Pandas:

Pandas is a python library it is used for analyzing and manipulating the data.
we can also read and create dataset using python
pandas is of 2 types
    1.Series (contain single column)
    2.Dataframe (contain multiple rows and columns)

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

In [2]:
```
#Creating a series
a = ['a','b','c','d']
print(a)
```

```
['a', 'b', 'c', 'd']
```

In [3]:
```
s = pd.Series(a)
print(s)
```

```
0    a
1    b
2    c
3    d
dtype: object
```

In [4]:
```
type(s)
```

Out[4]:
```
pandas.core.series.Series
```

In [5]:
```
s1 = pd.Series(a,index=[10,20,30,40])
print(s1)
```

```
10    a
20    b
30    c
40    d
dtype: object
```

In [6]:
```
#Creating Series from dictionary
d = {1:'A',2:'B',3:'C',4:'D'}
print(d)
```

```
{1: 'A', 2: 'B', 3: 'C', 4: 'D'}
```

In [7]:
```
d1 = pd.Series(d)
print(d1)
```

```
1    A
2    B
3    C
4    D
dtype: object
```

In [8]:
```
s2 = pd.Series(d,index=[10,20,30,40])
print(s2)
```

```
10    NaN
20    NaN
30    NaN
40    NaN
dtype: object
```

In [9]:
```
print(s1)
```

```
10    a
20    b
30    c
40    d
dtype: object
```

In [10]:
```
#indexing
s1[20]
```

Out[10]:
```
'b'
```

In [11]:
```
s1[1]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3802, in Index.get_loc(self, key, method, tolerance)
   3801 try:
-> 3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:

File ~\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:138, in pandas._libs.index.IndexEngine.get_loc()

File ~\anaconda3\Lib\site-packages\pandas\_libs\index.pyx:165, in pandas._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:2263, in pandas._libs.hashtable.Int64HashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:2273, in pandas._libs.hashtable.Int64HashTable.get_item()

KeyError: 1

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[11], line 1
----> 1 s1[1]

File ~\anaconda3\Lib\site-packages\pandas\core\series.py:981, in Series.__getitem__(self, key)
    978     return self._values[key]
    980 elif key_is_scalar:
--> 981     return self._get_value(key)
    983 if is_hashable(key):
    984     # Otherwise index.get_value will raise InvalidIndexError
    985     try:
    986         # For labels that don't resolve as scalars like tuples and frozensets

File ~\anaconda3\Lib\site-packages\pandas\core\series.py:1089, in Series._get_value(self, label, takeable)
   1086     return self._values[label]
   1088 # Similar to Index.get_value, but we do not fall back to positional
-> 1089 loc = self.index.get_loc(label)
   1090 return self.index._get_values_for_loc(self, loc, label)

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:3804, in Index.get_loc(self, key, method, tolerance)
   3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:
-> 3804     raise KeyError(key) from err
   3805 except TypeError:
   3806     # If we have a listlike key, _check_indexing_error will raise
   3807     #  InvalidIndexError. Otherwise we fall through and re-raise
   3808     #  the TypeError.
   3809     self._check_indexing_error(key)

KeyError: 1
```

In [12]:
```
#Creating DataFrame
d = {'name':['Arpita','Biswajit','Smruti','Subha','Rihan','Anusaya'],
    'Age':[91,46,96,77,202,106],'Salary':[150,20,1,5,0.02,3]}
print(d)
```

```
{'name': ['Arpita', 'Biswajit', 'Smruti', 'Subha', 'Rihan', 'Anusaya'], 'Age': [91, 46, 96, 77, 202, 106], 'Salary': [150, 20, 1, 5, 0.02, 3]}
```

In [13]:
```
df = pd.DataFrame(d)
print(df)
```

```
       name  Age  Salary
0    Arpita   91  150.00
1  Biswajit   46   20.00
2    Smruti   96    1.00
3     Subha   77    5.00
4     Rihan  202    0.02
5   Anusaya  106    3.00
```

In [14]:
```
#head():Return 5 rows from top bydefault
df.head()
```

Out[14]:

	name	Age	Salary
0	Arpita	91	150.00
1	Biswajit	46	20.00
2	Smruti	96	1.00
3	Subha	77	5.00
4	Rihan	202	0.02

In [15]:
```
df.head(4)
```

Out[15]:

	name	Age	Salary
0	Arpita	91	150.0
1	Biswajit	46	20.0
2	Smruti	96	1.0
3	Subha	77	5.0

In [16]:
```
#tail():Return 5 rows from bottom bydefault
df.tail()
```

Out[16]:

	name	Age	Salary
1	Biswajit	46	20.00
2	Smruti	96	1.00
3	Subha	77	5.00
4	Rihan	202	0.02
5	Anusaya	106	3.00

In [17]:
```
df.tail(3)
```

Out[17]:

	name	Age	Salary
3	Subha	77	5.00
4	Rihan	202	0.02
5	Anusaya	106	3.00

In [18]:
```
#renaming columns
df.rename(columns={'name':'Student Name'})
```

Out[18]:

	Student Name	Age	Salary
0	Arpita	91	150.00
1	Biswajit	46	20.00
2	Smruti	96	1.00
3	Subha	77	5.00
4	Rihan	202	0.02
5	Anusaya	106	3.00

In [19]:
```
d = {'name':['Arpita',np.NaN,'Smruti','Subha','Rihan','Anusaya'],
    'Age':[91,46,96,77,202,106],'Salary':[150,20,1,5,np.NaN,3]}
print(d)
```

```
{'name': ['Arpita', nan, 'Smruti', 'Subha', 'Rihan', 'Anusaya'], 'Age': [91, 46, 96, 77, 202, 106], 'Salary': [150, 20, 1, 5, nan, 3]}
```

In [20]:
```
df1 = pd.DataFrame(d)
df1.head()
```

Out[20]:

	name	Age	Salary
0	Arpita	91	150.0
1	NaN	46	20.0
2	Smruti	96	1.0
3	Subha	77	5.0
4	Rihan	202	NaN

In [21]:
```
#check null values or missing values
df1.isnull()
```

Out[21]:

	name	Age	Salary
0	False	False	False
1	True	False	False
2	False	False	False
3	False	False	False
4	False	False	True
5	False	False	False

In [22]:
```
df1.isnull().sum()
```

Out[22]:
```
name      1
Age       0
Salary    1
dtype: int64
```

In [23]:
```
#checking datatypes
df1.dtypes
```

Out[23]:
```
name       object
Age         int64
Salary    float64
dtype: object
```

In [24]:
```
#info()
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   name    5 non-null      object 
 1   Age     6 non-null      int64  
 2   Salary  5 non-null      float64
dtypes: float64(1), int64(1), object(1)
memory usage: 276.0+ bytes
```

In [25]:
```
df1.columns
```

Out[25]:
```
Index(['name', 'Age', 'Salary'], dtype='object')
```

In [26]:
```
df1['name'].unique()
```

Out[26]:
```
array(['Arpita', nan, 'Smruti', 'Subha', 'Rihan', 'Anusaya'], dtype=object)
```

In [27]:
```
for i in df1.columns:
    print(i,':',"\n",df1[i].unique())
```

```
name :
 ['Arpita' nan 'Smruti' 'Subha' 'Rihan' 'Anusaya']
Age :
 [ 91  46  96  77 202 106]
Salary :
 [150.  20.   1.   5.  nan   3.]
```

In [28]:
```
#Describe
df1.describe()
```

Out[28]:

	Age	Salary
count	6.000000	5.000000
mean	103.000000	35.800000
std	52.778784	64.278301
min	46.000000	1.000000
25%	80.500000	3.000000
50%	93.500000	5.000000
75%	103.500000	20.000000
max	202.000000	150.000000

In [29]:
```
df1.describe(include='O')
```

Out[29]:

	name
count	5
unique	5
top	Arpita
freq	1

In [30]:
```
d = df1.describe(include='all')
d
```

Out[30]:

	name	Age	Salary
count	5	6.000000	5.000000
unique	5	NaN	NaN
top	Arpita	NaN	NaN
freq	1	NaN	NaN
mean	NaN	103.000000	35.800000
std	NaN	52.778784	64.278301
min	NaN	46.000000	1.000000
25%	NaN	80.500000	3.000000
50%	NaN	93.500000	5.000000
75%	NaN	103.500000	20.000000
max	NaN	202.000000	150.000000

In [31]:
```
df1.Salary.fillna(35.800000,inplace=True)
```

In [32]:
```
df1.isnull().sum()
```

Out[32]:
```
name      1
Age       0
Salary    0
dtype: int64
```

In [33]:
```
# df1.name.replace(np.NaN,d[i][2],inplace=True)
```

In [34]:
```
df1.name.fillna('Arpita',inplace=True)
```

In [35]:
```
df1.isnull().sum()
```

Out[35]:
```
name      0
Age       0
Salary    0
dtype: int64
```

In [36]:
```
df1.head()
```

Out[36]:

	name	Age	Salary
0	Arpita	91	150.0
1	Arpita	46	20.0
2	Smruti	96	1.0
3	Subha	77	5.0
4	Rihan	202	35.8

Data Cleaning Steps

1.Data reading(for '.data' and '.csv'extension write 'pd.read_csv' and for '.xlsx' extension write 'pd.read_excel'.
2.Renaming columns(if required)
3.Check null values or missing values(var.isnull().sum())
4.check datatypes (var.dtypes)
5.check unique values for each columns
6.print describe and use the values from it to replace whenever you find problemsin any attribute or column of dataset.
Change datatype after replacing if required.

In [37]:
```
auto = pd.read_csv(r"D:\Dataset\auto-mpg.data",delim_whitespace=True,header=None)
auto.head()
```

Out[37]:

	0	1	2	3	4	5	6	7	8
0	18.0	8	307.0	130.0	3504.0	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693.0	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150.0	3436.0	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150.0	3433.0	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140.0	3449.0	10.5	70	1	ford torino

In [38]:
```
auto.columns = ['mpg','cylinders','displacement','horsepower','weight','acceleration',
                'model year','origin','car name']
```

In [39]:
```
auto.head()
```

Out[39]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130.0	3504.0	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693.0	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150.0	3436.0	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150.0	3433.0	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140.0	3449.0	10.5	70	1	ford torino

In [40]:
```
auto.isnull().sum()
```

Out[40]:
```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    0
model year      0
origin          0
car name        0
dtype: int64
```

In [41]:
```
auto.dtypes
```

Out[41]:
```
mpg             float64
cylinders         int64
displacement    float64
horsepower       object
weight          float64
acceleration    float64
model year        int64
origin            int64
car name         object
dtype: object
```

In [42]:
```
auto.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object 
 4   weight        398 non-null    float64
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64  
 7   origin        398 non-null    int64  
 8   car name      398 non-null    object 
dtypes: float64(4), int64(3), object(2)
memory usage: 28.1+ KB
```

In [43]:
```
for i in auto.columns:
    print(i,':',"\n",auto[i].unique())
```

```
 'mazda glc' 'vw rabbit c (diesel)' 'vw dasher (diesel)'
 'audi 5000s (diesel)' 'mercedes-benz 240d' 'honda civic 1500 gl'
 'renault lecar deluxe' 'vokswagen rabbit' 'datsun 280-zx' 'mazda rx-7 gs'
 'triumph tr7 coupe' 'ford mustang cobra' 'honda accord'
 'plymouth reliant' 'dodge aries wagon (sw)' 'toyota starlet'
 'plymouth champ' 'honda civic 1300' 'datsun 210 mpg' 'toyota tercel'
 'mazda glc 4' 'plymouth horizon 4' 'ford escort 4w' 'ford escort 2h'
 'volkswagen jetta' 'renault 18i' 'honda prelude' 'datsun 200sx'
 'peugeot 505s turbo diesel' 'volvo diesel' 'toyota cressida'
 'datsun 810 maxima' 'oldsmobile cutlass ls' 'ford granada gl'
 'chrysler lebaron salon' 'chevrolet cavalier' 'chevrolet cavalier wagon'
 'chevrolet cavalier 2-door' 'pontiac j2000 se hatchback' 'dodge aries se'
 'ford fairmont futura' 'amc concord dl' 'volkswagen rabbit l'
 'mazda glc custom l' 'mazda glc custom' 'plymouth horizon miser'
 'mercury lynx l' 'nissan stanza xe' 'honda civic (auto)' 'datsun 310 gx'
 'buick century limited' 'oldsmobile cutlass ciera (diesel)'
 'chrysler lebaron medallion' 'ford granada l' 'toyota celica gt'
 'dodge charger 2.2' 'chevrolet camaro' 'ford mustang gl' 'vw pickup'
 'dodge rampage' 'ford ranger' 'chevy s-10']
```

In [44]:
```
for i in auto.columns:
    print(i,':',sum((auto[i])=='?'))
```

```
mpg : 0
cylinders : 0
displacement : 0
horsepower : 6
weight : 0
acceleration : 0
model year : 0
origin : 0
car name : 0
```

In [45]:
```
#describe
d = auto.describe(include='all')
d
```

Out[45]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
count	398.000000	398.000000	398.000000	398	398.000000	398.000000	398.000000	398.000000	398
unique	NaN	NaN	NaN	94	NaN	NaN	NaN	NaN	305
top	NaN	NaN	NaN	150.0	NaN	NaN	NaN	NaN	ford pinto
freq	NaN	NaN	NaN	22	NaN	NaN	NaN	NaN	6
mean	23.514573	5.454774	193.425879	NaN	2970.424623	15.568090	76.010050	1.572864	NaN
std	7.815984	1.701004	104.269838	NaN	846.841774	2.757689	3.697627	0.802055	NaN
min	9.000000	3.000000	68.000000	NaN	1613.000000	8.000000	70.000000	1.000000	NaN
25%	17.500000	4.000000	104.250000	NaN	2223.750000	13.825000	73.000000	1.000000	NaN
50%	23.000000	4.000000	148.500000	NaN	2803.500000	15.500000	76.000000	1.000000	NaN
75%	29.000000	8.000000	262.000000	NaN	3608.000000	17.175000	79.000000	2.000000	NaN
max	46.600000	8.000000	455.000000	NaN	5140.000000	24.800000	82.000000	3.000000	NaN

In [46]:
```
auto['horsepower'].replace('?',d[i][2],inplace=True)
```

In [47]:
```
for i in auto.columns:
    print(i,':',sum((auto[i])=='ford pinto'))
```

```
mpg : 0
cylinders : 0
displacement : 0
horsepower : 6
weight : 0
acceleration : 0
model year : 0
origin : 0
car name : 6
```

In [48]:
```
auto['horsepower'].unique()
```

Out[48]:
```
array(['130.0', '165.0', '150.0', '140.0', '198.0', '220.0', '215.0',
       '225.0', '190.0', '170.0', '160.0', '95.00', '97.00', '85.00',
       '88.00', '46.00', '87.00', '90.00', '113.0', '200.0', '210.0',
       '193.0', 'ford pinto', '100.0', '105.0', '175.0', '153.0', '180.0',
       '110.0', '72.00', '86.00', '70.00', '76.00', '65.00', '69.00',
       '60.00', '80.00', '54.00', '208.0', '155.0', '112.0', '92.00',
       '145.0', '137.0', '158.0', '167.0', '94.00', '107.0', '230.0',
       '49.00', '75.00', '91.00', '122.0', '67.00', '83.00', '78.00',
       '52.00', '61.00', '93.00', '148.0', '129.0', '96.00', '71.00',
       '98.00', '115.0', '53.00', '81.00', '79.00', '120.0', '152.0',
       '102.0', '108.0', '68.00', '58.00', '149.0', '89.00', '63.00',
       '48.00', '66.00', '139.0', '103.0', '125.0', '133.0', '138.0',
       '135.0', '142.0', '77.00', '62.00', '132.0', '84.00', '64.00',
       '74.00', '116.0', '82.00'], dtype=object)
```

In [49]:
```
auto.horsepower.replace('ford pinto',d[i][2],inplace=True)
```

In [50]:
```
auto['horsepower'].unique()
```

Out[50]:
```
array(['130.0', '165.0', '150.0', '140.0', '198.0', '220.0', '215.0',
       '225.0', '190.0', '170.0', '160.0', '95.00', '97.00', '85.00',
       '88.00', '46.00', '87.00', '90.00', '113.0', '200.0', '210.0',
       '193.0', 'ford pinto', '100.0', '105.0', '175.0', '153.0', '180.0',
       '110.0', '72.00', '86.00', '70.00', '76.00', '65.00', '69.00',
       '60.00', '80.00', '54.00', '208.0', '155.0', '112.0', '92.00',
       '145.0', '137.0', '158.0', '167.0', '94.00', '107.0', '230.0',
       '49.00', '75.00', '91.00', '122.0', '67.00', '83.00', '78.00',
       '52.00', '61.00', '93.00', '148.0', '129.0', '96.00', '71.00',
       '98.00', '115.0', '53.00', '81.00', '79.00', '120.0', '152.0',
       '102.0', '108.0', '68.00', '58.00', '149.0', '89.00', '63.00',
       '48.00', '66.00', '139.0', '103.0', '125.0', '133.0', '138.0',
       '135.0', '142.0', '77.00', '62.00', '132.0', '84.00', '64.00',
       '74.00', '116.0', '82.00'], dtype=object)
```

In [51]:
```
auto.dtypes
```

Out[51]:
```
mpg             float64
cylinders         int64
displacement    float64
horsepower       object
weight          float64
acceleration    float64
model year        int64
origin            int64
car name         object
dtype: object
```

In [58]:
```
auto['horsepower'] = d['horsepower'].astype('float64')
```

In [59]:
```
auto.dtypes
```

Out[59]:
```
mpg             float64
cylinders         int64
displacement    float64
horsepower      float64
weight          float64
acceleration    float64
model year        int64
origin            int64
car name         object
dtype: object
```

In [52]:
```
auto.drop('car name',axis=1)
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin
0	18.0	8	307.0	130.0	3504.0	12.0	70	1
1	15.0	8	350.0	165.0	3693.0	11.5	70	1
2	18.0	8	318.0	150.0	3436.0	11.0	70	1
3	16.0	8	304.0	150.0	3433.0	12.0	70	1
4	17.0	8	302.0	140.0	3449.0	10.5	70	1
...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86.00	2790.0	15.6	82	1
394	44.0	4	97.0	52.00	2130.0	24.6	82	2
395	32.0	4	135.0	84.00	2295.0	11.6	82	1
396	28.0	4	120.0	79.00	2625.0	18.6	82	1
397	31.0	4	119.0	82.00	2720.0	19.4	82	1

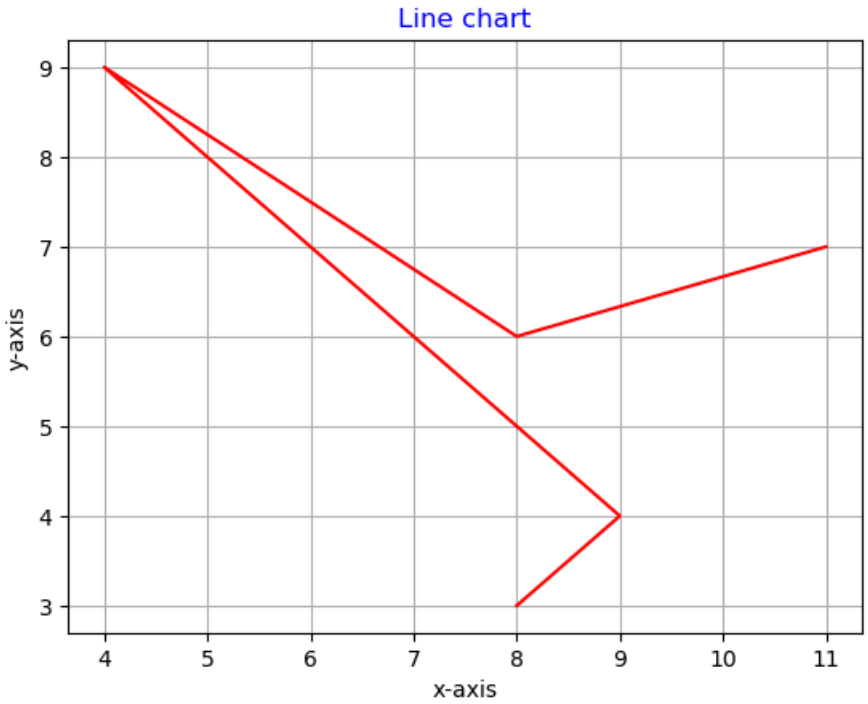
398 rows × 8 columns

In [ ]:

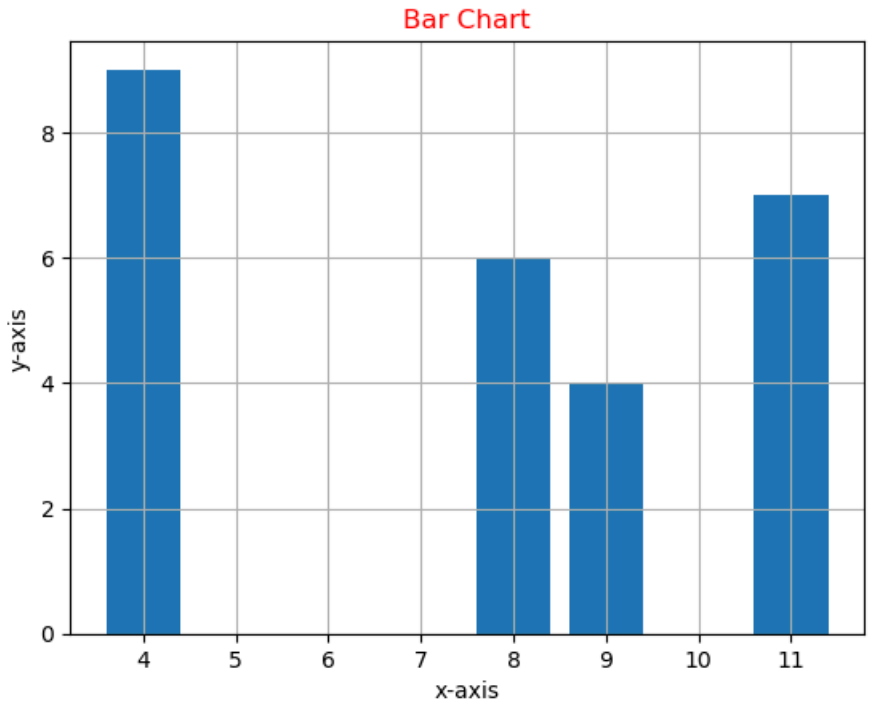
Matplotlib

Data visualization technique

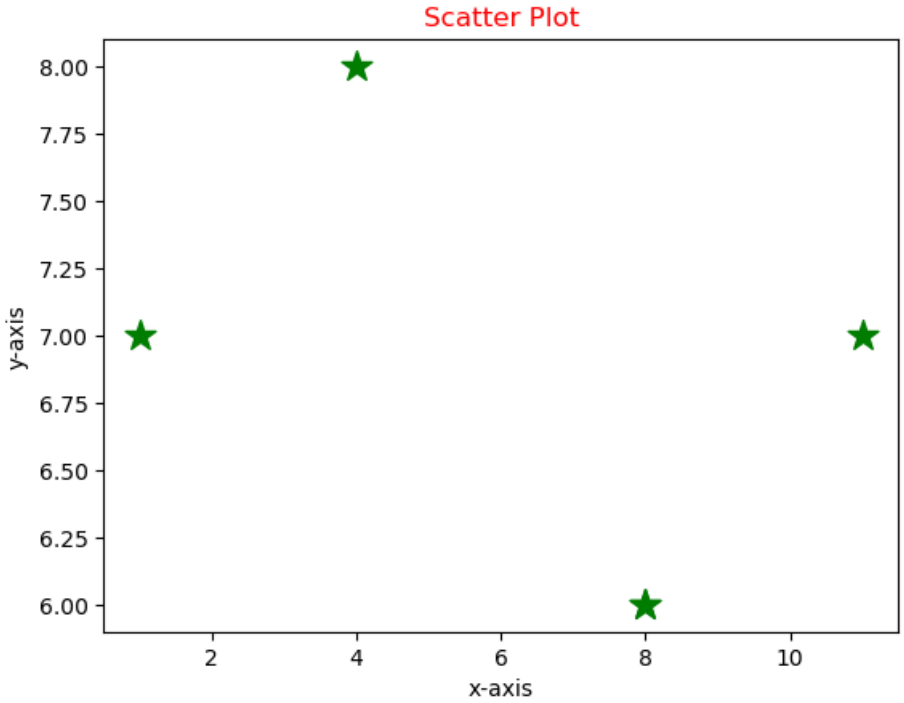
In [53]:
```
import matplotlib.pyplot as plt
from matplotlib import pyplot as plt
#Line plot
x = [8,9,4,8,11]
y = [3,4,9,6,7]
plt.plot(x,y,c='r')
plt.title('Line chart',c='b')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.grid()
plt.show()
```



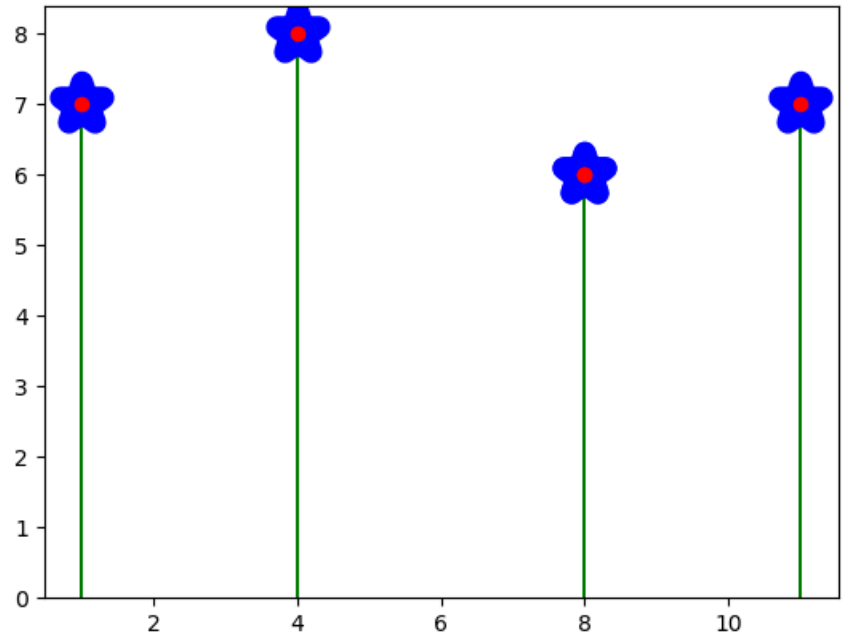
In [54]:
```
#bar Chart
#line plot
x = [8,9,4,8,11]
y = [3,4,9,6,7]
plt.bar(x,y)
plt.title('Bar Chart',c='r')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.grid()
plt.show()
```



In [55]:
```
#Scatter Plot
x = [8,1,4,8,11]
y = [6,7,8,6,7]
plt.scatter(x,y,s=200,marker='*',c='g')
plt.title('Scatter Plot',c='r')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.show()
```



In [56]:
```
x = [8,1,4,8,11]
y = [6,7,8,6,7]
plt.bar(x,y,color='g',width=0.05)
plt.scatter(x,y,s=400,c='blue',marker='*',linewidths=10)
plt.scatter(x,y,c='red')
plt.show()
```



In [ ]: