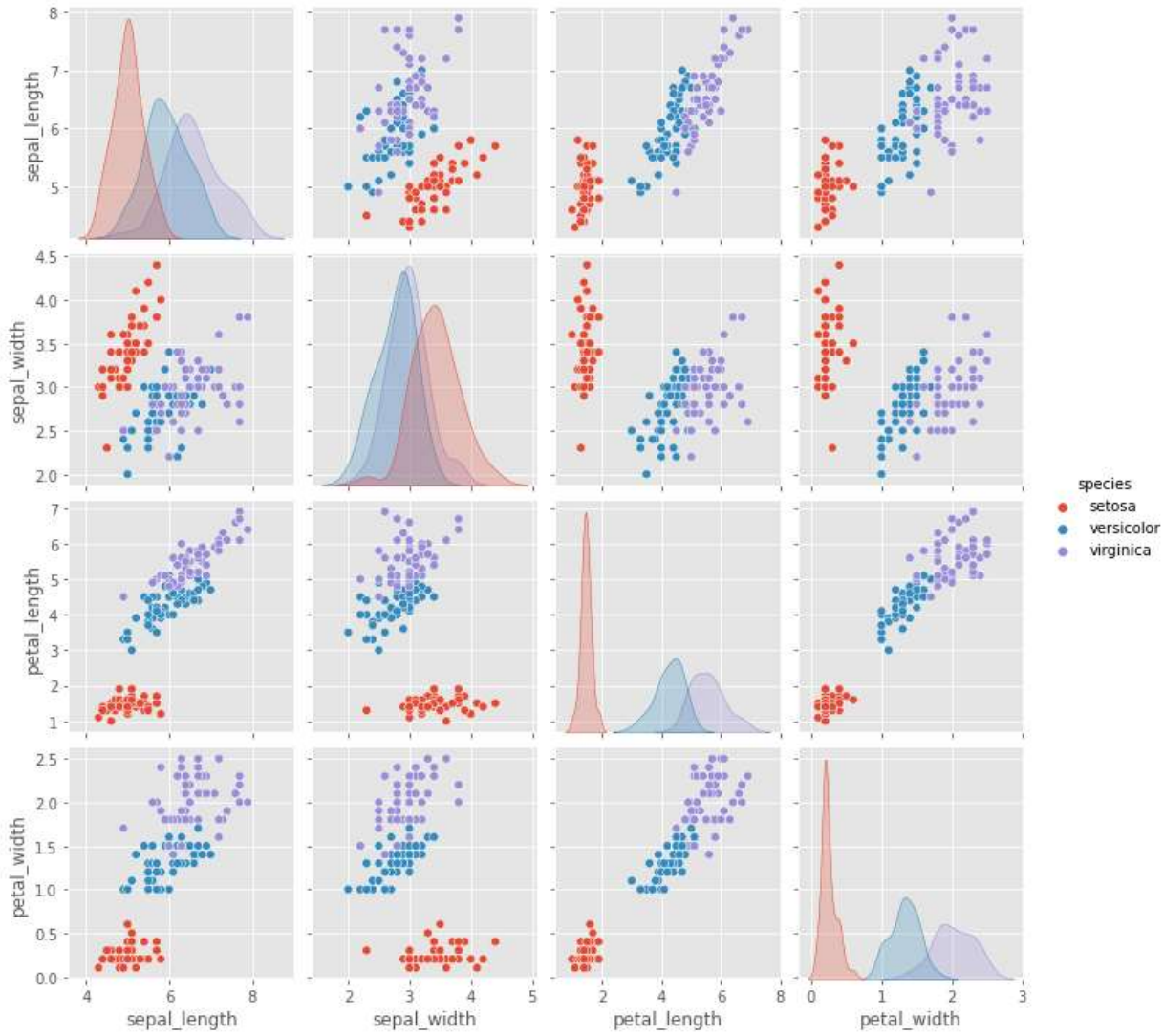
In [123]:
```
iris.head()
```

Out[123]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [125]:
```
sns.pairplot(iris,hue='species', height = 2.5)
```

Out[125]:
```
<seaborn.axisgrid.PairGrid at 0x291305b8580>
```

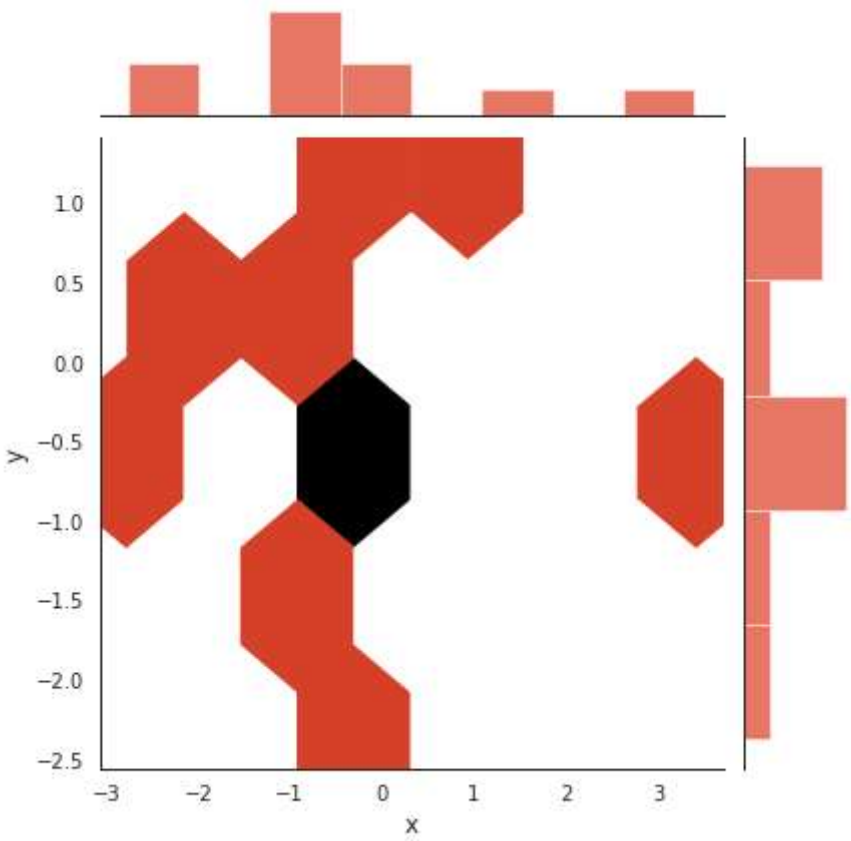


joint plot:

- tells the joint distribution of data points

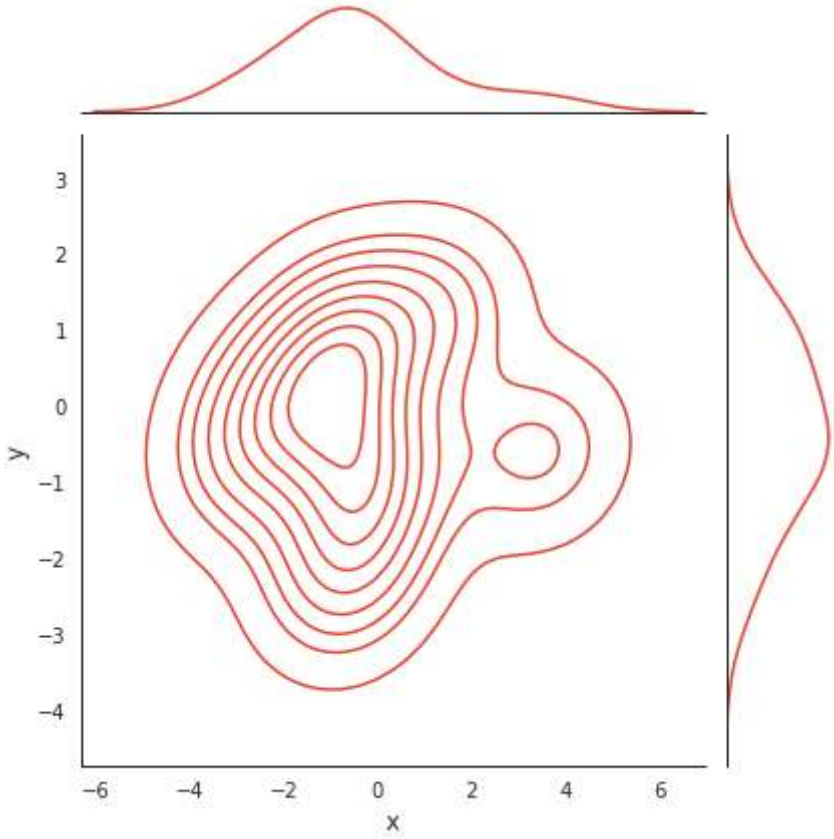
In [126]:
```
with sns.axes_style('white'):
    sns.jointplot('x','y',df, kind = 'hex')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y, data. From version 0.
12, the only valid positional argument will be `data`, and passing other argume
nts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



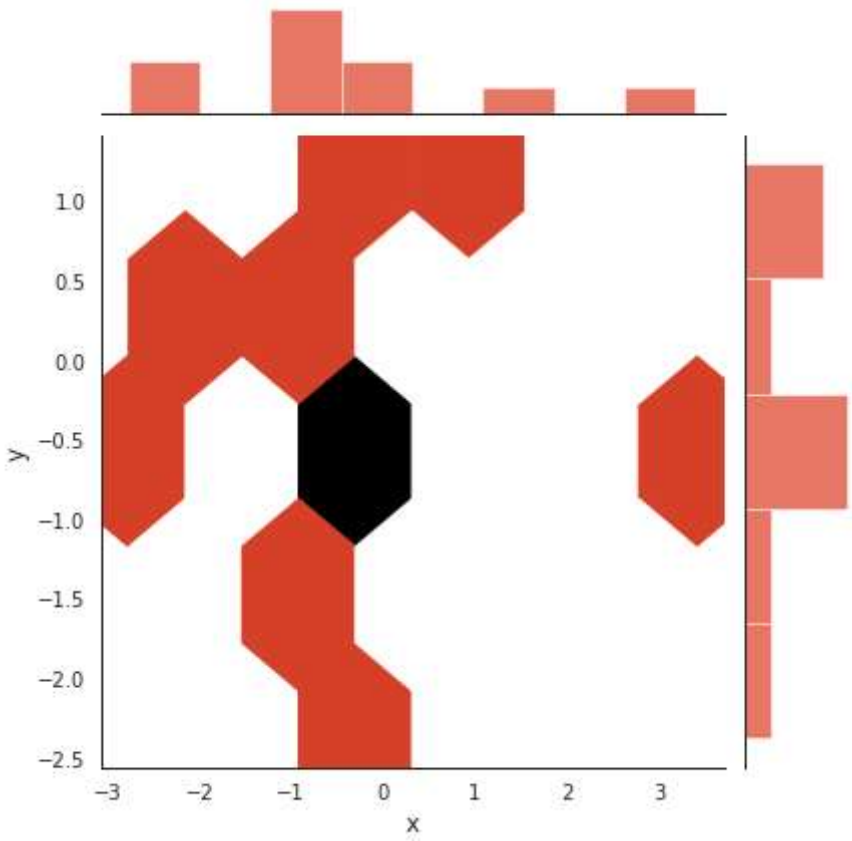
In [127]:
```
with sns.axes_style('white'):
    sns.jointplot('x','y',df, kind = 'kde')
```

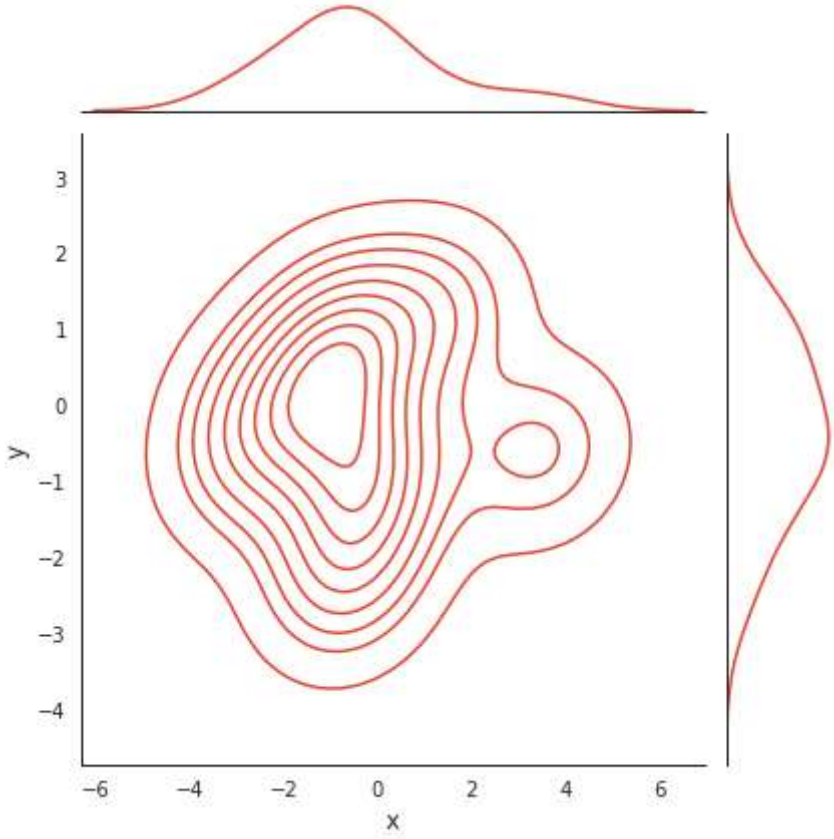
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y, data. From version 0.
12, the only valid positional argument will be `data`, and passing other argume
nts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [128]:
```
with sns.axes_style('white'):
    sns.jointplot('x','y',df, kind = 'hex')


with sns.axes_style('white'):
    sns.jointplot('x','y',df, kind = 'kde')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y, data. From version 0.
12, the only valid positional argument will be `data`, and passing other argume
nts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWar
ning: Pass the following variables as keyword args: x, y, data. From version 0.
12, the only valid positional argument will be `data`, and passing other argume
nts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```





In [ ]: