Python basic programs

[1]:
```
import numpy as np
arr=np.array([1,2,3,4])
print(arr)
```

```
[1 2 3 4]
```

[2]:
```
arr=np.array([1,2,3,4])
print(arr+10)
print(arr*2)
```

```
[11 12 13 14]
[2 4 6 8]
```

[5]:
```
print(np.mean(arr))
print(np.max(arr))
print(np.min(arr))
```

```
2.5
4
1
```

[6]:
```
import pandas as pd
data={"Name":["Arun","priya","kumar"],"Age":[22,21,25],"City":
  ↪['colombo','kandy','Galle']}
df=pd.DataFrame(data)
print(df)
```

```
    Name  Age     City
0   Arun   22  colombo
1  priya   21    kandy
2  kumar   25    Galle
```

[9]:
```
print(df['Name'])
print(df.loc[0])
print(df.iloc[1])
```

```
0     Arun
1    priya
```

```
2    kumar
Name: Name, dtype: object
Name       Arun
Age          22
City    colombo
Name: 0, dtype: object
Name    priya
Age        21
City    kandy
Name: 1, dtype: object
```

[13]:
```
print(df.describe())
print(df['Age'].mean())
print(df['City'].unique())
```

```
             Age
count   3.000000
mean   22.666667
std     2.081666
min    21.000000
25%    21.500000
50%    22.000000
75%    23.500000
max    25.000000
22.666666666666668
['colombo' 'kandy' 'Galle']
```

[14]:
```
import pandas as pd
data={"Name":["Arun","priya","kumar"],"Age":[22,21,25],"City":
  ↪['colombo','kandy','Galle']}
df=pd.DataFrame(data)
print(df)
```

```
    Name  Age     City
0   Arun   22  colombo
1  priya   21    kandy
2  kumar   25    Galle
```

[16]:
```
print(df['Name'])
print(df.loc[0])
print(df.iloc[1])
```

```
0     Arun
1    priya
2    kumar
Name: Name, dtype: object
Name       Arun
Age          22
```

```
City    colombo
Name: 0, dtype: object
Name    priya
Age        21
City    kandy
Name: 1, dtype: object
```

[17]:
```
df[df['Age']>22]
```

[17]:
```
    Name  Age   City
2  kumar   25  Galle
```

[19]:
```
df["Age_plus_1"]=df['Age']+1
```

[21]:
```
print(df['Age_plus_1'])
```

```
0    23
1    22
2    26
Name: Age_plus_1, dtype: int64
```

[22]:
```
df.sort_values(by="Age",ascending=False)
```

[22]:
```
    Name  Age     City  Age_plus_1
2  kumar   25    Galle          26
0   Arun   22  colombo          23
1  priya   21    kandy          22
```

[23]:
```
df.fillna(0)
df.dropna()
```

[23]:
```
    Name  Age     City  Age_plus_1
0   Arun   22  colombo          23
1  priya   21    kandy          22
2  kumar   25    Galle          26
```

[24]:
```
df=pd.read_csv("data.csv")
print(df)
```

```
             Name   Age  Grade     Subject
0        John Doe    15      A        Math
1      Jane Smith    16      B     Science
2  Michael Brown    14     A-     English
3  Sarah Johnson    17     C+     History
4       Chris Lee    16      B   Geography
5     Emily Davis    15      A         Art
```

[25]:
```
df.to_csv("output.csv",index=False)
```

[26]:
```
df=pd.read_excel("data.xlsx")
print(df.head())
```

```
    Name  Age     City
0   Arun   22  colombo
1  Priya   21    kandy
2  Kumar   25    galle
```

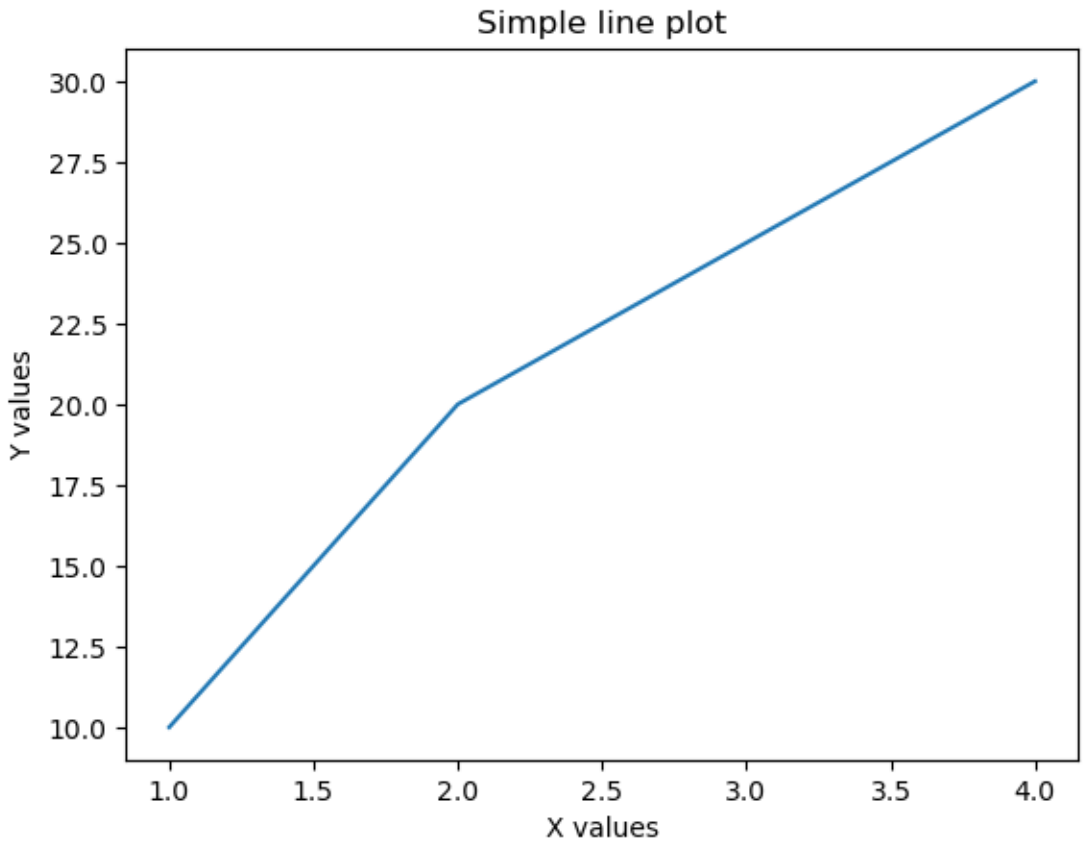
[27]:
```
df.to_excel("output.xlsx",index=False)
```

[28]:
```
df1=pd.read_excel("output.xlsx")
print(df1)
```

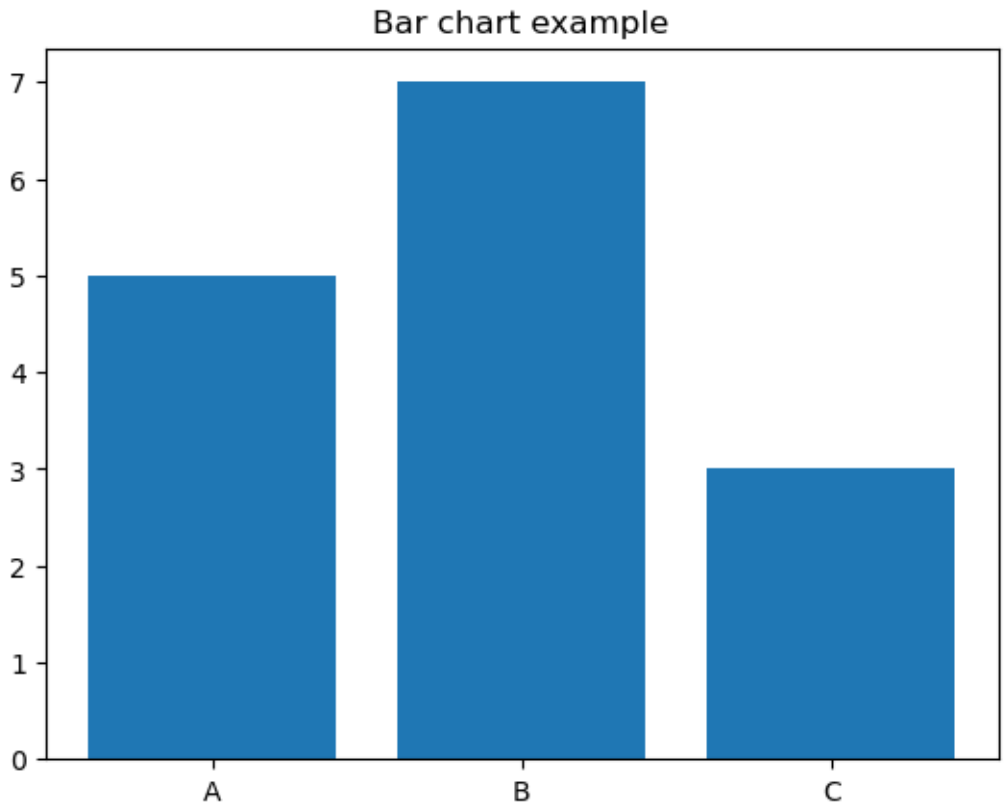
```
    Name  Age     City
0   Arun   22  colombo
1  Priya   21    kandy
2  Kumar   25    galle
```

[29]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

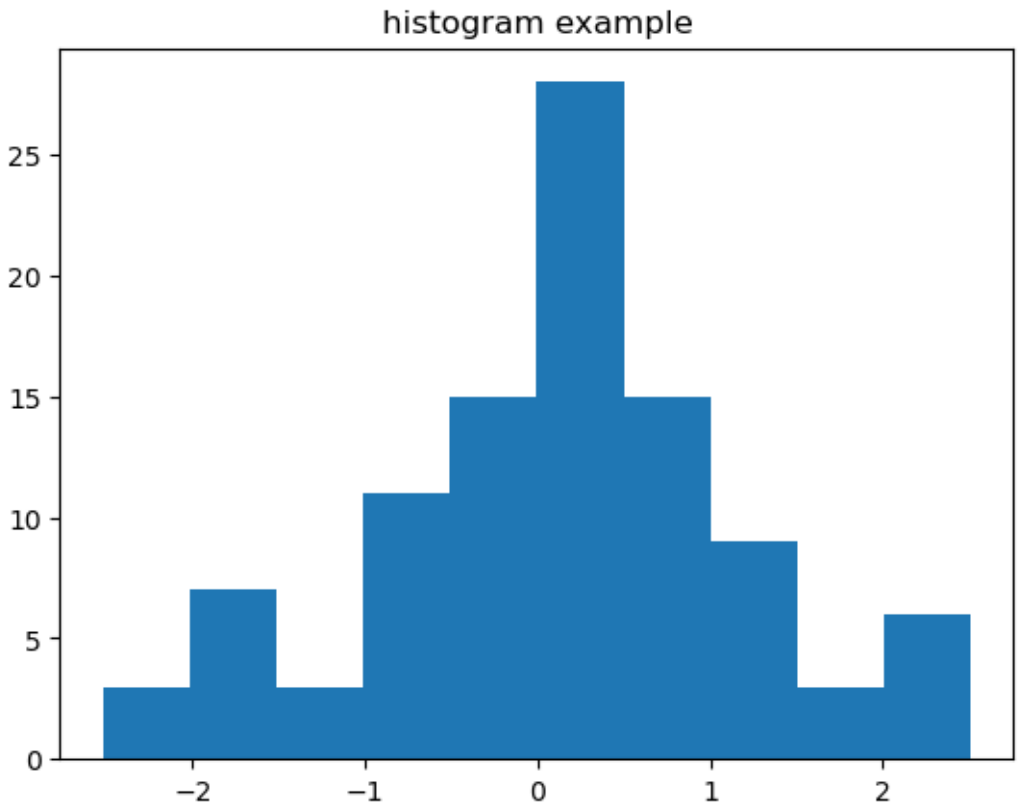
[30]:
```
x=[1,2,3,4]
y=[10,20,25,30]
plt.plot(x,y)
plt.xlabel("X values")
plt.ylabel("Y values")
plt.title("Simple line plot")
plt.show()
```



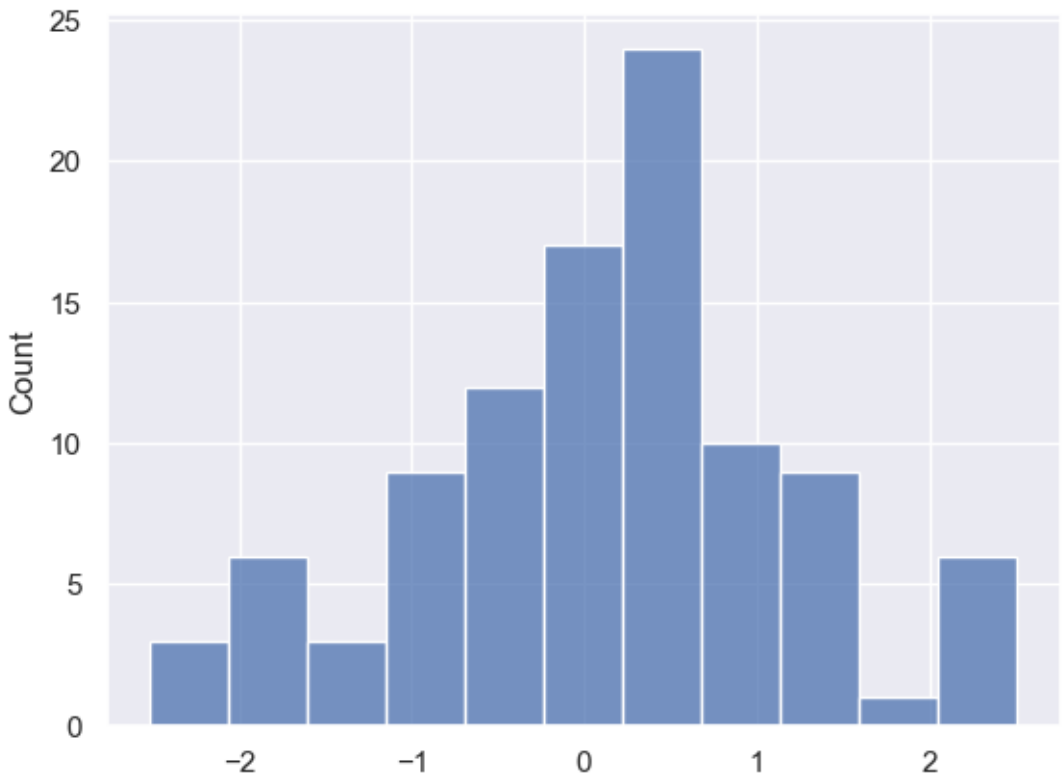
[31]:
```
names=["A","B","C"]
values=[5,7,3]
plt.bar(names,values)
plt.title("Bar chart example")
plt.show()
```



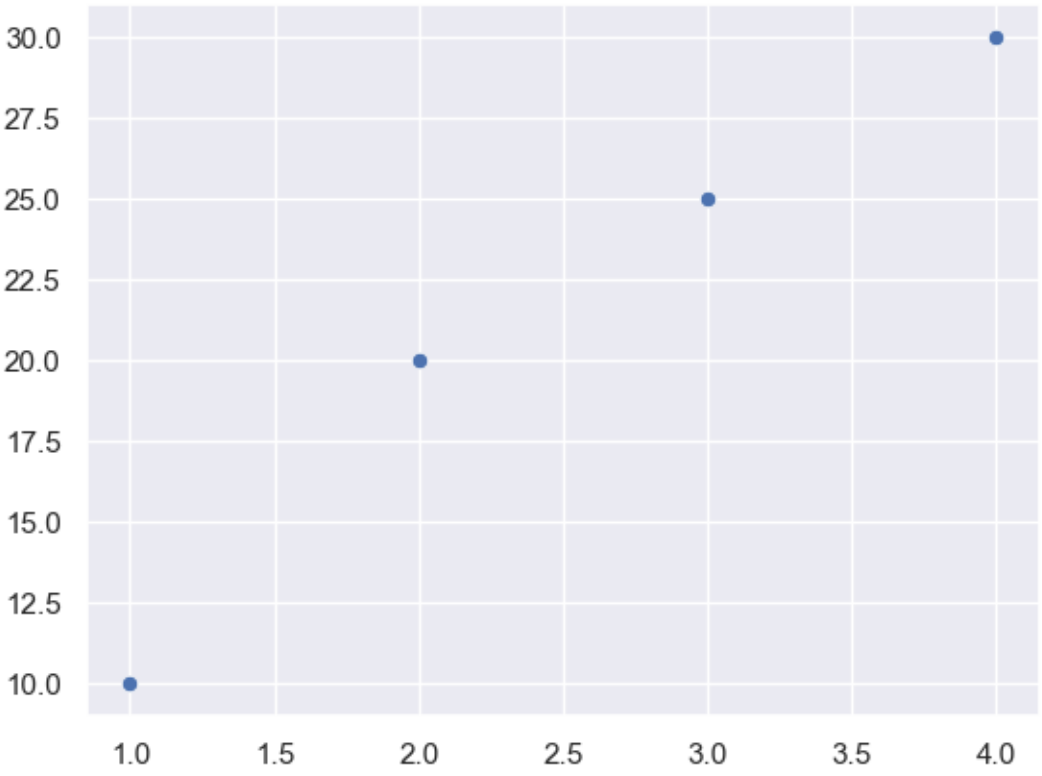
[32]:
```
data=np.random.randn(100)
plt.hist(data)
plt.title("histogram example")
plt.show()
```



[33]:
```
sns.set_theme()
sns.histplot(data)
plt.show()
```



[34]:
```
sns.scatterplot(x=x,y=y)
plt.show()
```


30.0
27.5
25.0
22.5
20.0
17.5
15.0
12.5
10.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0

[ ]:

In [52]:
```
np.array([7,2,8,4],dtype=np.float32)
```

Out[52]:
```
array([7., 2., 8., 4.], dtype=float32)
```

In [53]:
```
np.array([7,2.0,8,4])
```

Out[53]:
```
array([7., 2., 8., 4.])
```

In [55]:
```
np.array([[1,2,3,4],[5,6,7,8]])
```

Out[55]:
```
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```

In [56]:
```
np.zeros(5)
```

Out[56]:
```
array([0., 0., 0., 0., 0.])
```

In [58]:
```
np.zeros((2,3))
```

Out[58]:
```
array([[0., 0., 0.],
       [0., 0., 0.]])
```

In [59]:
```
np.ones(5,dtype=np.int32)
```

Out[59]:
```
array([1, 1, 1, 1, 1])
```

In [60]:
```
np.random.rand(2,3)
```

Out[60]:
```
array([[0.81973961, 0.41145436, 0.04286307],
       [0.39904304, 0.64622033, 0.31561125]])
```

In [61]:
```
np.full((2,2),7)
```

Out[61]:
```
array([[7, 7],
       [7, 7]])
```

In [62]:
```
np.eye(3)
```

Out[62]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

In [63]:
```
np.eye(3,k=1)
```

Out[63]:
```
array([[0., 1., 0.],
       [0., 0., 1.],
       [0., 0., 0.]])
```

In [64]:
```
np.eye(3,k=-2)
```

Out[64]:
```
array([[0., 0., 0.],
       [0., 0., 0.],
       [1., 0., 0.]])
```

In [65]:
```
np.arange(5)
```

Out[65]:
```
array([0, 1, 2, 3, 4])
```

In [66]:
```
np.arange(2,10,2)
```

Out[66]:
```
array([2, 4, 6, 8])
```

In [68]:
```
np.linspace(0,1,5)
```

Out[68]:
```
array([0.  , 0.25, 0.5 , 0.75, 1.  ])
```

In [69]:
```
a=np.array([[5,10,15],[20,25,30]])
print("Array:","\n",a.ndim)
```

```
Array:
 2
```

In [71]:
```
a=np.array([[1,2,3],[4,5,6]])
print("Array:","\n",a)
print("Shape","\n",a.shape)
print("Rows=",a.shape[0])
print("Columns =",a.shape[1])
```

```
Array:
 [[1 2 3]
 [4 5 6]]
Shape
 (2, 3)
Rows= 2
Columns = 3
```

In [73]:
```
a=np.array([[5,10,15],[20,25,20]])
print("Size of the array:",a.size)
print("Manual determination of size of the array:",a.shape[0]*a.shape[1])
```

```
Size of the array: 6
Manual determination of size of the array: 6
```

In [74]:
```
a=np.array([3,6,9,12,18,24])
print("Three rows:","\n",np.reshape(a,(3,-1)))
```

```
Three rows:
 [[ 3  6]
 [ 9 12]
 [18 24]]
```

In [75]:
```
print("Three columns:","\n",np.reshape(a,(-1,3)))
```

```
Three columns:
 [[ 3  6  9]
 [12 18 24]]
```

In [88]:
```
a=np.ones((2,2))
b=a.flatten()
c=a.ravel()
```

In [89]:
```
print('Original Shape:',a.shape)
print("Array:","\n",a)
print("Shape after flatten:",b.shape)
print("Array:","\n",b)
print("Shape after ravel:",c.shape)
print("Array:","\n",c)
```

```
Original Shape: (2, 2)
Array:
 [[1. 1.]
 [1. 1.]]
Shape after flatten: (4,)
Array:
 [1. 1. 1. 1.]
Shape after ravel: (4,)
Array:
 [1. 1. 1. 1.]
```

In [90]:
```
b[0]=0
print(a)
```

```
[[1. 1.]
 [1. 1.]]
```

In [91]:
```
c[0]=0
print(a)
```

```
[[0. 1.]
 [1. 1.]]
```

In [92]:
```
a=np.array([[1,2,3],[4,5,6]])
b=np.transpose(a)
```

In [95]:
```
print("Original ","\n","Shape",a.shape,"\n",a)
print("Expand along columns: ","\n","Shape",b.shape,"\n",b)
```

```
Original
 Shape (2, 3)
 [[1 2 3]
 [4 5 6]]
Expand along columns:
 Shape (3, 2)
 [[1 4]
 [2 5]
 [3 6]]
```

In [97]:
```
a=np.array([1,2,3,4,5,6])
print(a[1:5:2])
```

```
[2 4]
```

In [98]:
```
a=np.array([1,2,3,4,5,6])
print(a[1:6:2])
```

```
[2 4 6]
```

In [99]:
```
a=np.array([1,2,3,4,5,6])
print(a[:6:2])
print(a[1::2])
print(a[1:6:1])
```

```
[1 3 5]
[2 4 6]
[2 3 4 5 6]
```

In [103]:
```
a=np.array([[1,2,3],[4,5,6]])
print(a[0,0])
print(a[1,2])
print(a[1,0])
```

```
[[1 2 3]
 [4 5 6]]
6
4
```

In [104]:
```
a=np.array([[1,2,3],[4,5,6]])
print('First row values',"\n",a[0:1,:])
print("Alternate values from first row:","\n",a[0:1:,::2])
```

```
First row values
 [[1 2 3]]
Alternate values from first row:
 [[1 3]]
```

In [113]:
```
a=np.array([[[1,2],[3,4]],[[5,6],[7,8]]])
```

In [114]:
```
print(a)
```

```
[[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]
```

In [118]:
```
print("First array , first row, first column values:","\n",a[0,0,0])
print("First array , last column:","\n",a[0,:,1])
print("First two rows for second and third arrays :",a[1:,0:2,0:2])
print("Printing as a single array:","\n",a[1:,0:2,0:2].flatten())
```

```
First array , first row, first column values:
 1
First array , last column:
 [2 4]
First two rows for second and third arrays : [[[5 6]
  [7 8]]]
Printing as a single array:
 [5 6 7 8]
```

In [121]:
```
a=np.array([[1,2,3,4,5],[6,7,8,9,10]])
print(a[:,-1])
print(a[:,-1:-3:-1])
```

```
[ 5 10]
[[ 5  4]
 [10  9]]
```

In [122]:
```
a=np.array([[1,2,3,4,5],[6,7,8,9,10]])
print("Original array:","\n",a)
print("Reversed array :","\n",a[::-1,::-1])
```

```
Original array:
 [[ 1  2  3  4  5]
 [ 6  7  8  9 10]]
Reversed array :
 [[10  9  8  7  6]
 [ 5  4  3  2  1]]
```

In [123]:
```
a=np.array([[1,2,3,4,5],[6,7,8,9,10]])
print("Original Array","\n",a)
print("Reversed array vertically:","\n",np.flip(a,axis=1))
print("Reversed array horizantally:","\n",np.flip(a,axis=0))
```

```
Original Array
 [[ 1  2  3  4  5]
 [ 6  7  8  9 10]]
Reversed array vertically:
 [[ 5  4  3  2  1]
 [10  9  8  7  6]]
Reversed array horizantally:
 [[ 6  7  8  9 10]
 [ 1  2  3  4  5]]
```

In [ ]:
```
```

Pandas

[1]:
```
import pandas as pd
data_england={"Name":["Kane","Sterling","Saka","Marguire"],"Age":[27,26,19,18]}
data_italy={"Name":["Immobile","Insigne","Chiellini","Cheisa"],"Age":
 ↪[31,30,36,23]}
df_england=pd.DataFrame(data_england)
df_england
```

[1]:

	Name	Age
0	Kane	27
1	Sterling	26
2	Saka	19
3	Marguire	18

[2]:
```
df_italy=pd.DataFrame(data_italy)
df_italy
```

[2]:

	Name	Age
0	Immobile	31
1	Insigne	30
2	Chiellini	36
3	Cheisa	23

[3]:
```
import pandas as pd
frames=[df_england,df_italy]
both_teams=pd.concat(frames)
both_teams
```

[3]:

	Name	Age
0	Kane	27
1	Sterling	26
2	Saka	19
3	Marguire	18
0	Immobile	31
1	Insigne	30
2	Chiellini	36
3	Cheisa	23

[4]:
```
df_england.append(df_italy)
pd.concat(frames,keys=["England","Italy"])
```

```
C:\Users\y23acs482\AppData\Local\Temp\ipykernel_16916\241983346.py:1:
FutureWarning: The frame.append method is deprecated and will be removed from
pandas in a future version. Use pandas.concat instead.
  df_england.append(df_italy)
```

[4]:

		Name	Age
England	0	Kane	27
	1	Sterling	26
	2	Saka	19
	3	Marguire	18
Italy	0	Immobile	31
	1	Insigne	30
	2	Chiellini	36
	3	Cheisa	23

[5]:
```
both_teams[both_teams["Age"]>=30]
```

[5]:

	Name	Age
0	Immobile	31
1	Insigne	30
2	Chiellini	36

[6]:
```
both_teams[both_teams["Name"].str.startswith('S')]
```

[6]:

	Name	Age
1	Sterling	26
2	Saka	19

[7]:
```
clubs=["Tottenham","Man city","Arsenal","ManUtd"]
df_england["Associated Clubs"]=clubs
df_england
```

[7]:

	Name	Age	Associated Clubs
0	Kane	27	Tottenham
1	Sterling	26	Man city
2	Saka	19	Arsenal
3	Marguire	18	ManUtd

[8]:
```
frames=[df_england,df_italy]
both_teams=pd.concat(frames,sort=False)
both_teams
```

[8]:

	Name	Age	Associated Clubs
0	Kane	27	Tottenham

```
1   Sterling   26         Man city
2       Saka   19          Arsenal
3   Marguire   18           ManUtd
0   Immobile   31              NaN
1    Insigne   30              NaN
2  Chiellini   36              NaN
3     Cheisa   23              NaN
```

[9]:
```
with open('ML.txt','r')as f:
    print(f.read())
```

```
computer science and engineering in bapatla engineering college
```

[10]:
```
with open('ML.txt','r')as f:
    print(f.read(10))
```

```
computer s
```

[11]:
```
with open('ML.txt','r')as f:
    print(f.readline())
```

```
computer science and engineering in bapatla engineering college
```

[12]:
```
with open('ML.txt','r')as f:
    print(f.readlines())
```

```
['computer science and engineering in bapatla engineering college']
```

[13]:
```
import pandas as pd
df=pd.read_csv("csv file.csv")
df
```

[13]:
```
Empty DataFrame
Columns: [Bapatla Engineering College]
Index: []
```

[14]:
```
import pandas as pd
df=pd.read_excel("products.xlsx")
df
```

[14]:
```
   1       Pen  10
0  2    Pencil   5
1  3    Eraser   2
2  4  Notebook  40
3  5   Stapler  50
```

[15]:
```
import pandas as pd
calories=[420,380,390]
duration=[50,40,45]
myvar1=pd.Series(calories)
print(myvar1)
```

```
0    420
1    380
2    390
dtype: int64
```

[16]:
```
myvar2=pd.Series(duration)
print(myvar2)
```

```
0    50
1    40
2    45
dtype: int64
```

[18]:
```
import pandas as pd
data={"Id":[1,2,3,4,5],
      "Product":["Pen","Pencil","Eraser","Notebook","Stapler"],
      "Price":[10,5,2,40,50]
     }
print(data)
```

```
   Id   Product  Price
0   1       Pen     10
1   2    Pencil      5
2   3    Eraser      2
3   4  Notebook     40
4   5   Stapler     50
```

[77]:
```
import pandas as pd
myvar=pd.DataFrame(data)
print(myvar)
```

```
   Id   Product  Price
0   1       Pen     10
1   2    Pencil      5
2   3    Eraser      2
3   4  Notebook     40
4   5   Stapler     50
```

[20]:
```
import pandas as pd
print(f"pandas version:{pd.__version__}")
```

```
pandas version:1.5.3
```

[21]:
```
cars=pd.Series(["BMW","Toyota","Honda"])
cars
colours=pd.Series(["Blue","Red","White"])
colours
```

[21]:
```
0     Blue
1      Red
2    White
dtype: object
```

[22]:
```
car_data=pd.DataFrame({"Car type":cars,"colour":colours})
car_data
```

[22]:
```
  Car type colour
0      BMW   Blue
1   Toyota    Red
2    Honda  White
```

[23]:
```
car_data=pd.DataFrame({"Car type":cars,"colour":colours})
car_data
```

[23]:
```
  Car type colour
0      BMW   Blue
1   Toyota    Red
2    Honda  White
```

[60]:
```
car_sales=pd.read_csv("car_sales.csv")
car_sales
```

[60]:
```
       car   sales
0      BMW  100000
1    Honda  300000
2   Toyota  120000
```

[61]:
```
df=pd.read_csv("car_sales.csv")
df
```

[61]:
```
       car   sales
0      BMW  100000
1    Honda  300000
2   Toyota  120000
```

[62]:
```
car_sales.dtypes
```

[62]:
```
 car     object
sales     int64
dtype: object
```

[63]:
```
car_sales.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0    car    3 non-null      object
 1   sales   3 non-null      int64 
dtypes: int64(1), object(1)
memory usage: 176.0+ bytes
```

[64]:
```
car_sales.describe()
```

[64]:
```
              sales
count       3.000000
mean   173333.333333
std    110151.410946
min    100000.000000
25%    110000.000000
50%    120000.000000
75%    210000.000000
max    300000.000000
```

[65]:
```
car_sales.mean(numeric_only=True)
```

[65]:
```
sales    173333.333333
dtype: float64
```

[66]:
```
car_sales.sum(numeric_only=True)
```

[66]:
```
sales    520000
dtype: int64
```

[67]:
```
car_sales.index
```

[67]:
```
RangeIndex(start=0, stop=3, step=1)
```

[68]:
```
len(car_sales)
```

[68]:
```
3
```

[69]:
```
car_sales.head()
```

[69]:
```
       car   sales
0      BMW  100000
1    Honda  300000
```

```
2   Toyota  120000
```

[70]:
```
car_sales.tail()
```

[70]:
```
       car   sales
0      BMW  100000
1    Honda  300000
2   Toyota  120000
```

[71]:
```
car_sales.loc[2]
```

[71]:
```
 car      Toyota
sales     120000
Name: 2, dtype: object
```

[72]:
```
car_sales.iloc[2]
```

[72]:
```
 car      Toyota
sales     120000
Name: 2, dtype: object
```

[73]:
```
car_sales.loc[:2]
```

[73]:
```
       car   sales
0      BMW  100000
1    Honda  300000
2   Toyota  120000
```

[74]:
```
car_sales["sales"] = car_sales["sales"].replace({'\$', ''}, regex=True)
car_sales
```

[74]:
```
       car   sales
0      BMW  100000
1    Honda  300000
2   Toyota  120000
```

[75]:
```
seats_column=pd.Series([5,5,5,5,5,5,5,5,5,5])
car_sales["seats"]=seats_column
car_sales
```

[75]:
```
       car   sales  seats
0      BMW  100000      5
1    Honda  300000      5
2   Toyota  120000      5
```

[ ]:
```

```

matplotlib

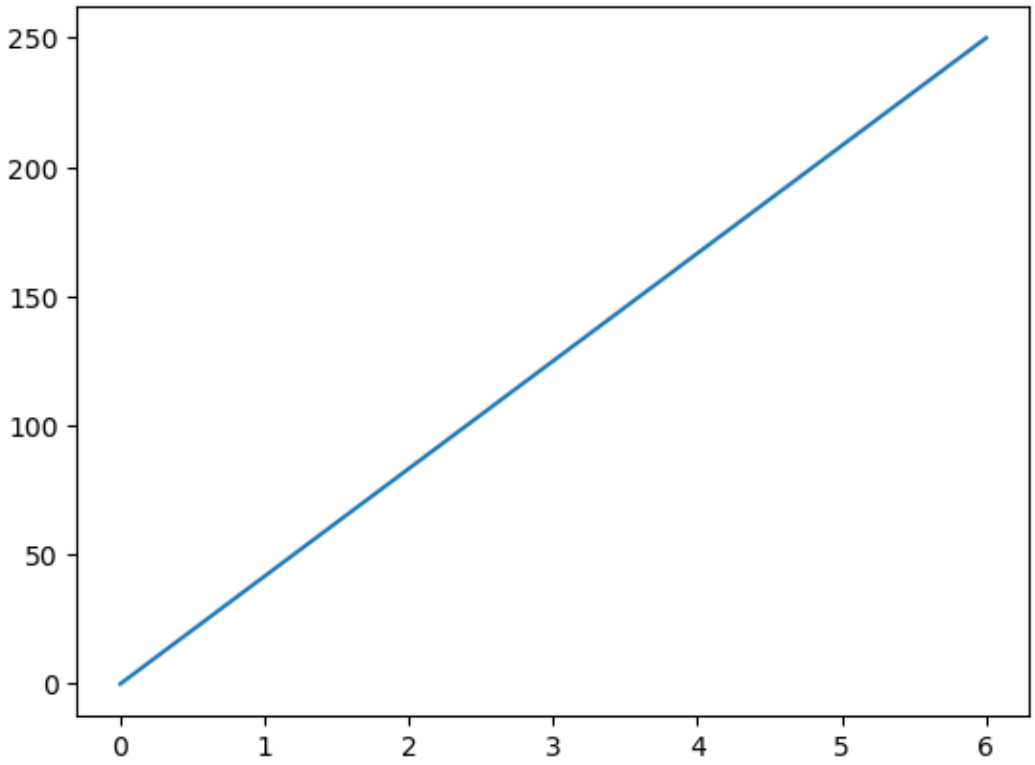
[1]:
```
#importing matplotlib
import matplotlib
```

[2]:
```
#checking version
print(matplotlib.__version__)
```

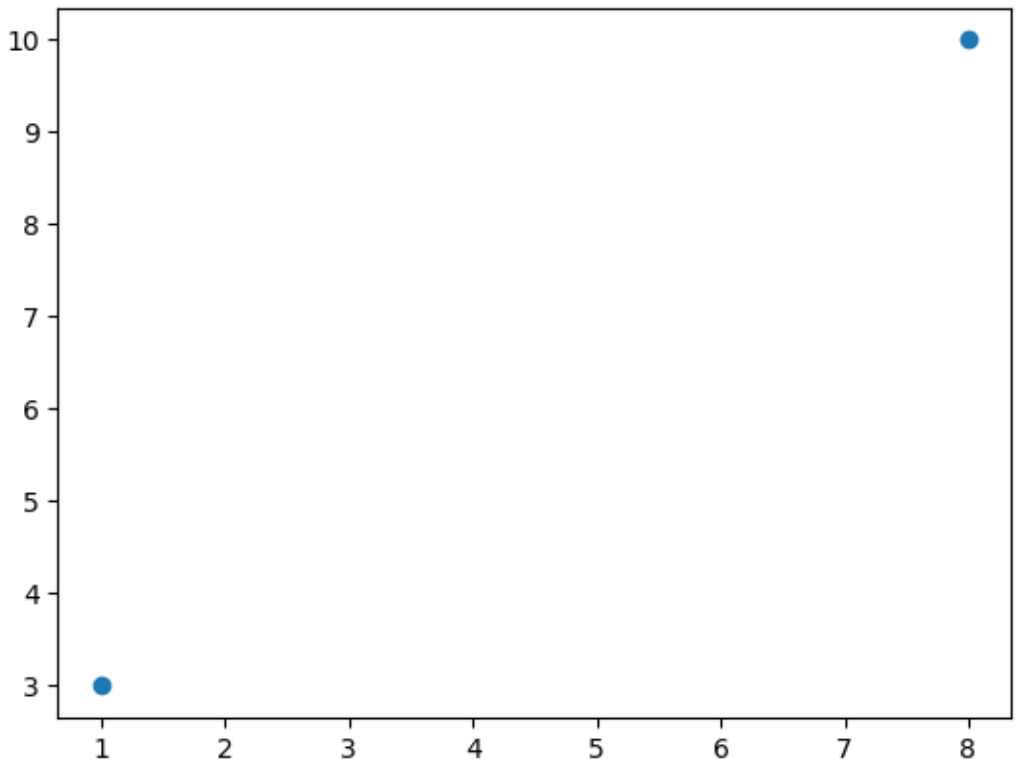
```
3.7.0
```

[3]:
```
#importing pyplot
import matplotlib.pyplot as plt
```

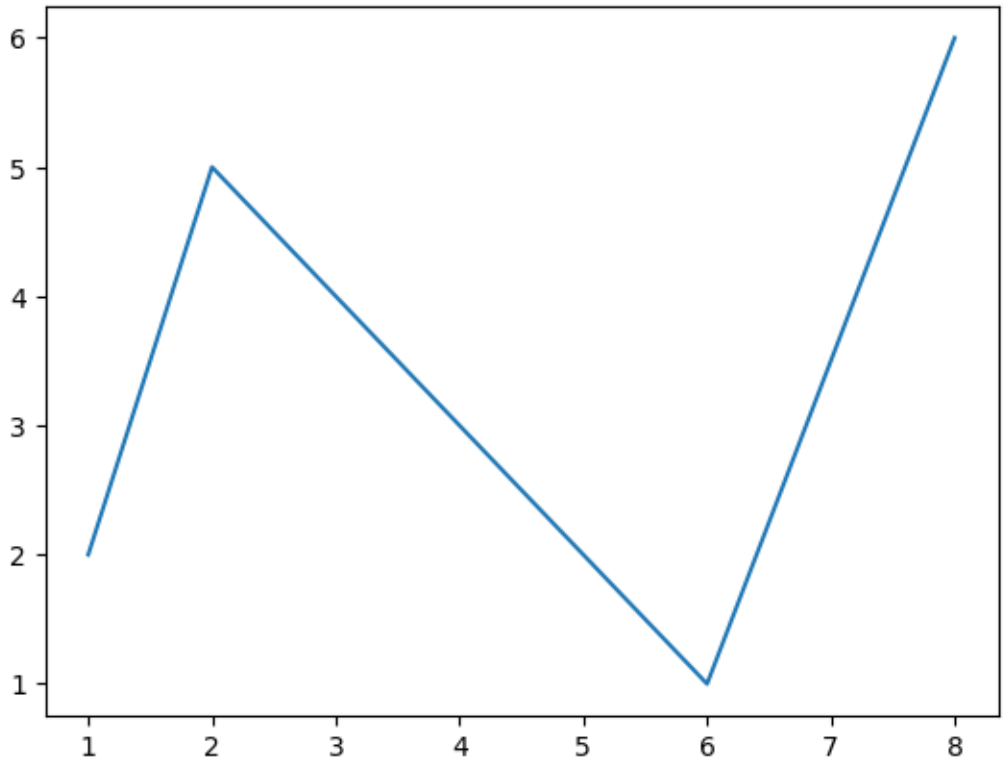
[5]:
```
#draw a line in diagram from position(0,0) to position(6,250)
import numpy as np
x=np.array([0,6])
y=np.array([0,250])
plt.plot(x,y)
plt.show()
```



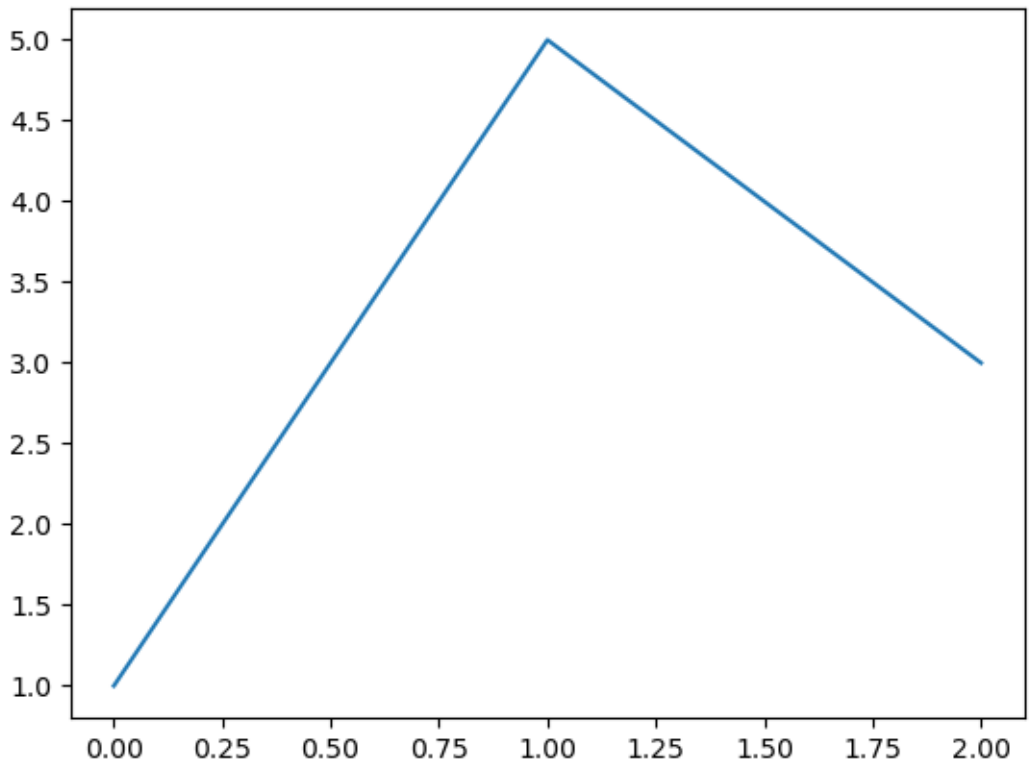
[7]:
```
#Draw two points in the diagram at (1,3) and (8,10)
import numpy as np
x=np.array([1,8])
y=np.array([3,10])
plt.plot(x,y,'o')
plt.show()
```



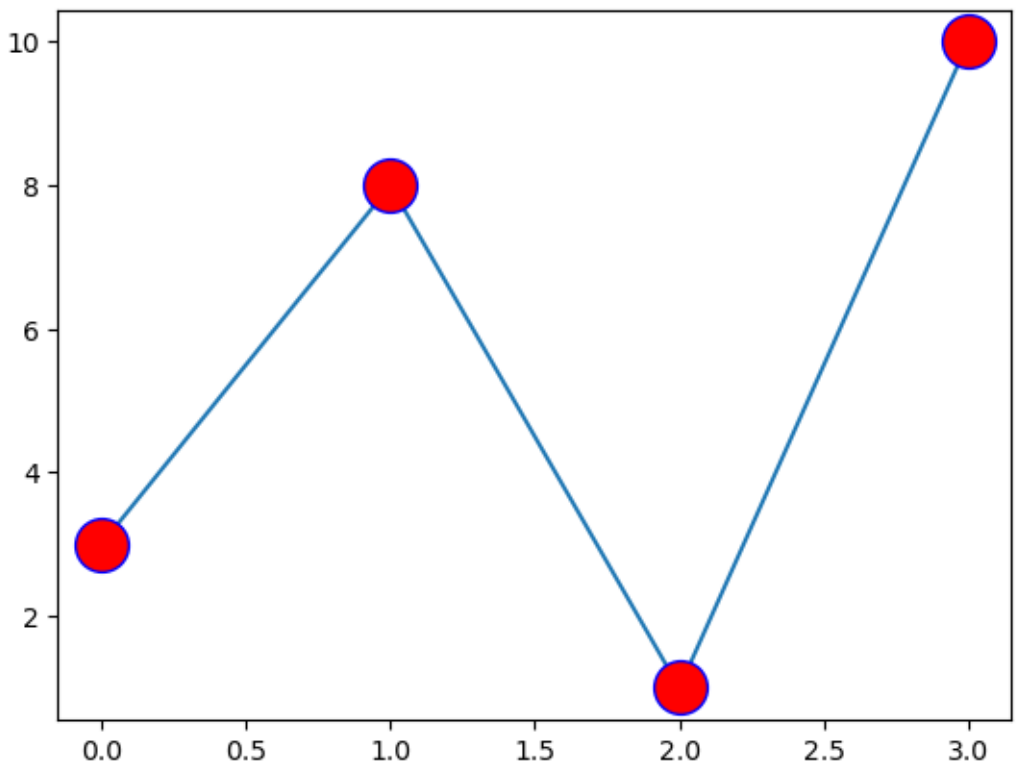
[8]:
```
#Draw a line in the diagram at (1,2) and (2,5) then (6,1) to (8,6)
import numpy as np
x=np.array([1,2,6,8])
y=np.array([2,5,1,6])
plt.plot(x,y)
plt.show()
```



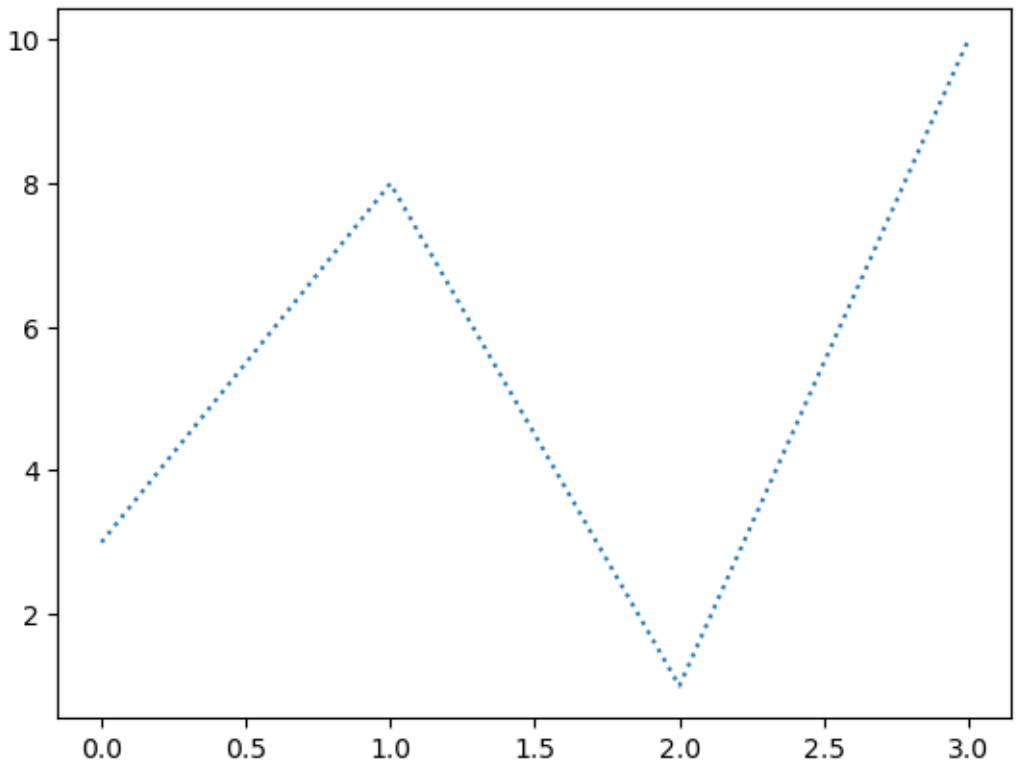
[9]:
```
#plotting without x-points
y=np.array([1,5,3])
plt.plot(y)
plt.show()
```



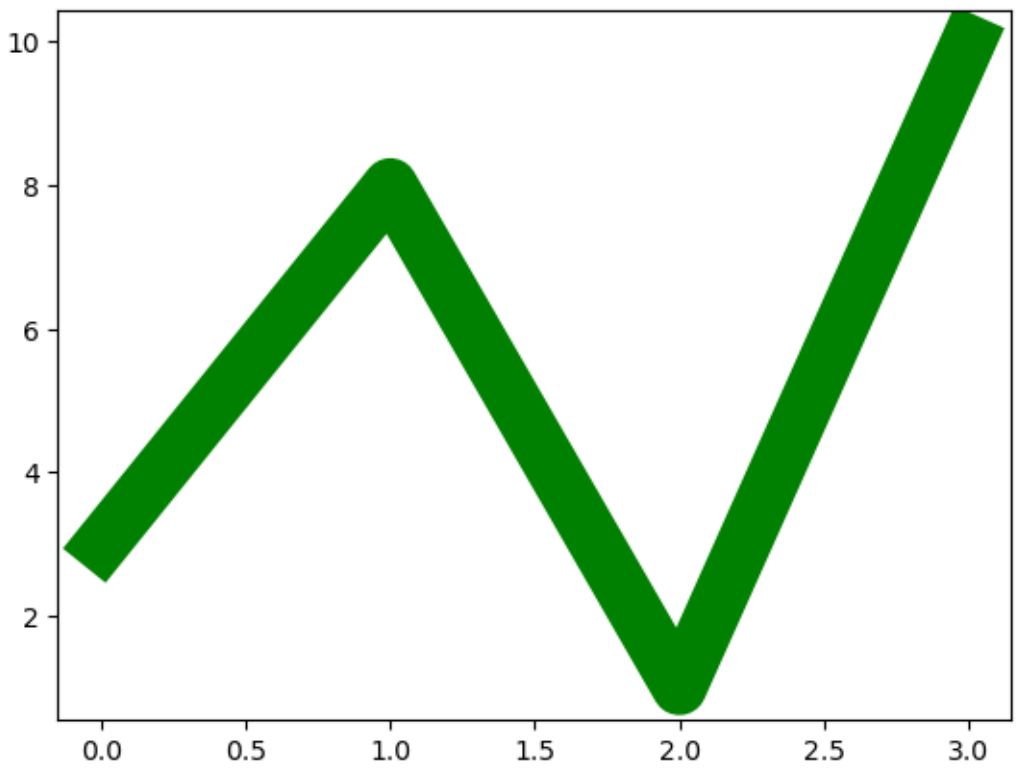
[10]:
```
#setting markersize(ms),marker edge color(mec) and marker face color(mfc)
y=np.array([3,8,1,10])
plt.plot(y,marker='o',ms=20,mec='b',mfc='r')
plt.show()
```



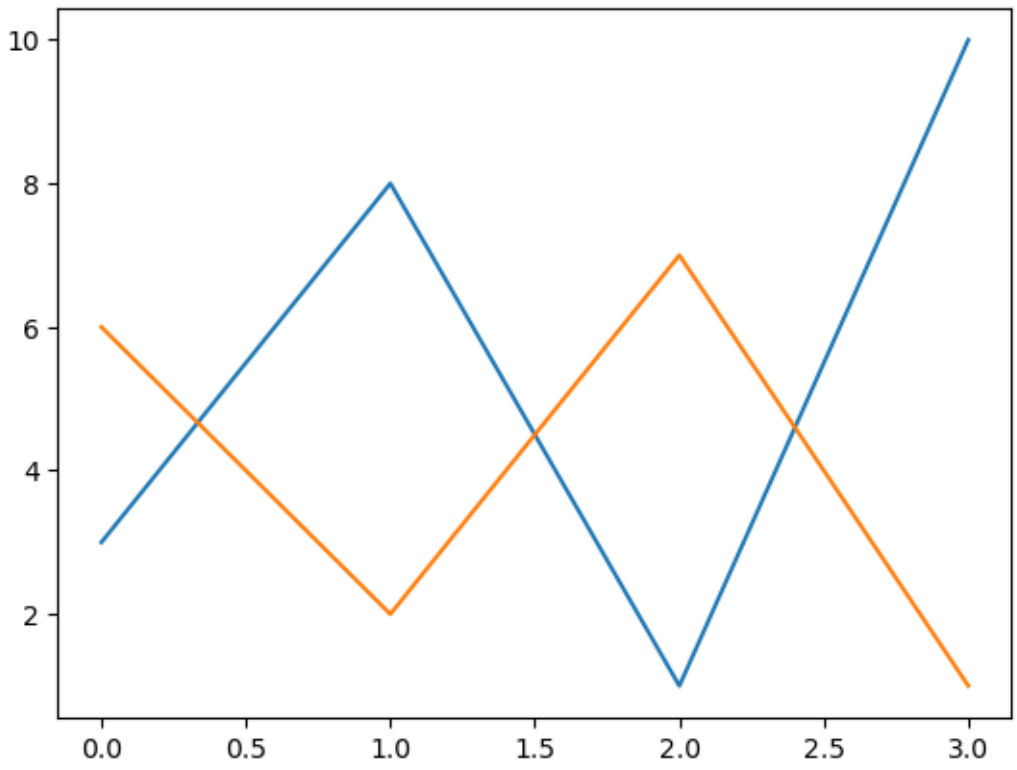
[13]:
```
#setting style of the line(ls)
y=np.array([3,8,1,10])
plt.plot(y,linestyle='dotted')
plt.show()
```



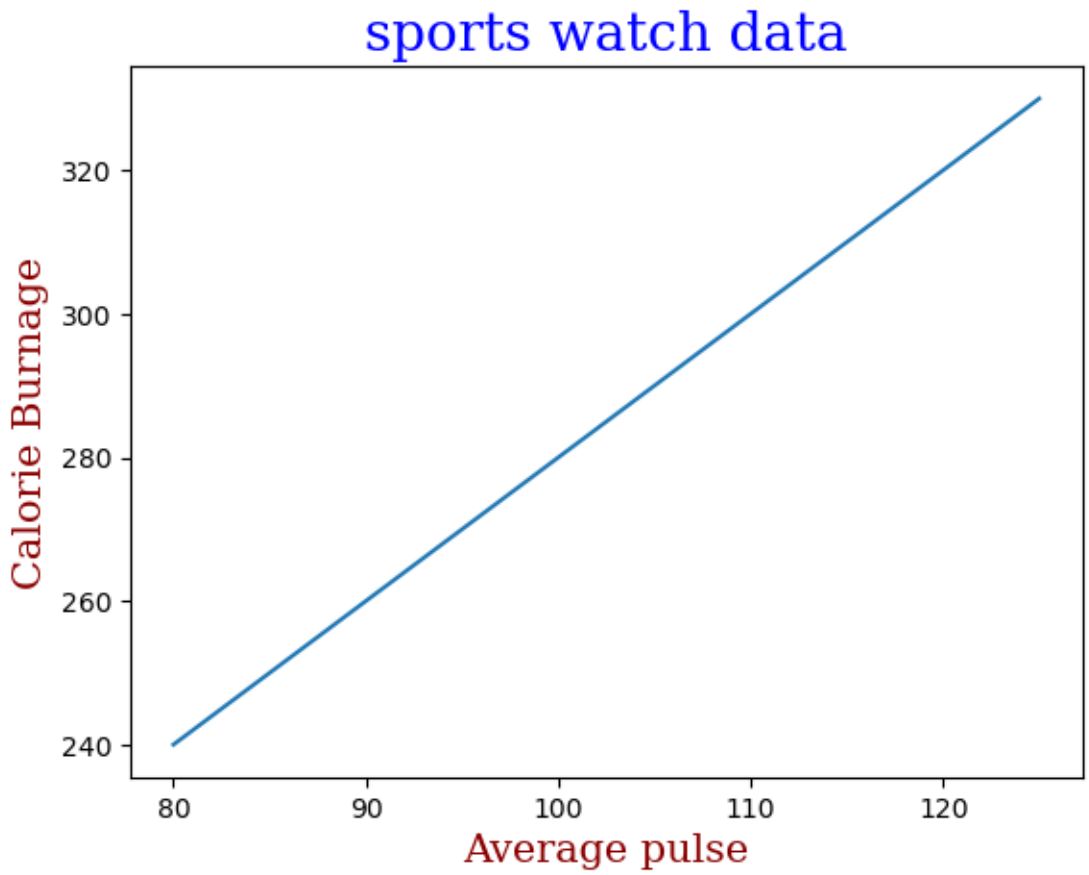
[14]:
```
#setting color(color or c) and width(linewidth or lw) of the line
y=np.array([3,8,1,10])
plt.plot(y,linewidth=20.5,c='g')
plt.show()
```



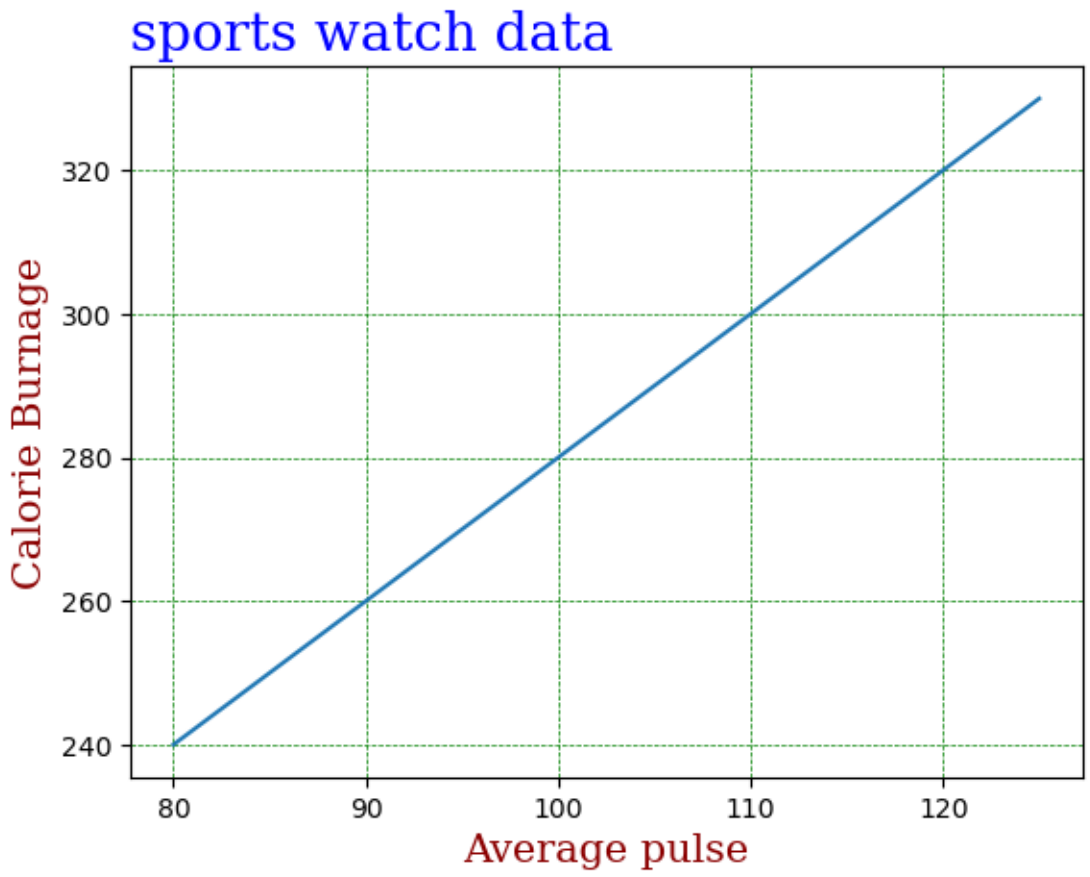
[15]:
```
#drawing multiple lines
y1=np.array([3,8,1,10])
y2=np.array([6,2,7,1])
plt.plot(y1)
plt.plot(y2)
plt.show()
```



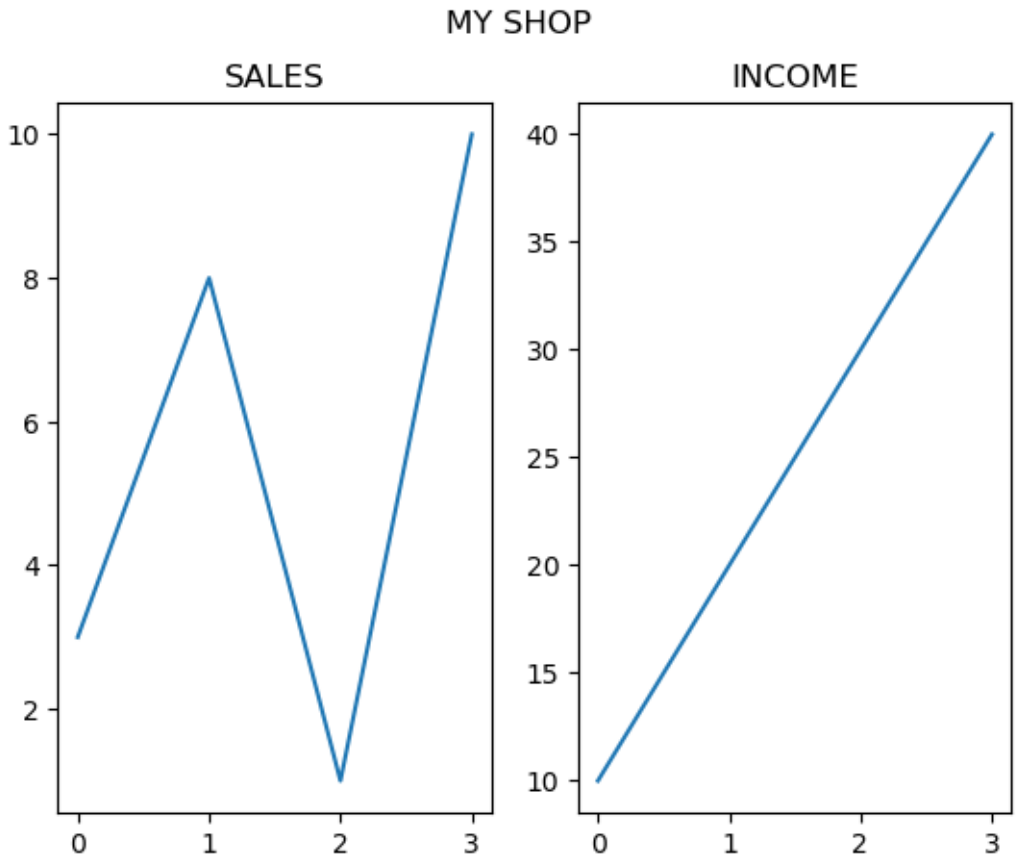
[19]:
```
#set labels,title and add fontdict parameter
x=np.array([80,85,90,95,100,105,110,115,120,125])
y=np.array([240,250,260,270,280,290,300,310,320,330])
font1={'family':'serif','color':'blue','size':20}
font2={'family':'serif','color':'darkred','size':15}
plt.title("sports watch data",fontdict=font1)
plt.xlabel("Average pulse",fontdict=font2)
plt.ylabel("Calorie Burnage",fontdict=font2)
plt.plot(x,y)
plt.show()
```



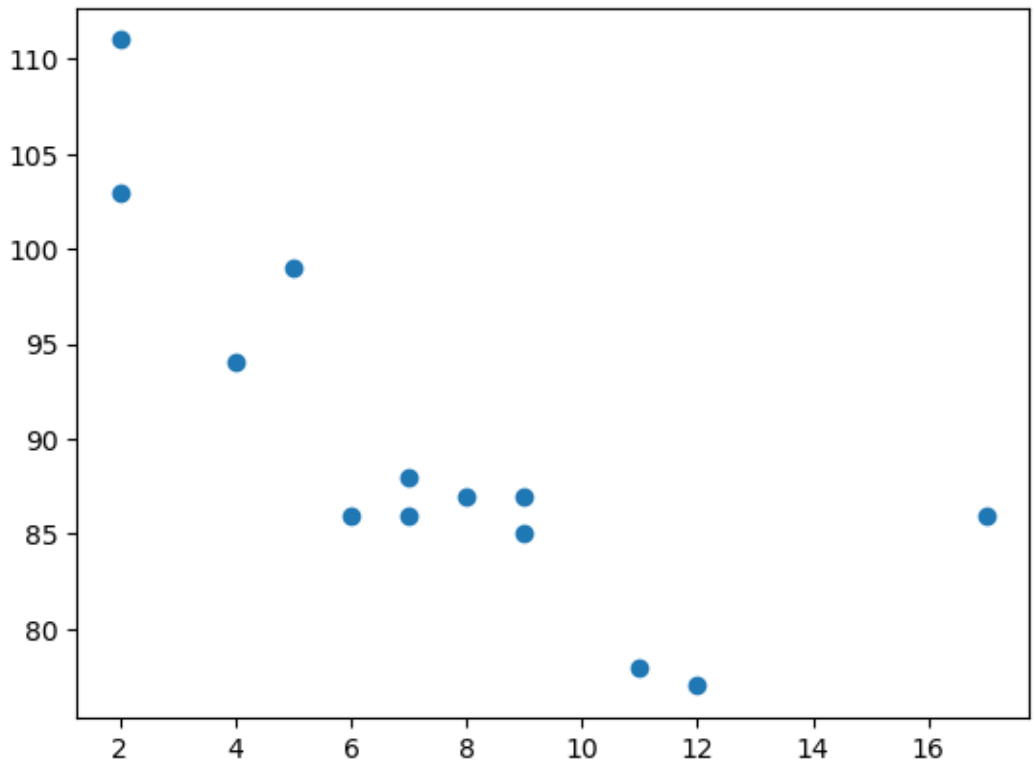
[22]:
```
#using loc() parameter to change the location of the title and add line␣
 ↪properties of grid
x=np.array([80,85,90,95,100,105,110,115,120,125])
y=np.array([240,250,260,270,280,290,300,310,320,330])
font1={'family':'serif','color':'blue','size':20}
font2={'family':'serif','color':'darkred','size':15}
plt.title("sports watch data",fontdict=font1,loc="left")
plt.xlabel("Average pulse",fontdict=font2)
plt.ylabel("Calorie Burnage",fontdict=font2)
plt.plot(x,y)
plt.grid(color="green",linestyle='dashed',linewidth='0.5')
plt.show()
```



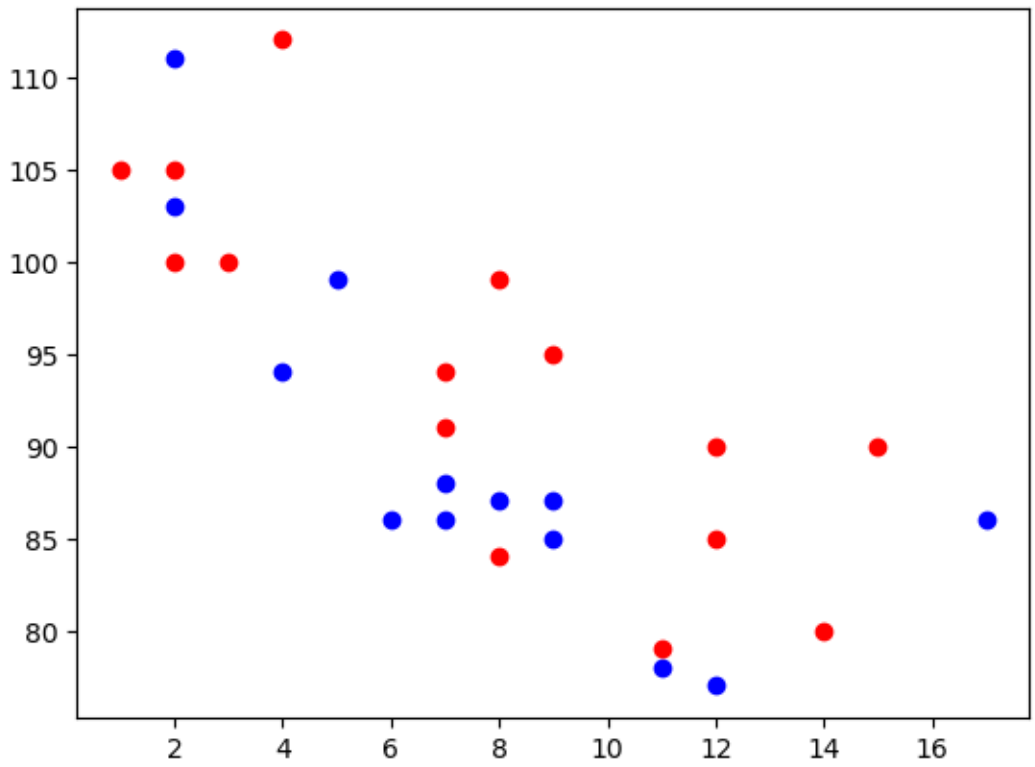
[25]:
```
#using subplot,title() and suptitle() functions
x=np.array([0,1,2,3])
y=np.array([3,8,1,10])
plt.subplot(1,2,1)
plt.plot(x,y)
plt.title("SALES")
x=np.array([0,1,2,3])
y=np.array([10,20,30,40])
plt.subplot(1,2,2)
plt.plot(x,y)
plt.title("INCOME")
plt.suptitle("MY SHOP")
plt.show()
```



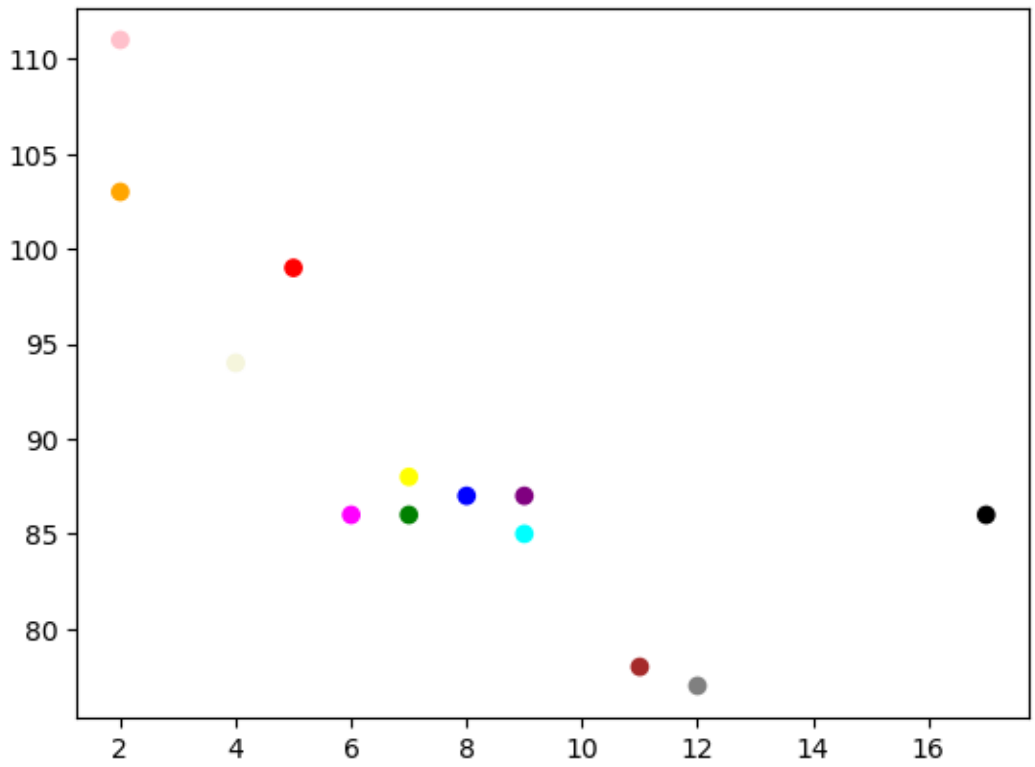
[26]:
```
#scatter plot
x=np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y=np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x,y)
plt.show()
```



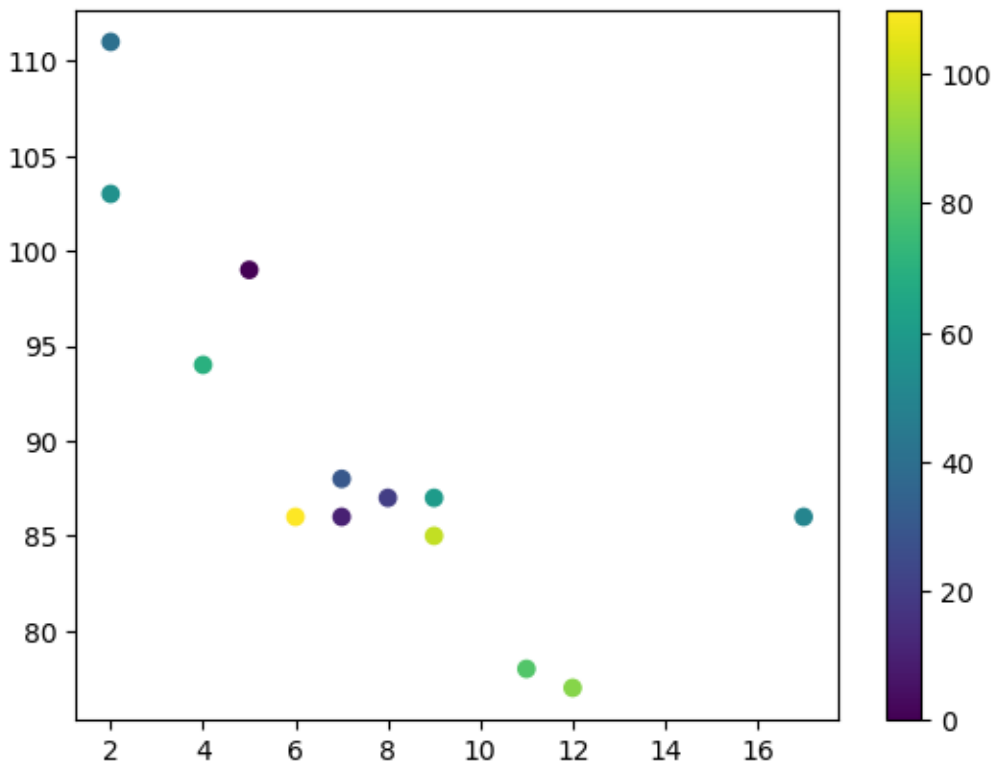
[27]:
```
#setting color for scatterplot using color or c parameter
x=np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y=np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x,y,color='blue')
x=np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
y=np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])
plt.scatter(x,y,color="red")
plt.show()
```



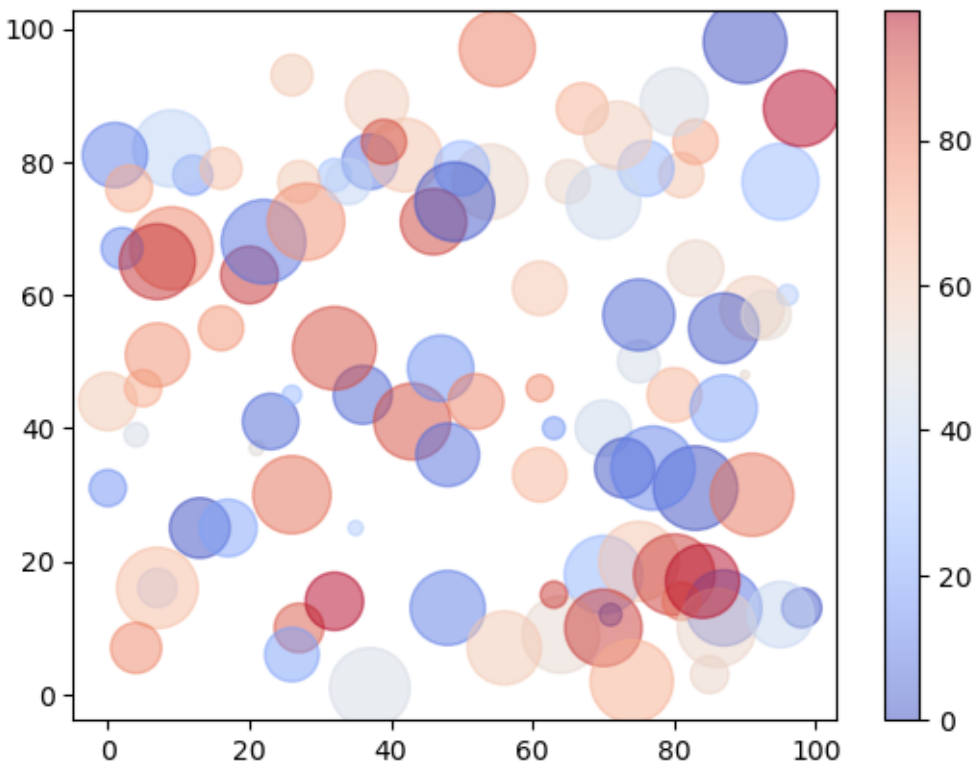
[28]:
```
#color each dot in scatter plot
x=np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y=np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
colors=np.
 ↪array(['red','green','blue','yellow','pink','black','orange','purple','beige','brown','grey
plt.scatter(x,y,c=colors)
plt.show()
```



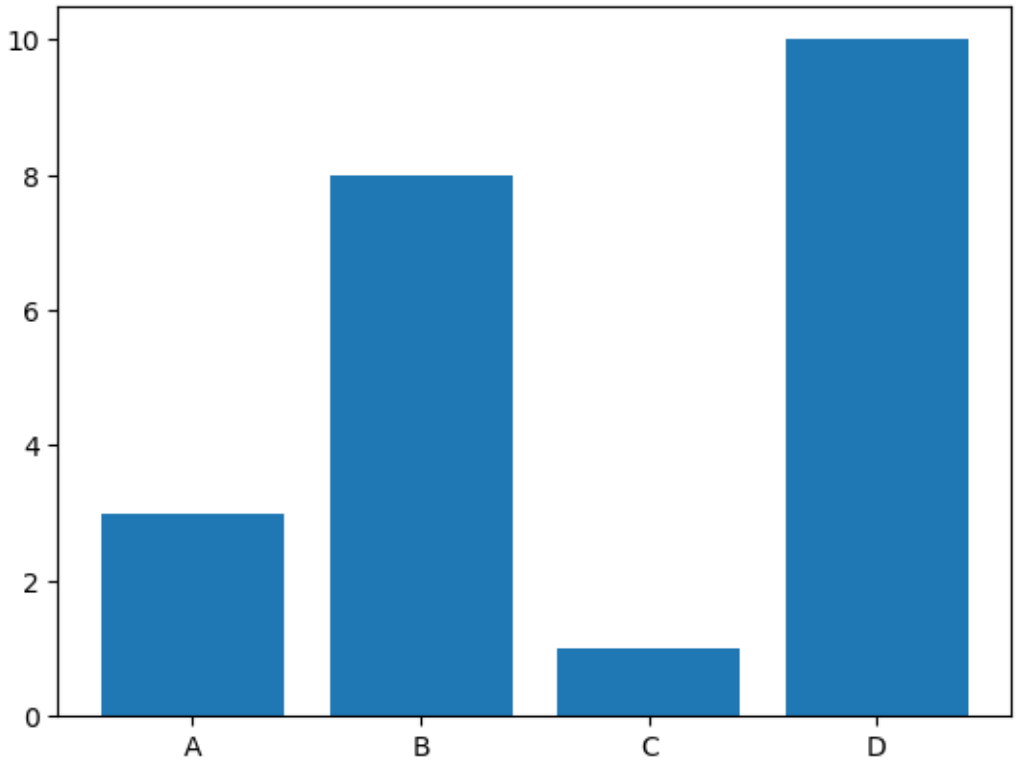
[30]:
```
#using colormap
x=np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y=np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
colors=np.array([0,10,20,30,40,50,55,60,70,80,90,100,110])
plt.scatter(x,y,c=colors,cmap='viridis')
plt.colorbar()
plt.show()
```



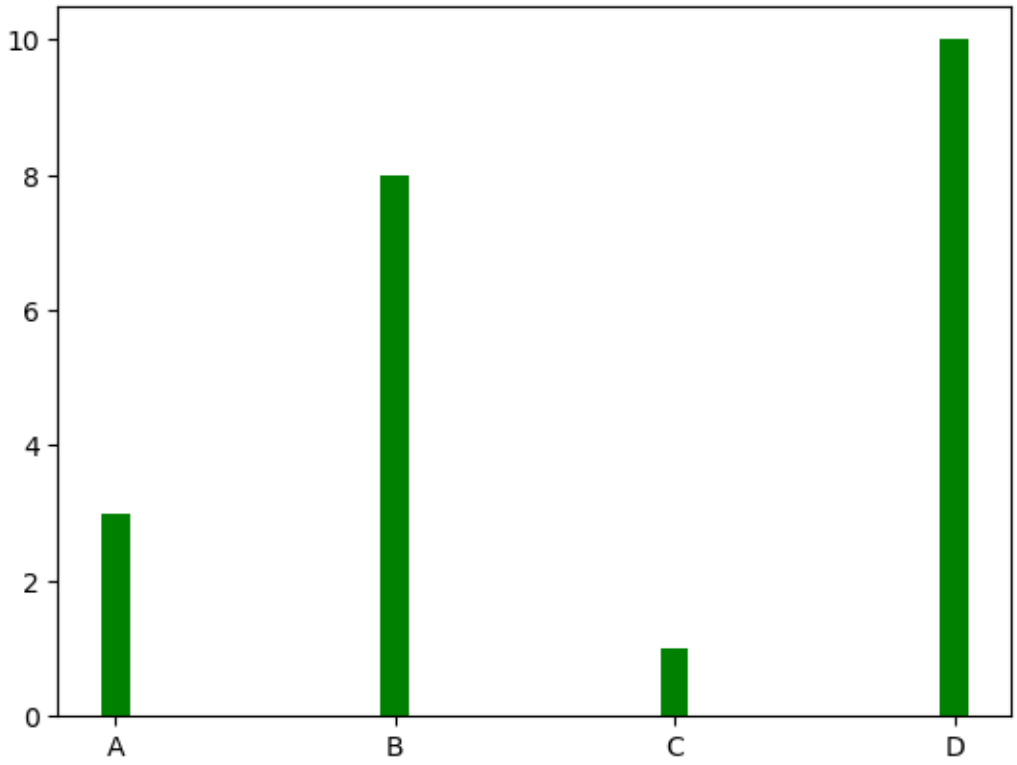
[34]:
```
#changing the size and transparency using size and aplha parameters
x=np.random.randint(100,size=(100))
y=np.random.randint(100,size=(100))
colors=np.random.randint(100,size=(100))
size=10*np.random.randint(100,size=(100))
plt.scatter(x,y,c=colors,s=size,alpha=0.5,cmap='coolwarm')
plt.colorbar()
plt.show()
```



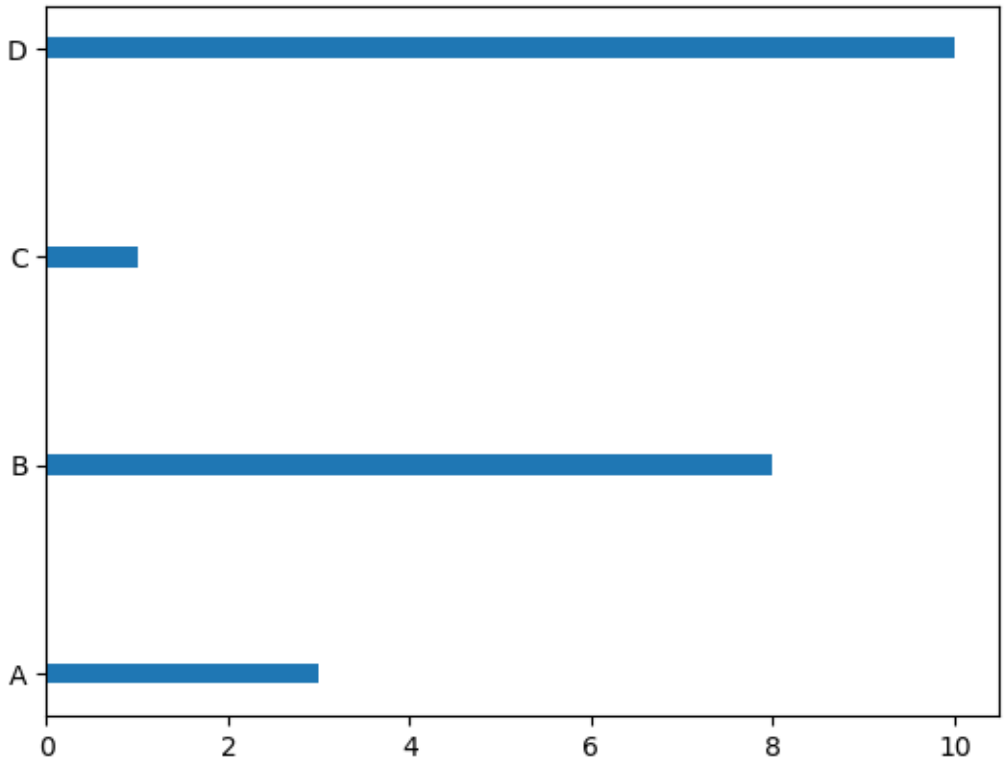
[35]:
```
#creating bargraph
x=np.array(["A","B","C","D"])
y=np.array([3,8,1,10])
plt.bar(x,y)
plt.show()
```



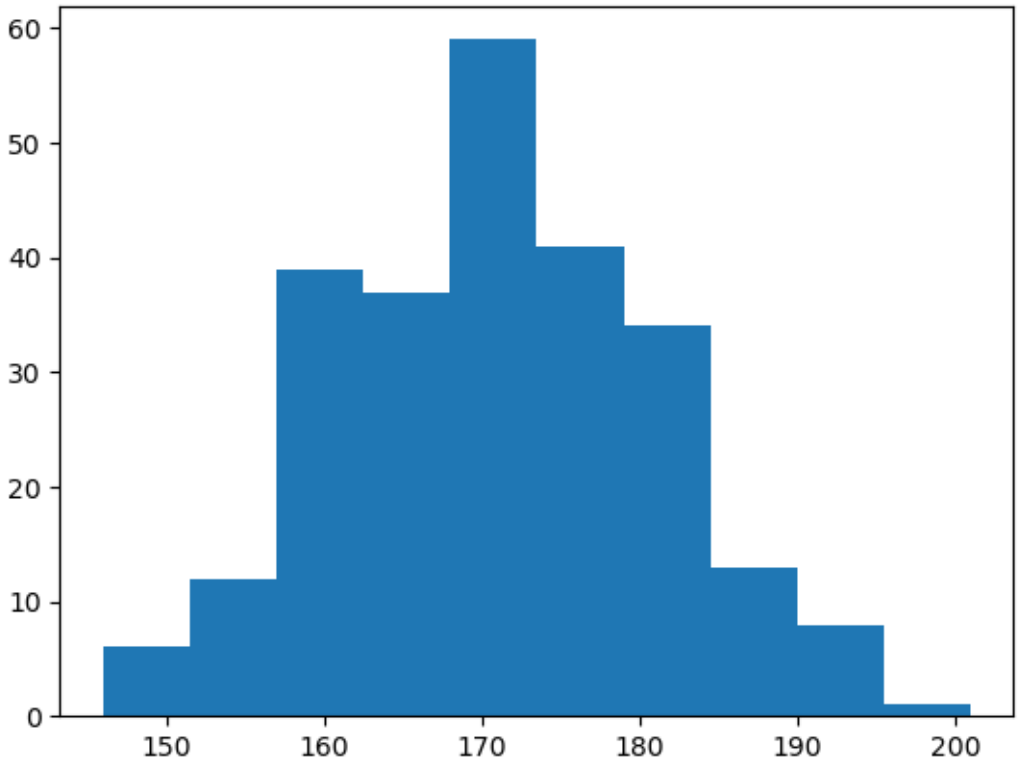
[36]:
```
#setting bar color and width
x=np.array(["A","B","C","D"])
y=np.array([3,8,1,10])
plt.bar(x,y,width=0.1,color='green')
plt.show()
```



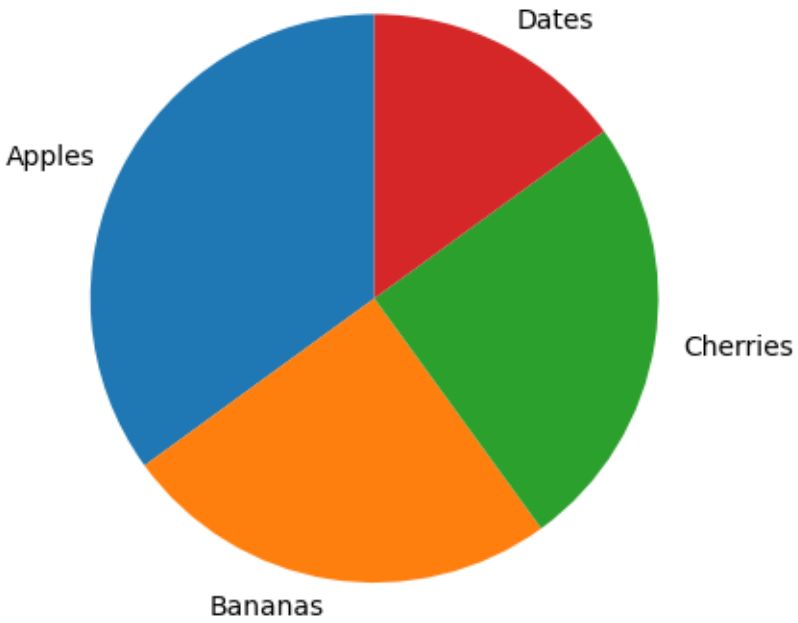
[37]:
```
#horizontal bars using barh() and setting height of the bar
x=np.array(["A","B","C","D"])
y=np.array([3,8,1,10])
plt.barh(x,y,height=0.1)
plt.show()
```



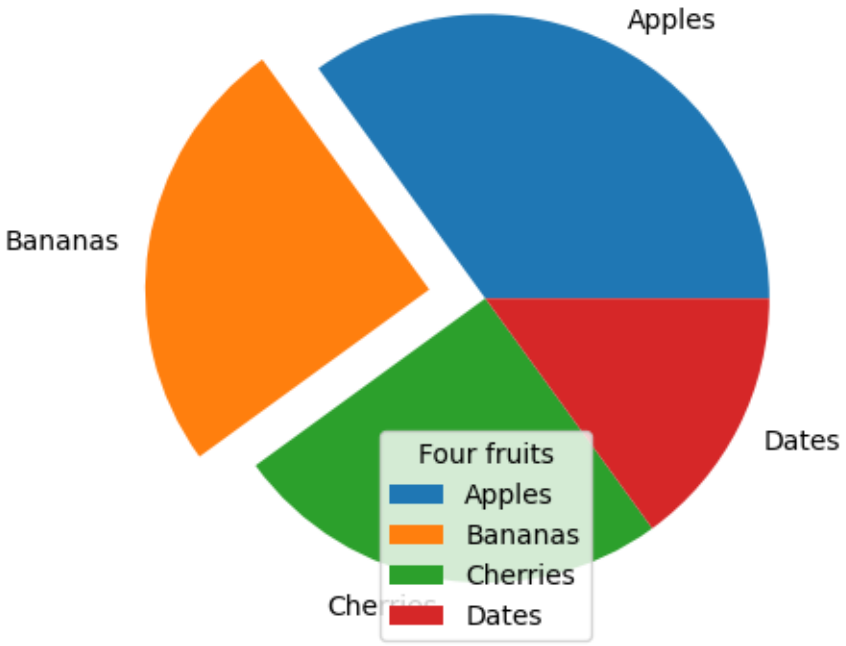
[38]:
```
#histogram
x=np.random.normal(170,10,250)
plt.hist(x)
plt.show()
```



[39]:
```
#pie Chart with labels
y=np.array([35,25,25,15])
mylabels=["Apples","Bananas","Cherries","Dates"]
plt.pie(y,labels=mylabels,startangle=90)
plt.show()
```



[43]:
```
#using explode and legend()
y=np.array([35,25,25,15])
mylabels=["Apples","Bananas","Cherries","Dates"]
explode = [0, 0.2, 0, 0]
plt.pie(y,labels=mylabels,explode=explode)
plt.legend(title="Four fruits")
plt.show()
```



[44]:
```
pip install scikit-learn
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: scikit-learn in
c:\programdata\anaconda3\lib\site-packages (1.2.1)
Requirement already satisfied: joblib>=1.1.1 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn) (1.1.1)
Requirement already satisfied: scipy>=1.3.2 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn) (1.10.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn) (2.2.0)
Requirement already satisfied: numpy>=1.17.3 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn) (1.23.5)
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```
#loading data using sklearn
import pandas as pd
from sklearn.datasets import load_wine
wine_data=load_wine()
wine_df=pd.DataFrame(wine_data.data,columns=wine_data.feature_names)
wine_df
```

[2]:
```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_phenols  \
0      14.23        1.71  2.43               15.6      127.0           2.80
1      13.20        1.78  2.14               11.2      100.0           2.65
2      13.16        2.36  2.67               18.6      101.0           2.80
3      14.37        1.95  2.50               16.8      113.0           3.85
4      13.24        2.59  2.87               21.0      118.0           2.80
..       …          …   …                  …         …             …
173    13.71        5.65  2.45               20.5       95.0           1.68
174    13.40        3.91  2.48               23.0      102.0           1.80
175    13.27        4.28  2.26               20.0      120.0           1.59
176    13.17        2.59  2.37               20.0      120.0           1.65
177    14.13        4.10  2.74               24.5       96.0           2.05

     flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue  \
0          3.06                  0.28             2.29             5.64  1.04
1          2.76                  0.26             1.28             4.38  1.05
2          3.24                  0.30             2.81             5.68  1.03
3          3.49                  0.24             2.18             7.80  0.86
4          2.69                  0.39             1.82             4.32  1.04
..          …                    …                …              …   …
173        0.61                  0.52             1.06             7.70  0.64
174        0.75                  0.43             1.41             7.30  0.70
175        0.69                  0.43             1.35            10.20  0.59
176        0.68                  0.53             1.46             9.30  0.60
177        0.76                  0.56             1.35             9.20  0.61

     od280/od315_of_diluted_wines  proline
0                            3.92   1065.0
1                            3.40   1050.0
2                            3.17   1185.0
3                            3.45   1480.0
4                            2.93    735.0
..                            …      …
173                          1.74    740.0
174                          1.56    750.0
175                          1.56    835.0
176                          1.62    840.0
177                          1.60    560.0

[178 rows x 13 columns]
```

[3]:
```
#Data Exploration
wine_df["target"]=wine_data.target
wine_df.head()
```

[3]:
```
   alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_phenols  \
0    14.23        1.71  2.43               15.6      127.0           2.80
```

```
1    13.20        1.78  2.14               11.2      100.0           2.65
2    13.16        2.36  2.67               18.6      101.0           2.80
3    14.37        1.95  2.50               16.8      113.0           3.85
4    13.24        2.59  2.87               21.0      118.0           2.80

   flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue  \
0        3.06                  0.28             2.29             5.64  1.04
1        2.76                  0.26             1.28             4.38  1.05
2        3.24                  0.30             2.81             5.68  1.03
3        3.49                  0.24             2.18             7.80  0.86
4        2.69                  0.39             1.82             4.32  1.04

   od280/od315_of_diluted_wines  proline  target
0                          3.92   1065.0       0
1                          3.40   1050.0       0
2                          3.17   1185.0       0
3                          3.45   1480.0       0
4                          2.93    735.0       0
```

[4]:
```
wine_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   alcohol                       178 non-null    float64
 1   malic_acid                    178 non-null    float64
 2   ash                           178 non-null    float64
 3   alcalinity_of_ash             178 non-null    float64
 4   magnesium                     178 non-null    float64
 5   total_phenols                 178 non-null    float64
 6   flavanoids                    178 non-null    float64
 7   nonflavanoid_phenols          178 non-null    float64
 8   proanthocyanins               178 non-null    float64
 9   color_intensity               178 non-null    float64
 10  hue                           178 non-null    float64
 11  od280/od315_of_diluted_wines  178 non-null    float64
 12  proline                       178 non-null    float64
 13  target                        178 non-null    int32
dtypes: float64(13), int32(1)
memory usage: 18.9 KB
```

[50]:
```
wine_df.describe()
```

[50]:
```
          alcohol  malic_acid         ash  alcalinity_of_ash   magnesium  \
count  178.000000  178.000000  178.000000         178.000000  178.000000
```

```
mean    13.000618    2.336348    2.366517          19.494944   99.741573
std      0.811827    1.117146    0.274344           3.339564   14.282484
min     11.030000    0.740000    1.360000          10.600000   70.000000
25%     12.362500    1.602500    2.210000          17.200000   88.000000
50%     13.050000    1.865000    2.360000          19.500000   98.000000
75%     13.677500    3.082500    2.557500          21.500000  107.000000
max     14.830000    5.800000    3.230000          30.000000  162.000000

       total_phenols  flavanoids  nonflavanoid_phenols  proanthocyanins  \
count     178.000000  178.000000            178.000000       178.000000
mean        2.295112    2.029270              0.361854         1.590899
std         0.625851    0.998859              0.124453         0.572359
min         0.980000    0.340000              0.130000         0.410000
25%         1.742500    1.205000              0.270000         1.250000
50%         2.355000    2.135000              0.340000         1.555000
75%         2.800000    2.875000              0.437500         1.950000
max         3.880000    5.080000              0.660000         3.580000

       color_intensity         hue  od280/od315_of_diluted_wines      proline  \
count       178.000000  178.000000                    178.000000   178.000000
mean          5.058090    0.957449                      2.611685   746.893258
std           2.318286    0.228572                      0.709990   314.907474
min           1.280000    0.480000                      1.270000   278.000000
25%           3.220000    0.782500                      1.937500   500.500000
50%           4.690000    0.965000                      2.780000   673.500000
75%           6.200000    1.120000                      3.170000   985.000000
max          13.000000    1.710000                      4.000000  1680.000000

           target
count  178.000000
mean     0.938202
std      0.775035
min      0.000000
25%      0.000000
50%      1.000000
75%      2.000000
max      2.000000
```

[5]:
```
wine_df.tail()
```

[5]:
```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_phenols  \
173    13.71        5.65  2.45               20.5       95.0           1.68
174    13.40        3.91  2.48               23.0      102.0           1.80
175    13.27        4.28  2.26               20.0      120.0           1.59
176    13.17        2.59  2.37               20.0      120.0           1.65
177    14.13        4.10  2.74               24.5       96.0           2.05
```

```
     flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue  \
173        0.61                  0.52             1.06              7.7  0.64
174        0.75                  0.43             1.41              7.3  0.70
175        0.69                  0.43             1.35             10.2  0.59
176        0.68                  0.53             1.46              9.3  0.60
177        0.76                  0.56             1.35              9.2  0.61

     od280/od315_of_diluted_wines  proline  target
173                          1.74    740.0       2
174                          1.56    750.0       2
175                          1.56    835.0       2
176                          1.62    840.0       2
177                          1.60    560.0       2
```

[6]:
```
#Data preprocesing
from sklearn.preprocessing import StandardScaler
X = wine_df[wine_data.feature_names].copy()
y = wine_df["target"].copy()
scaler = StandardScaler()
scaler.fit(X)
X_scaled = scaler.transform(X.values)
print(X_scaled[0])
```

```
[ 1.51861254 -0.5622498   0.23205254 -1.16959318  1.91390522  0.80899739
  1.03481896 -0.65956311  1.22488398  0.25171685  0.36217728  1.84791957
  1.01300893]

C:\ProgramData\anaconda3\lib\site-packages\sklearn\base.py:420: UserWarning: X
does not have valid feature names, but StandardScaler was fitted with feature
names
  warnings.warn(
```

[9]:
```
#Training the model
from sklearn.model_selection import train_test_split
X_train_scaled, X_test_scaled, y_train, y_test =train_test_split(X_scaled, y,␣
 ↪train_size=0.7, random_state=25)
print( f"Train Size : {round(len(X_train_scaled)/len(X)*100)} %\nTest Size :␣
 ↪{round(len(X_test_scaled)/len(X)*100)} %")
```

```
Train Size : 70 %
Test Size : 30 %
```

[10]:
```
#Building the model
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
#insantiating the model
logistic_regression = LogisticRegression()
```

```
svm = SVC()
tree = DecisionTreeClassifier()
#Training the Models
logistic_regression.fit(X_train_scaled, y_train)
svm.fit(X_train_scaled, y_train)
tree.fit(X_train_scaled, y_train)
#Making Predictions
log_reg_preds = logistic_regression.predict(X_test_scaled)
svm_preds = svm.predict(X_test_scaled)
tree_preds = tree.predict(X_test_scaled)
```

[12]:
```
#Model evaulation
from sklearn.metrics import classification_report
model_preds = {
    "Logistic Regression": log_reg_preds,
    "Support Vector Machine": svm_preds,
    "Decision Tree": tree_preds
}
for model, preds in model_preds.items():
    print(f"{model} Results:\n{classification_report(y_test, preds)}",␣
 ↪sep="\n\n" )
```

```
Logistic Regression Results:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       1.00      0.92      0.96        25
           2       0.86      1.00      0.92        12

    accuracy                           0.96        54
   macro avg       0.95      0.97      0.96        54
weighted avg       0.97      0.96      0.96        54

Support Vector Machine Results:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        17
           1       1.00      1.00      1.00        25
           2       1.00      1.00      1.00        12

    accuracy                           1.00        54
   macro avg       1.00      1.00      1.00        54
weighted avg       1.00      1.00      1.00        54

Decision Tree Results:
              precision    recall  f1-score   support
```

```
              0       0.89      0.94      0.91        17
              1       0.96      0.88      0.92        25
              2       0.92      1.00      0.96        12

       accuracy                           0.93        54
      macro avg       0.92      0.94      0.93        54
   weighted avg       0.93      0.93      0.93        54
```

[ ]: