Assignment Question _List

1st set of Assignment

1.Write a python program which accepts a sequence of comma - separated numbers from user and generate a list and tuple with those

Solution:

In [1]:
```
sample_data=3,5,7,23
```

In [2]:
```
values = input("Input some comma seprated numbers : ")
list = values.split(",")
tuple = tuple(list)
print('List : ',list)
print('Tuple : ',tuple)
```

```
Input some comma seprated numbers : 3,5,7,23
List :  ['3', '5', '7', '23']
Tuple :  ('3', '5', '7', '23')
```

2. Write a python program to display the first and last colors from the following list.

Solution:

In [3]:
```
color_list=["red","green","White","black"]
color_list
```

Out[3]:
```
['red', 'green', 'White', 'black']
```

In [4]:
```
color_list[0:1]
```

Out[4]:
```
['red']
```

In [5]:
```
color_list[0:3]
```

Out[5]:
```
['red', 'green', 'White']
```

In [6]:
```
color_list[0]
```

Out[6]:
```
'red'
```

In [7]:
```
color_list[3]
```

Out[7]:
```
'black'
```

3.Write a python program to print the even number froma given list

Solution:

In [8]:
```
sample_list=[1,2,3,4,5,6,7,8,9]
sample_list
```

Out[8]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [9]:
```
for num in sample_list:
 if num % 2 == 0:
    print(num)
```

```
2
4
6
8
```

==========

Module

1.Write a python program to calculate number of days between two dates.Hint:Datetime package/module.

Solution:

In [10]:
```
Sample_dates=(2014,7,2),(2014,7,11)
```

In [11]:
```
from datetime import date
```

In [12]:
```
def numOfDays(date1,date2):
    return (date2-date1).days
```

In [13]:
```
date1=date(2014,7,2)
date2=date(2014,7,11)
```

In [14]:
```
print(numOfDays(date1,date2),"days")
```

```
9 days
```

==========

FUNCTIONS

1.Write a python program to get the volume of a sphere with radius 6.

Solution:

In [15]:
```
radius=6
```

In [16]:
```
pi=3.1427
```

In [17]:
```
volume=(4/3)*pi*radius**3
```

In [18]:
```
print("The Volume of the sphere is :",volume)
```

```
The Volume of the sphere is : 905.0975999999998
```

2. Write a python program to calculate the sum of the three given numbers.if the values are equal then return three times of their sum (hint-write User defined functions

Solution:

In [19]:
```
def sum_thrice(x, y, z):

     sum = x + y + z

     if x == y == z:
      sum = sum * 3
     return sum

print(sum_thrice(1, 2, 3))
print(sum_thrice(3, 3, 3))
```

```
6
27
```

3.Write a python program to count the number 4 in a given list

Solution:

In [20]:
```
List=[1,4,6,8,9,4]
```

In [21]:
```
def list_count_4(nums):
  count = 0
  for num in nums:
    if num == 4:
      count = count + 1

  return count

print(list_count_4([1, 4, 6, 8, 9,4]))
```

```
2
```

4.Write a python program to print all even numbers from a given number list in the same order and stop the printing if any numbers that come after 237 in the sequence .Go to all editor Sample number list

Solution:

In [22]:
```
list=[399,162,758,219,918,237,412,566,826,248,866,950,626,949,687,217,815,67,104,
      445,742,717,958,743,527]
```

In [23]:
```
numbers = [386, 462, 47, 418, 907, 344, 236, 375, 823, 566, 597, 978, 328, 615, 9
815, 67, 104, 58, 512, 24, 892, 894, 767, 553, 81, 379, 843, 831, 445, 742, 717,

for x in numbers:
    if x == 237:
        print(x)
        break;
    elif x % 2 == 0:
        print(x)
```

```
386
462
418
344
236
566
978
328
162
758
918
237
```

5.Write a python program to find those number which are divisible by 7 and mutiple of 5 ,between 1500 and 2700(both included)

Solution:

In [24]:
```
for i in range(1500,2701):
    if i%7==0 and i%5==0:
        print(" ",i)
```

```
 1505
 1540
 1575
 1610
 1645
 1680
 1715
 1750
 1785
 1820
 1855
 1890
 1925
 1960
 1995
 2030
 2065
 2100
 2135
 2170
 2205
 2240
 2275
 2310
 2345
 2380
 2415
 2450
 2485
 2520
 2555
 2590
 2625
 2660
 2695
```

6.Write a python program that prints all the number from 0 to 6 except 3 and 6

Solution:

In [25]:
```
for x in range(7):
    if(x == 3 or x == 6):
        continue
    print(x,end='  ')
print("\n")
```

```
0  1  2  4  5
```

7.write a python program to get the Fibonacci series between 0 to 50

Note:The fibonacci Sequence is the series of numbers

Solution:

In [26]:
```
x,y=0,1
while y<50:
    print(y)
    x,y = y,x+y
```

```
1
1
2
3
5
8
13
21
34
```

**0,1,1,2,3,5,8,13,21 every next number is found by adding up the two numbers before it.

8. same as 7.

9.Write a python function that takes a list and returns a new list with unique elements of the first list

Solution:

In [27]:
```
Sample_list=[1,2,3,3,3,3,4,5]
```

In [28]:
```
Unique=[1,2,3,4,5]
```

In [29]:
```
def unique_list(l):
    x = []
    for a in l:
        if a not in x:
         x.append(a)
    return x
print(unique_list([1,2,3,3,3,3,4,5]))
```

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

==========

STRING

1.Write a python program to concatenate all elements in a list into a string and return it.

Solution:

In [30]:
```
def concatenate_list_data(list):
    result = ''
    for element in list:
        result += str(element)
    return result
print(concatenate_list_data(['I','joined','excelr','tutorial']))
```

```
Ijoinedexcelrtutorial
```

In [31]:
```
def concatenate_list_data(list):
    result = ''
    for element in list:
        result += str(element)
    return result
print(concatenate_list_data(['2071','89','78','34']))
```

```
2071897834
```

==========

DICTIONARY

1.Write a python script to concatenate following dictionaries to create a new one dict1={1:10,2:20} dict2={3:30,4:40} dict3={5:50,6:60}¶

Solution:

In [32]:
```
dict1={1:10,2:20}
dict2={3:30,4:40}
dict3={5:50,6:60}
dict4={}
for d in (dict1,dict2,dict3):dict4.update(d)
print(dict4)
```

```
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
```

==========

SERIES

1.Write a python to add ,subtract ,mutiple and divide two pandas Series.

Solution:

In [33]:
```
Sample_series=[2,4,6,8,10],[1,3,5,7,9]
```

In [34]:
```
import pandas as pd
ds1=pd.Series([2,4,6,8,10])
ds2=pd.Series([1,3,5,7,9])
ds=ds1+ds2
print("Add two series:")
print(ds)
print("Subract two series:")
ds=ds1-ds2
print(ds)
print("mutiply two series:")
ds=ds1*ds2
print(ds)
print("divide two series:")
ds=ds1/ds2
print(ds)
```

```
Add two series:
0     3
1     7
2    11
3    15
4    19
dtype: int64
Subract two series:
0    1
1    1
2    1
3    1
4    1
dtype: int64
mutiply two series:
0     2
1    12
2    30
3    56
4    90
dtype: int64
divide two series:
0    2.000000
1    1.333333
2    1.200000
3    1.142857
4    1.111111
dtype: float64
```

==========

DATA FRAME

1.Write a pandas program to select the specified columns and rows

from a given data frame.Go to the editor Sample Python dictionary data and list labels.

Select "name" and 'Score' columns in rows 1,3,5,6 from the following data frame

exam_data={'name':
['Anastasia','Dima',"katherine",'james','emily','Michael',"matthew",'laura','kevin','jonas' score:
[12.5,9,16.5,np,nan,9,20,14.5,np,nan,8,19], attempts:[1,3,2,3,2,3,1,1,2,1], quality:
['yes','no','yes','no','no','yes','yes','no','no','yes] labels=['a','b','c','d','e','f','g','h','i','j']

Solution:

In [33]:
```
import pandas as pd
import numpy as np

exam_data  = {'Name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Micha
        'Score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'Attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'Qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'ye
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data)
print("Select specific columns and rows:")
print(df.iloc[[1, 3, 5, 6], [0, 1]])
```

```
Select specific columns and rows:
      Name  Score
1     Dima    9.0
3    James    NaN
5  Michael   20.0
6  Matthew   14.5
```

2. Use Crime datasets from LMS

i) Find the aggregations like all moments of business decisions for all columns ,value counts. ii) do the plotting like plotting like histogram ,boxplot,scatterplot,barplot,piechart,dot chart

In [45]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plot
import seaborn as sns
```

In [46]:
```
crime_data=pd.read_csv("crime_data.csv")
crime_data
```

Out[46]:

	Unnamed: 0	Murder	Assault	UrbanPop	Rape
0	Alabama	13.2	236	58	21.2
1	Alaska	10.0	263	48	44.5
2	Arizona	8.1	294	80	31.0
3	Arkansas	8.8	190	50	19.5
4	California	9.0	276	91	40.6
5	Colorado	7.9	204	78	38.7
6	Connecticut	3.3	110	77	11.1
7	Delaware	5.9	238	72	15.8
8	Florida	15.4	335	80	31.9
9	Georgia	17.4	211	60	25.8
10	Hawaii	5.3	46	83	20.2
11	Idaho	2.6	120	54	14.2
12	Illinois	10.4	249	83	24.0
13	Indiana	7.2	113	65	21.0
14	Iowa	2.2	56	57	11.3
15	Kansas	6.0	115	66	18.0
16	Kentucky	9.7	109	52	16.3
17	Louisiana	15.4	249	66	22.2
18	Maine	2.1	83	51	7.8
19	Maryland	11.3	300	67	27.8
20	Massachusetts	4.4	149	85	16.3
21	Michigan	12.1	255	74	35.1
22	Minnesota	2.7	72	66	14.9
23	Mississippi	16.1	259	44	17.1
24	Missouri	9.0	178	70	28.2
25	Montana	6.0	109	53	16.4
26	Nebraska	4.3	102	62	16.5
27	Nevada	12.2	252	81	46.0
28	New Hampshire	2.1	57	56	9.5
29	New Jersey	7.4	159	89	18.8
30	New Mexico	11.4	285	70	32.1
31	New York	11.1	254	86	26.1
32	North Carolina	13.0	337	45	16.1
33	North Dakota	0.8	45	44	7.3

	Unnamed: 0	Murder	Assault	UrbanPop	Rape
34	Ohio	7.3	120	75	21.4
35	Oklahoma	6.6	151	68	20.0
36	Oregon	4.9	159	67	29.3
37	Pennsylvania	6.3	106	72	14.9
38	Rhode Island	3.4	174	87	8.3
39	South Carolina	14.4	279	48	22.5
40	South Dakota	3.8	86	45	12.8
41	Tennessee	13.2	188	59	26.9
42	Texas	12.7	201	80	25.5
43	Utah	3.2	120	80	22.9
44	Vermont	2.2	48	32	11.2
45	Virginia	8.5	156	63	20.7
46	Washington	4.0	145	73	26.2
47	West Virginia	5.7	81	39	9.3
48	Wisconsin	2.6	53	66	10.8
49	Wyoming	6.8	161	60	15.6

In [37]:
```
crime_data.describe()
```

Out[37]:

	Murder	Assault	UrbanPop	Rape
count	50.00000	50.000000	50.000000	50.000000
mean	7.78800	170.760000	65.540000	21.232000
std	4.35551	83.337661	14.474763	9.366385
min	0.80000	45.000000	32.000000	7.300000
25%	4.07500	109.000000	54.500000	15.075000
50%	7.25000	159.000000	66.000000	20.100000
75%	11.25000	249.000000	77.750000	26.175000
max	17.40000	337.000000	91.000000	46.000000

In [106]:
```
crime_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  50 non-null     object 
 1   Murder      50 non-null     float64
 2   Assault     50 non-null     int64  
 3   UrbanPop    50 non-null     int64  
 4   Rape        50 non-null     float64
dtypes: float64(2), int64(2), object(1)
memory usage: 2.1+ KB
```

In [108]:
```
crime_data.value_counts()
```

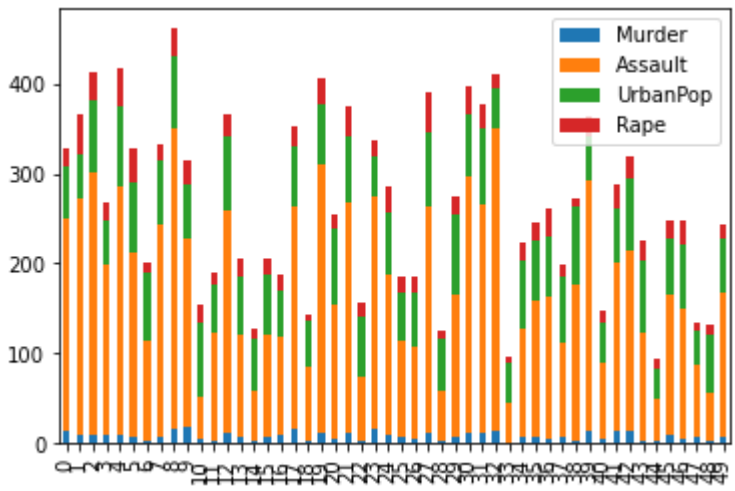
Out[108]:

Unnamed: 0	Murder	Assault	UrbanPop	Rape	
Alabama	13.2	236	58	21.2	1
Pennsylvania	6.3	106	72	14.9	1
Nevada	12.2	252	81	46.0	1
New Hampshire	2.1	57	56	9.5	1
New Jersey	7.4	159	89	18.8	1
New Mexico	11.4	285	70	32.1	1
New York	11.1	254	86	26.1	1
North Carolina	13.0	337	45	16.1	1
North Dakota	0.8	45	44	7.3	1
Ohio	7.3	120	75	21.4	1
Oklahoma	6.6	151	68	20.0	1
Oregon	4.9	159	67	29.3	1
Rhode Island	3.4	174	87	8.3	1
Alaska	10.0	263	48	44.5	1
South Carolina	14.4	279	48	22.5	1
South Dakota	3.8	86	45	12.8	1
Tennessee	13.2	188	59	26.9	1
Texas	12.7	201	80	25.5	1
Utah	3.2	120	80	22.9	1
Vermont	2.2	48	32	11.2	1
Virginia	8.5	156	63	20.7	1
Washington	4.0	145	73	26.2	1
West Virginia	5.7	81	39	9.3	1
Wisconsin	2.6	53	66	10.8	1
Nebraska	4.3	102	62	16.5	1
Montana	6.0	109	53	16.4	1
Missouri	9.0	178	70	28.2	1
Mississippi	16.1	259	44	17.1	1
Arizona	8.1	294	80	31.0	1
Arkansas	8.8	190	50	19.5	1
California	9.0	276	91	40.6	1
Colorado	7.9	204	78	38.7	1
Connecticut	3.3	110	77	11.1	1
Delaware	5.9	238	72	15.8	1
Florida	15.4	335	80	31.9	1
Georgia	17.4	211	60	25.8	1
Hawaii	5.3	46	83	20.2	1
Idaho	2.6	120	54	14.2	1
Illinois	10.4	249	83	24.0	1
Indiana	7.2	113	65	21.0	1
Iowa	2.2	56	57	11.3	1
Kansas	6.0	115	66	18.0	1
Kentucky	9.7	109	52	16.3	1
Louisiana	15.4	249	66	22.2	1
Maine	2.1	83	51	7.8	1
Maryland	11.3	300	67	27.8	1
Massachusetts	4.4	149	85	16.3	1
Michigan	12.1	255	74	35.1	1
Minnesota	2.7	72	66	14.9	1
Wyoming	6.8	161	60	15.6	1

dtype: int64

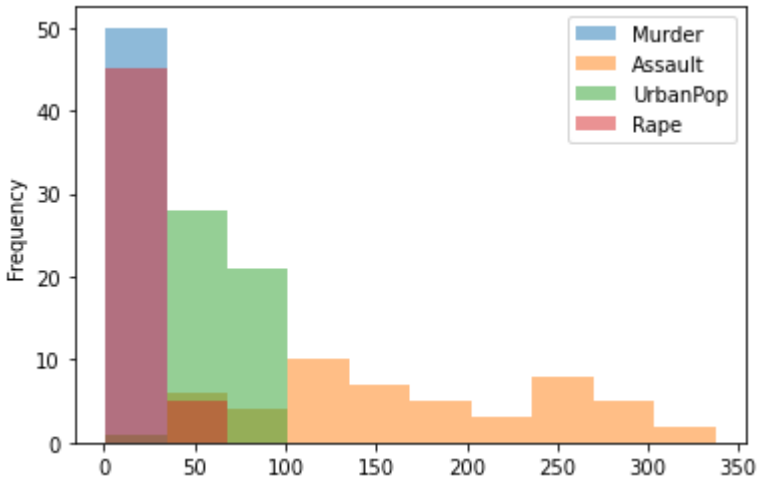
In [49]:
```
crime_data.plot.bar(stacked=True)
```

Out[49]: <AxesSubplot:>



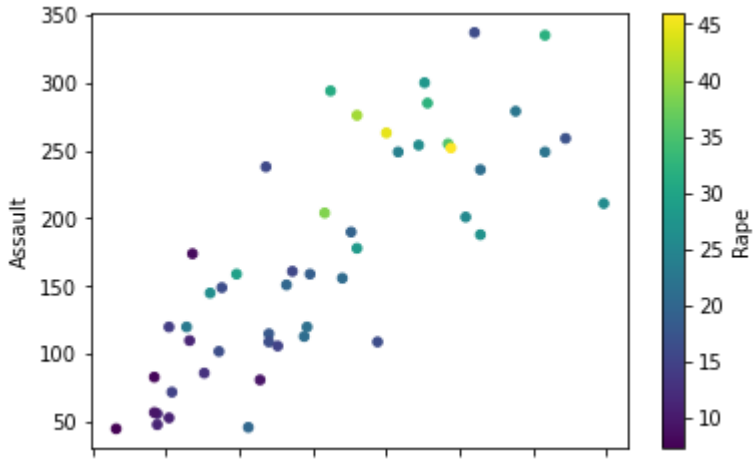
In [51]:
```
crime_data.plot.hist(alpha=0.5)
```

Out[51]: <AxesSubplot:ylabel='Frequency'>

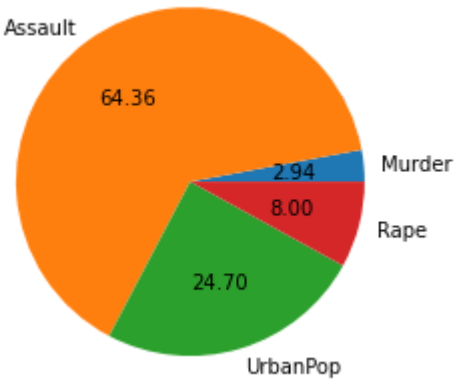


In [58]:
```
crime_data.plot.scatter(x='Murder',y="Assault",c="Rape",colormap='viridis')
```

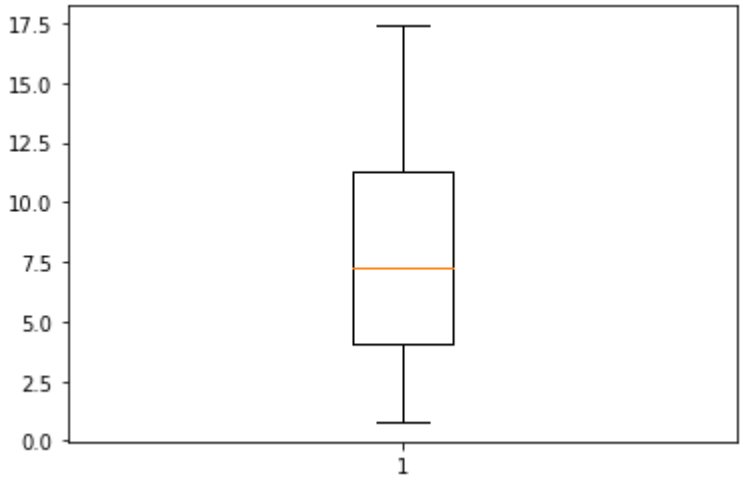
Out[58]: <AxesSubplot:xlabel='Murder', ylabel='Assault'>



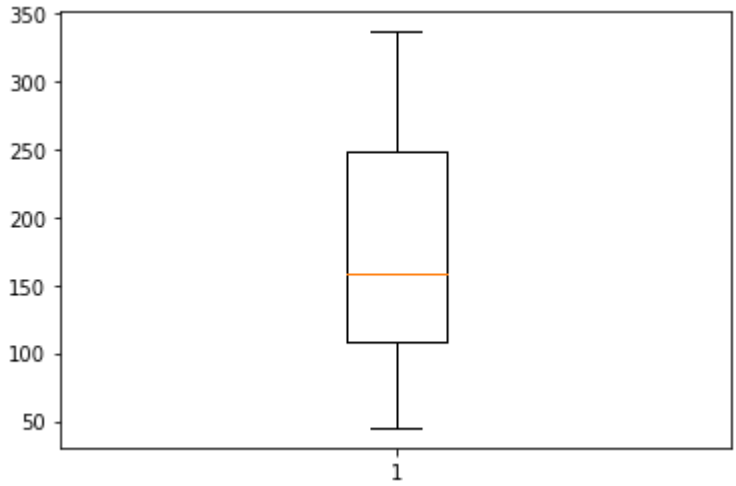
In [81]:
```
mean=7.78800,170.760000,65.540000,21.232000
label='Murder','Assault','UrbanPop','Rape'
plot.pie(mean, labels = label, autopct = "%.2f")
plot.show()
```



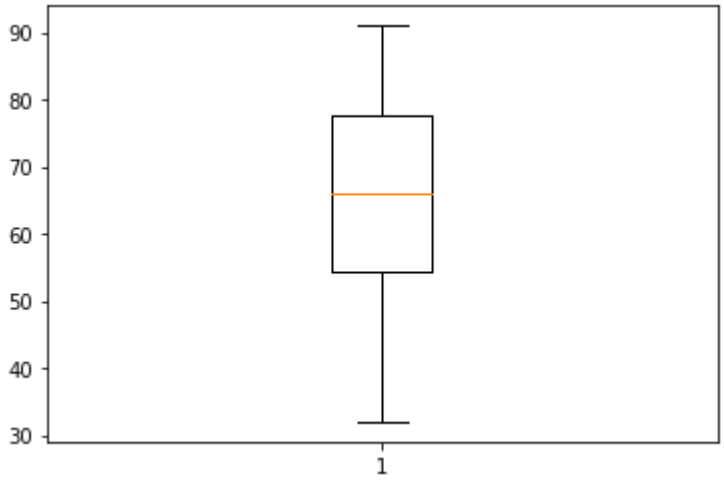
In [74]:
```
plot.boxplot(crime_data['Murder'])
plot.show()
```



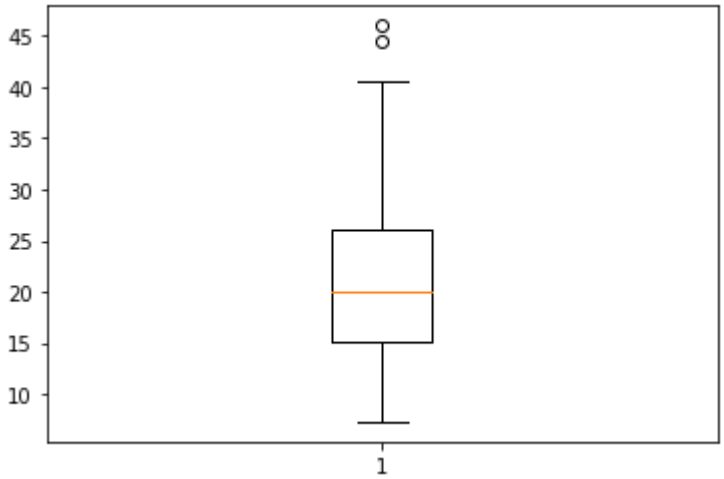
In [75]:
```
plot.boxplot(crime_data['Assault'])
plot.show()
```



In [76]:
```
plot.boxplot(crime_data['UrbanPop'])
plot.show()
```

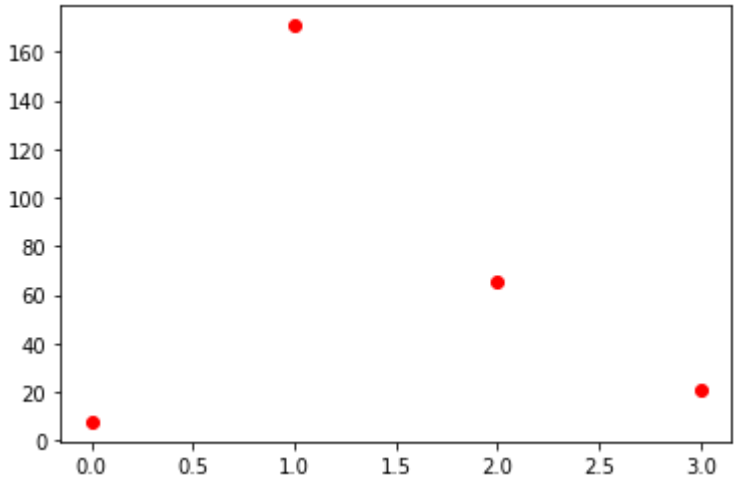


In [77]:
```
plot.boxplot(crime_data['Rape'])
plot.show()
```

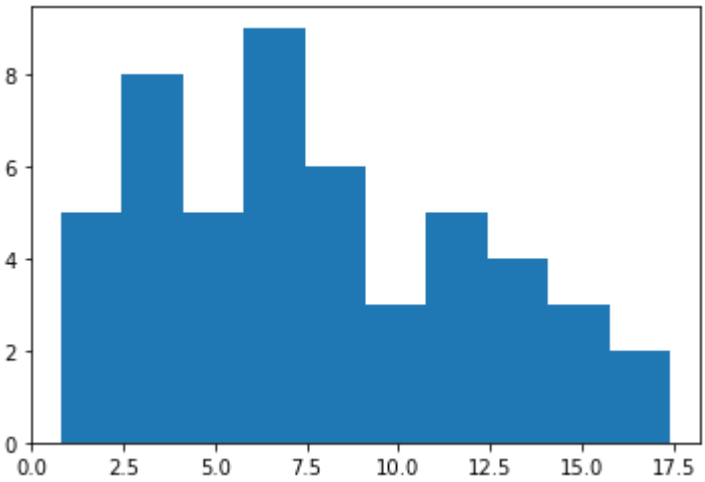


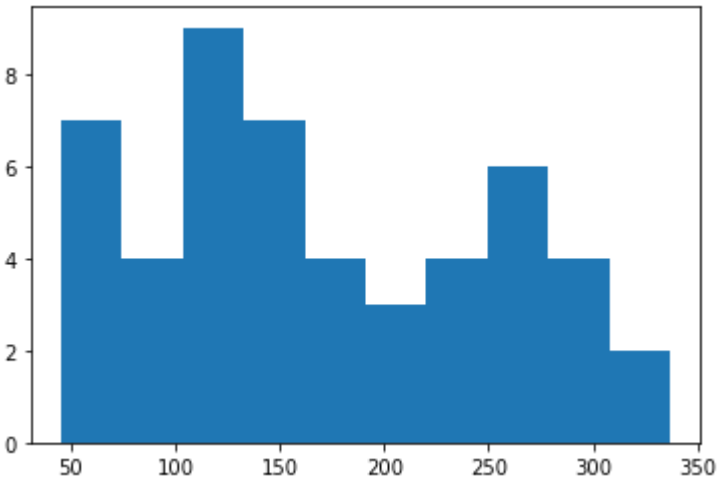
In [103]:
```
plot.plot([7.78800,170.760000,65.540000,21.232000], 'ro')
```

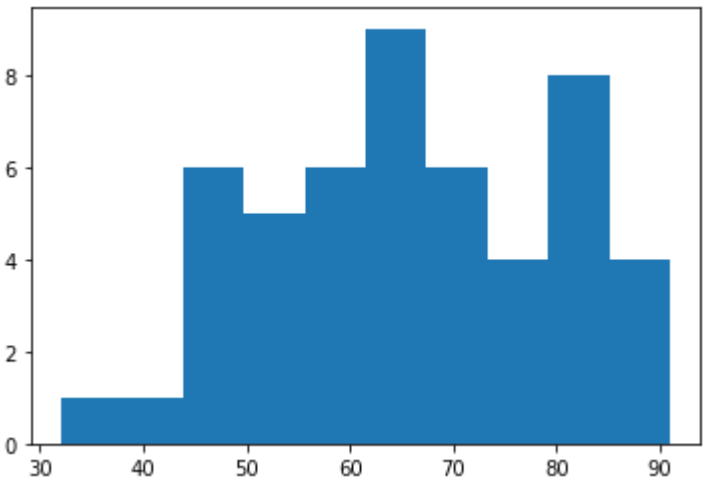
Out[103]:
```
[<matplotlib.lines.Line2D at 0x2775fb1e940>]
```

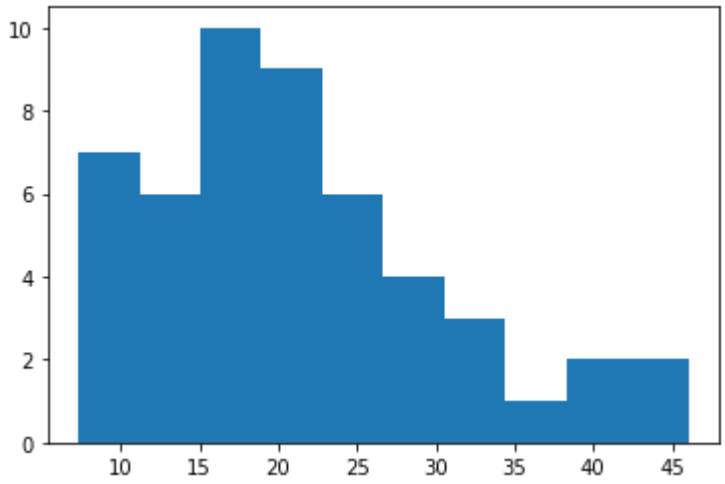


In [38]:
```
plot.hist(x='Murder',data=crime_data,)
plot.show()
plot.hist(x="Assault",data=crime_data)
plot.show()
plot.hist(x='UrbanPop',data=crime_data)
plot.show()
plot.hist(x='Rape',data=crime_data)
plot.show()
```









3. Use mtcars dataset from LMS

A) delete /drop rows 10 to 15 of all columns B) drop the VOL column c) Write the for loop to get value_counts of all columns

In [4]:
```
import pandas as pd
mtcars_data=pd.read_csv('mtcars.csv')
mtcars_data
```

Out[4]:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
5	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
6	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
7	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
8	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
9	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
10	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
11	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
12	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
13	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
14	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
15	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
16	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
17	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
18	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
19	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
20	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
21	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
22	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
23	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
24	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
25	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
26	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
27	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
28	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
29	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
30	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8
31	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2

In [5]:
```
mtcars_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 11 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   mpg     32 non-null     float64
 1   cyl     32 non-null     int64  
 2   disp    32 non-null     float64
 3   hp      32 non-null     int64  
 4   drat    32 non-null     float64
 5   wt      32 non-null     float64
 6   qsec    32 non-null     float64
 7   vs      32 non-null     int64  
 8   am      32 non-null     int64  
 9   gear    32 non-null     int64  
 10  carb    32 non-null     int64  
dtypes: float64(5), int64(6)
memory usage: 2.9 KB
```

In [40]:
```
mtcars1 = mtcars_data.drop(labels=[10,11,12,13,14,15], axis=0)
mtcars1
```

Out[40]:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
5	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
6	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
7	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
8	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
9	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
16	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
17	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
18	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
19	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
20	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
21	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
22	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
23	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
24	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
25	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
26	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
27	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
28	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
29	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
30	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8
31	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2

In [12]:
```
mtcars1=mtcars_data.drop(columns=['vs'], axis=0)
mtcars1
```

Out[12]:

	mpg	cyl	disp	hp	drat	wt	qsec	am	gear	carb
0	21.0	6	160.0	110	3.90	2.620	16.46	1	4	4
1	21.0	6	160.0	110	3.90	2.875	17.02	1	4	4
2	22.8	4	108.0	93	3.85	2.320	18.61	1	4	1
3	21.4	6	258.0	110	3.08	3.215	19.44	0	3	1
4	18.7	8	360.0	175	3.15	3.440	17.02	0	3	2
5	18.1	6	225.0	105	2.76	3.460	20.22	0	3	1
6	14.3	8	360.0	245	3.21	3.570	15.84	0	3	4
7	24.4	4	146.7	62	3.69	3.190	20.00	0	4	2
8	22.8	4	140.8	95	3.92	3.150	22.90	0	4	2
9	19.2	6	167.6	123	3.92	3.440	18.30	0	4	4
10	17.8	6	167.6	123	3.92	3.440	18.90	0	4	4
11	16.4	8	275.8	180	3.07	4.070	17.40	0	3	3
12	17.3	8	275.8	180	3.07	3.730	17.60	0	3	3
13	15.2	8	275.8	180	3.07	3.780	18.00	0	3	3
14	10.4	8	472.0	205	2.93	5.250	17.98	0	3	4
15	10.4	8	460.0	215	3.00	5.424	17.82	0	3	4
16	14.7	8	440.0	230	3.23	5.345	17.42	0	3	4
17	32.4	4	78.7	66	4.08	2.200	19.47	1	4	1
18	30.4	4	75.7	52	4.93	1.615	18.52	1	4	2
19	33.9	4	71.1	65	4.22	1.835	19.90	1	4	1
20	21.5	4	120.1	97	3.70	2.465	20.01	0	3	1
21	15.5	8	318.0	150	2.76	3.520	16.87	0	3	2
22	15.2	8	304.0	150	3.15	3.435	17.30	0	3	2
23	13.3	8	350.0	245	3.73	3.840	15.41	0	3	4
24	19.2	8	400.0	175	3.08	3.845	17.05	0	3	2
25	27.3	4	79.0	66	4.08	1.935	18.90	1	4	1
26	26.0	4	120.3	91	4.43	2.140	16.70	1	5	2
27	30.4	4	95.1	113	3.77	1.513	16.90	1	5	2
28	15.8	8	351.0	264	4.22	3.170	14.50	1	5	4
29	19.7	6	145.0	175	3.62	2.770	15.50	1	5	6
30	15.0	8	301.0	335	3.54	3.570	14.60	1	5	8
31	21.4	4	121.0	109	4.11	2.780	18.60	1	4	2

In [38]:
```
data=mtcars_data.value_counts
data
```

Out[38]:
```
<bound method DataFrame.value_counts of      mpg  cyl   disp   hp  drat     wt
qsec  vs  am  gear  carb
0   21.0    6  160.0  110  3.90  2.620  16.46   0   1     4     4
1   21.0    6  160.0  110  3.90  2.875  17.02   0   1     4     4
2   22.8    4  108.0   93  3.85  2.320  18.61   1   1     4     1
3   21.4    6  258.0  110  3.08  3.215  19.44   1   0     3     1
4   18.7    8  360.0  175  3.15  3.440  17.02   0   0     3     2
5   18.1    6  225.0  105  2.76  3.460  20.22   1   0     3     1
6   14.3    8  360.0  245  3.21  3.570  15.84   0   0     3     4
7   24.4    4  146.7   62  3.69  3.190  20.00   1   0     4     2
8   22.8    4  140.8   95  3.92  3.150  22.90   1   0     4     2
9   19.2    6  167.6  123  3.92  3.440  18.30   1   0     4     4
10  17.8    6  167.6  123  3.92  3.440  18.90   1   0     4     4
11  16.4    8  275.8  180  3.07  4.070  17.40   0   0     3     3
12  17.3    8  275.8  180  3.07  3.730  17.60   0   0     3     3
13  15.2    8  275.8  180  3.07  3.780  18.00   0   0     3     3
14  10.4    8  472.0  205  2.93  5.250  17.98   0   0     3     4
15  10.4    8  460.0  215  3.00  5.424  17.82   0   0     3     4
16  14.7    8  440.0  230  3.23  5.345  17.42   0   0     3     4
17  32.4    4   78.7   66  4.08  2.200  19.47   1   1     4     1
18  30.4    4   75.7   52  4.93  1.615  18.52   1   1     4     2
19  33.9    4   71.1   65  4.22  1.835  19.90   1   1     4     1
20  21.5    4  120.1   97  3.70  2.465  20.01   1   0     3     1
21  15.5    8  318.0  150  2.76  3.520  16.87   0   0     3     2
22  15.2    8  304.0  150  3.15  3.435  17.30   0   0     3     2
23  13.3    8  350.0  245  3.73  3.840  15.41   0   0     3     4
24  19.2    8  400.0  175  3.08  3.845  17.05   0   0     3     2
25  27.3    4   79.0   66  4.08  1.935  18.90   1   1     4     1
26  26.0    4  120.3   91  4.43  2.140  16.70   0   1     5     2
27  30.4    4   95.1  113  3.77  1.513  16.90   1   1     5     2
28  15.8    8  351.0  264  4.22  3.170  14.50   0   1     5     4
29  19.7    6  145.0  175  3.62  2.770  15.50   0   1     5     6
30  15.0    8  301.0  335  3.54  3.570  14.60   0   1     5     8
31  21.4    4  121.0  109  4.11  2.780  18.60   1   1     4     2>
```

4.Use Bank Datset from LMS

A) Change all the categorical columns into numerical by creating Dummies and using label encoder B)rename all the columns names DF c) Rename only one specific columns in DF

In [26]:
```
import pandas as pd
```

In [35]:
```
bank=pd.read_csv('bank-full.csv',sep=";")
bank
```

Out[35]:

	age	job	marital	education	default	balance	housing	loan	contact	day	mor
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	m
1	44	technician	single	secondary	no	29	yes	no	unknown	5	m
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	m
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	m
4	33	unknown	single	unknown	no	1	no	no	unknown	5	m
...	...	...	...	...	...	...	...	...	...	...	
45206	51	technician	married	tertiary	no	825	no	no	cellular	17	r
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17	r
45208	72	retired	married	secondary	no	5715	no	no	cellular	17	r
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17	r
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17	r

45211 rows × 17 columns

In [28]:
```
bank.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        45211 non-null  int64 
 1   job        45211 non-null  object
 2   marital    45211 non-null  object
 3   education  45211 non-null  object
 4   default    45211 non-null  object
 5   balance    45211 non-null  int64 
 6   housing    45211 non-null  object
 7   loan       45211 non-null  object
 8   contact    45211 non-null  object
 9   day        45211 non-null  int64 
 10  month      45211 non-null  object
 11  duration   45211 non-null  int64 
 12  campaign   45211 non-null  int64 
 13  pdays      45211 non-null  int64 
 14  previous   45211 non-null  int64 
 15  poutcome   45211 non-null  object
 16  y          45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

In [29]:
```
bank1=pd.get_dummies(bank,columns=['job','marital','education','contact','poutcom
bank1
```

Out[29]:

	age	default	balance	housing	loan	day	month	duration	campaign	pdays	...	educatio
0	58	no	2143	yes	no	5	may	261	1	-1	...	
1	44	no	29	yes	no	5	may	151	1	-1	...	
2	33	no	2	yes	yes	5	may	76	1	-1	...	
3	47	no	1506	yes	no	5	may	92	1	-1	...	
4	33	no	1	no	no	5	may	198	1	-1	...	
...	...	...	...	...	...	...	...	...	...	...	...	
45206	51	no	825	no	no	17	nov	977	3	-1	...	
45207	71	no	1729	no	no	17	nov	456	2	-1	...	
45208	72	no	5715	no	no	17	nov	1127	5	184	...	
45209	57	no	668	no	no	17	nov	508	4	-1	...	
45210	37	no	2971	no	no	17	nov	361	2	188	...	

45211 rows × 38 columns

In [30]:
```
bank1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 38 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   age                  45211 non-null  int64 
 1   default              45211 non-null  object
 2   balance              45211 non-null  int64 
 3   housing              45211 non-null  object
 4   loan                 45211 non-null  object
 5   day                  45211 non-null  int64 
 6   month                45211 non-null  object
 7   duration             45211 non-null  int64 
 8   campaign             45211 non-null  int64 
 9   pdays                45211 non-null  int64 
 10  previous             45211 non-null  int64 
 11  y                    45211 non-null  object
 12  job_admin.           45211 non-null  uint8 
 13  job_blue-collar      45211 non-null  uint8 
 14  job_entrepreneur     45211 non-null  uint8 
 15  job_housemaid        45211 non-null  uint8 
 16  job_management       45211 non-null  uint8 
 17  job_retired          45211 non-null  uint8 
 18  job_self-employed    45211 non-null  uint8 
 19  job_services         45211 non-null  uint8 
 20  job_student          45211 non-null  uint8 
 21  job_technician       45211 non-null  uint8 
 22  job_unemployed       45211 non-null  uint8 
 23  job_unknown          45211 non-null  uint8 
 24  marital_divorced     45211 non-null  uint8 
 25  marital_married      45211 non-null  uint8 
 26  marital_single       45211 non-null  uint8 
 27  education_primary    45211 non-null  uint8 
 28  education_secondary  45211 non-null  uint8 
 29  education_tertiary   45211 non-null  uint8 
 30  education_unknown    45211 non-null  uint8 
 31  contact_cellular     45211 non-null  uint8 
 32  contact_telephone    45211 non-null  uint8 
 33  contact_unknown      45211 non-null  uint8 
 34  poutcome_failure     45211 non-null  uint8 
 35  poutcome_other       45211 non-null  uint8 
 36  poutcome_success     45211 non-null  uint8 
 37  poutcome_unknown     45211 non-null  uint8 
dtypes: int64(7), object(5), uint8(26)
memory usage: 5.3+ MB
```

In [41]:
```
bank5=bank.rename(columns = {'education':'EDUCATION'}, inplace = True)
bank5
```

5.After doing all the change in bank data(Q.19) save the file in your directory in csv Fomat

solution:

In [22]:
```
import pandas as pd
export=bank1.to_csv('bank_csv')
export
```

In [24]:
```
bank_new =pd.read_csv("bank_csv")
bank_new
```

Out[24]:

	Unnamed: 0	age	default	balance	housing	loan	day	month	duration	campaign	...	educ
0	0	58	no	2143	yes	no	5	may	261	1	...	
1	1	44	no	29	yes	no	5	may	151	1	...	
2	2	33	no	2	yes	yes	5	may	76	1	...	
3	3	47	no	1506	yes	no	5	may	92	1	...	
4	4	33	no	1	no	no	5	may	198	1	...	
...	...	...	...	...	...	...	...	...	...	...	...	
45206	45206	51	no	825	no	no	17	nov	977	3	...	
45207	45207	71	no	1729	no	no	17	nov	456	2	...	
45208	45208	72	no	5715	no	no	17	nov	1127	5	...	
45209	45209	57	no	668	no	no	17	nov	508	4	...	
45210	45210	37	no	2971	no	no	17	nov	361	2	...	

45211 rows × 39 columns

==========

BASIC PROGRAMS

1.Write python program to use various operator in python

1. Arithematic operator

2. Comparsion Operator

3. Logical operator

4. Bitwise Operator

5. Assignment Operator

6.Special Operator

a) Identity Operator

b) Membership Operator

Solution:

In [45]:
```
x = 15
y = 4

# Output: x + y = 19
print('x + y =',x+y)

# Output: x - y = 11
print('x - y =',x-y)

# Output: x * y = 60
print('x * y =',x*y)

# Output: x / y = 3.75
print('x / y =',x/y)

# Output: x // y = 3
print('x // y =',x//y)

# Output: x ** y = 50625
print('x ** y =',x**y)
```

```
x + y = 19
x - y = 11
x * y = 60
x / y = 3.75
x // y = 3
x ** y = 50625
```

In [46]:
```
x = 10
y = 12

# Output: x > y is False
print('x > y is',x>y)

# Output: x < y is True
print('x < y is',x<y)

# Output: x == y is False
print('x == y is',x==y)

# Output: x != y is True
print('x != y is',x!=y)

# Output: x >= y is False
print('x >= y is',x>=y)

# Output: x <= y is True
print('x <= y is',x<=y)
```

```
x > y is False
x < y is True
x == y is False
x != y is True
x >= y is False
x <= y is True
```

In [47]:
```
x = True
y = False

print('x and y is',x and y)

print('x or y is',x or y)

print('not x is',not x)
```

```
x and y is False
x or y is True
not x is False
```

In [48]:
```
x1 = 5
y1 = 5
x2 = 'Hello'
y2 = 'Hello'
x3 = [1,2,3]
y3 = [1,2,3]

# Output: False
print(x1 is not y1)

# Output: True
print(x2 is y2)

# Output: False
print(x3 is y3)
```

```
False
True
False
```

In [49]:
```
x = 'Hello world'
y = {1:'a',2:'b'}

# Output: True
print('H' in x)

# Output: True
print('hello' not in x)

# Output: True
print(1 in y)

# Output: False
print('a' in y)
```

```
True
True
True
False
```

2.Create listof elements and slice and dice it

Solution:

In [50]:
```
# Initialize list
List = [1, 2, 3, 4, 5, 6, 7, 8, 9]

# Show original list
print("\nOriginal List:\n", List)

print("\nSliced Lists: ")

# Display sliced list
print(List[3:9:2])

# Display sliced list
print(List[::2])

# Display sliced list
print(List[::])
```

```
Original List:
 [1, 2, 3, 4, 5, 6, 7, 8, 9]

Sliced Lists:
[4, 6, 8]
[1, 3, 5, 7, 9]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [51]:
```
# Initialize list
List = ['Geeks', 4, 'geeks !']

# Show original list
print("\nOriginal List:\n", List)

print("\nSliced Lists: ")

# Display sliced list
print(List[::-1])

# Display sliced list
print(List[::-3])

# Display sliced list
print(List[:1:-2])
```

```
Original List:
 ['Geeks', 4, 'geeks !']

Sliced Lists:
['geeks !', 4, 'Geeks']
['geeks !']
['geeks !']
```

3. Using while loop accept numbers until sum of number is less than 100

In [ ]:
```
total=0
a=int(input())
while a>=100:
    total=total+a
    if total<=100:
        print("total sum is ",total)
    else:
        print("total sum is not equal to ",total)
```

```
988
total sum is not equal to  988
total sum is not equal to  1976
total sum is not equal to  2964
total sum is not equal to  3952
total sum is not equal to  4940
total sum is not equal to  5928
total sum is not equal to  6916
total sum is not equal to  7904
total sum is not equal to  8892
total sum is not equal to  9880
total sum is not equal to  10868
total sum is not equal to  11856
total sum is not equal to  12844
total sum is not equal to  13832
total sum is not equal to  14820
total sum is not equal to  15808
total sum is not equal to  16796
total sum is not equal to  17784
```

Solution:

In [23]:
```
total_sum =0
a=int(input())
while a!=0:
    total_sum +=a
    if total_sum <=100:
        print('Total sum is',total_sum)
        break
        a=int(input())
    else:
        print('Total sum is less than 100 and is equal to',total_sum,'.')
```

```
45
Total sum is 45
```

4.Write a Python program Read & write Excel files

In [1]:
```
pip install openpyxl
```

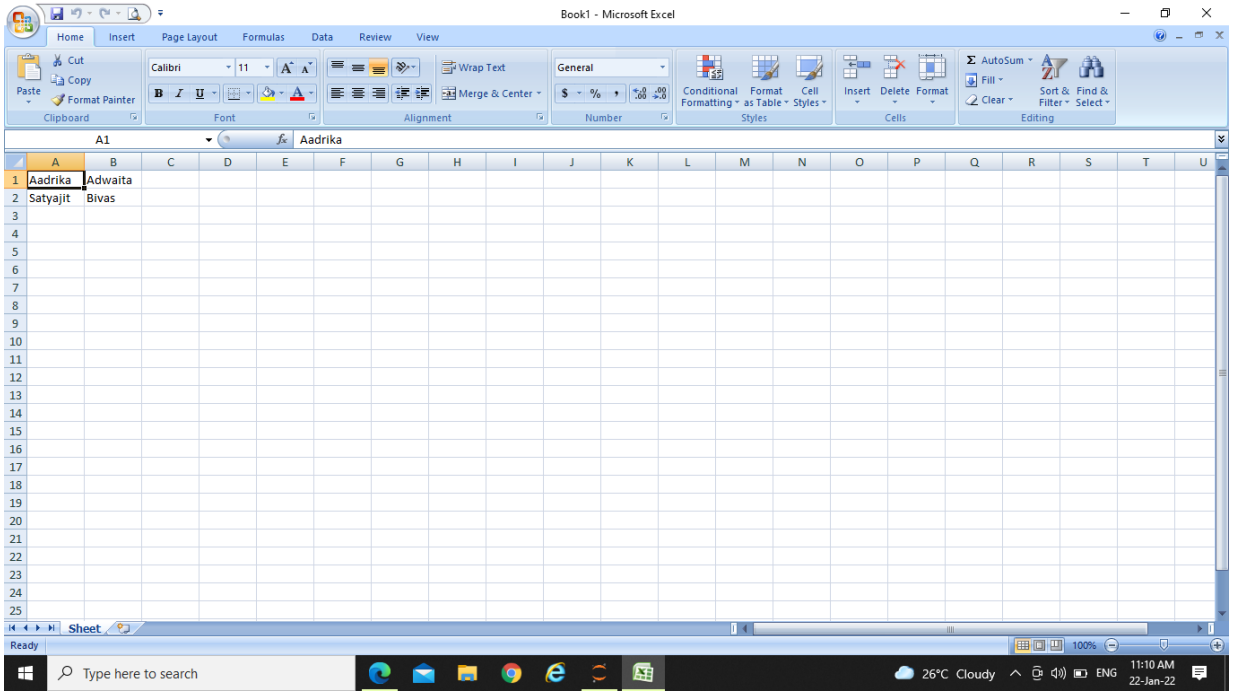
```
Requirement already satisfied: openpyxl in c:\users\ostrich_sales_mgmt\anaconda
3\lib\site-packages (3.0.9)
Requirement already satisfied: et-xmlfile in c:\users\ostrich_sales_mgmt\anacon
da3\lib\site-packages (from openpyxl) (1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
import openpyxl
my_wb = openpyxl.Workbook()
my_sheet = my_wb.active

c1 = my_sheet.cell(row = 1, column = 1)
c1.value = "Aadrika"
c2 = my_sheet.cell(row= 1 , column = 2)
c2.value = "Adwaita"
c3 = my_sheet['A2']
c3.value = "Satyajit"
# B2 = column = 2 & row = 2.
c4 = my_sheet['B2']
c4.value = "Bivas"

my_wb.save("Desktop\Book1.xlsx")
```

Solution: the Excel sheet will be display in Desptop name as book 1 .xlsx



5.Write a python program to scrape reviews from a commerical web site

Solution:

In [3]:
```
import requests
#Link ="https://en.wikipedia.org/wiki/Forbes_list_of_Indian_billionaires"
Link = "https://en.wikipedia.org/wiki/List_of_Indian_people_by_net_worth"
Link_text=requests.get(Link).text
print(Link_text)
```

```
<!DOCTYPE html>
<html class="client-nojs" lang="en" dir="ltr">
<head>
<meta charset="UTF-8"/>
<title>List of Indian people by net worth - Wikipedia</title>
<script>document.documentElement.className="client-js";RLCONF={"wgBreakFrame
s":false,"wgSeparatorTransformTable":["",""],"wgDigitTransformTable":
["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","January","Februar
y","March","April","May","June","July","August","September","October","Novemb
er","December"],"wgRequestId":"e35659a6-fbb1-4f6a-b2e7-3de8506e6f13","wgCSPNo
nce":false,"wgCanonicalNamespace":"","wgCanonicalSpecialPageName":false,"wgNa
mespaceNumber":0,"wgPageName":"List_of_Indian_people_by_net_worth","wgTitl
e":"List of Indian people by net worth","wgCurRevisionId":1066369968,"wgRevis
ionId":1066369968,"wgArticleId":61238433,"wgIsArticle":true,"wgIsRedirect":fa
lse,"wgAction":"view","wgUserName":null,"wgUserGroups":["*"],"wgCategories":
["CS1 maint: url-status","Wikipedia indefinitely semi-protected pages","Artic
les with short description","Short description is different from Wikidata","I
ndian billionaires","Demographics of India","Lists of Indian people","Lists o
f people by wealth",
```

In [4]:
```
from bs4 import BeautifulSoup
soup=BeautifulSoup(Link_text,'lxml')
print(soup)
```

```
<!DOCTYPE html>
<html class="client-nojs" dir="ltr" lang="en">
<head>
<meta charset="utf-8"/>
<title>List of Indian people by net worth - Wikipedia</title>
<script>document.documentElement.className="client-js";RLCONF={"wgBreakFrame
s":false,"wgSeparatorTransformTable":["",""],"wgDigitTransformTable":
["",""],"wgDefaultDateFormat":"dmy","wgMonthNames":["","January","Februar
y","March","April","May","June","July","August","September","October","Novemb
er","December"],"wgRequestId":"e35659a6-fbb1-4f6a-b2e7-3de8506e6f13","wgCSPNo
nce":false,"wgCanonicalNamespace":"","wgCanonicalSpecialPageName":false,"wgNa
mespaceNumber":0,"wgPageName":"List_of_Indian_people_by_net_worth","wgTitl
e":"List of Indian people by net worth","wgCurRevisionId":1066369968,"wgRevis
ionId":1066369968,"wgArticleId":61238433,"wgIsArticle":true,"wgIsRedirect":fa
lse,"wgAction":"view","wgUserName":null,"wgUserGroups":["*"],"wgCategories":
["CS1 maint: url-status","Wikipedia indefinitely semi-protected pages","Artic
les with short description","Short description is different from Wikidata","I
ndian billionaires","Demographics of India","Lists of Indian people","Lists o
f people by wealth",
```

In [5]:
```
print(soup.prettify())
```

```
<!DOCTYPE html>
<html class="client-nojs" dir="ltr" lang="en">
 <head>
  <meta charset="utf-8"/>
  <title>
   List of Indian people by net worth - Wikipedia
  </title>
  <script>
   document.documentElement.className="client-js";RLCONF={"wgBreakFrames":fal
se,"wgSeparatorTransformTable":["",""],"wgDigitTransformTable":["",""],"wgDef
aultDateFormat":"dmy","wgMonthNames":["","January","February","March","Apri
l","May","June","July","August","September","October","November","Decembe
r"],"wgRequestId":"e35659a6-fbb1-4f6a-b2e7-3de8506e6f13","wgCSPNonce":fals
e,"wgCanonicalNamespace":"","wgCanonicalSpecialPageName":false,"wgNamespaceNu
mber":0,"wgPageName":"List_of_Indian_people_by_net_worth","wgTitle":"List of
Indian people by net worth","wgCurRevisionId":1066369968,"wgRevisionId":10663
69968,"wgArticleId":61238433,"wgIsArticle":true,"wgIsRedirect":false,"wgActio
n":"view","wgUserName":null,"wgUserGroups":["*"],"wgCategories":["CS1 maint:
url-status","Wikipedia indefinitely semi-protected pages","Articles with shor
```

In [6]:
```
print(soup.title)
```

```
<title>List of Indian people by net worth - Wikipedia</title>
```

In [7]:
```
print(soup.title.string)
```

```
List of Indian people by net worth - Wikipedia
```

In [8]:
```
soup.a
```

Out[8]:
```
<a id="top"></a>
```

In [9]:
```
soup.find_all('a')
```

Out[9]:
```
[<a id="top"></a>,
 <a href="/wiki/Wikipedia:Protection_policy#semi" title="This article is semi
-protected."><img alt="Page semi-protected" data-file-height="512" data-file-
width="512" decoding="async" height="20" src="//upload.wikimedia.org/wikipedi
a/en/thumb/1/1b/Semi-protection-shackle.svg/20px-Semi-protection-shackle.svg.
png" srcset="//upload.wikimedia.org/wikipedia/en/thumb/1/1b/Semi-protection-s
hackle.svg/30px-Semi-protection-shackle.svg.png 1.5x, //upload.wikimedia.org/
wikipedia/en/thumb/1/1b/Semi-protection-shackle.svg/40px-Semi-protection-shac
kle.svg.png 2x" width="20"/></a>,
 <a class="mw-jump-link" href="#mw-head">Jump to navigation</a>,
 <a class="mw-jump-link" href="#searchInput">Jump to search</a>,
 <a href="/wiki/Forbes" title="Forbes">Forbes</a>,
 <a href="/wiki/United_States" title="United States">United States</a>,
 <a href="/wiki/China" title="China">China</a>,
 <a href="#cite_note-1">[1]</a>,
 <a href="/wiki/Mukesh_Ambani" title="Mukesh Ambani">Mukesh Ambani</a>,
 <a href="#cite_note-2">[2]</a>,
 <a href="#cite_note-3">[3]</a>,
 <a href="/wiki/Savitri_Jindal" title="Savitri Jindal">Savitri Jindal</a>,
```

In [10]:
```
print(soup.title.string)
```

```
List of Indian people by net worth - Wikipedia
```

In [11]:
```
soup.a
```

Out[11]:
```
<a id="top"></a>
```

In [12]:
```
soup.find_all('a')
```

Out[12]:
```
[<a id="top"></a>,
 <a href="/wiki/Wikipedia:Protection_policy#semi" title="This article is semi
-protected."><img alt="Page semi-protected" data-file-height="512" data-file-
width="512" decoding="async" height="20" src="//upload.wikimedia.org/wikipedi
a/en/thumb/1/1b/Semi-protection-shackle.svg/20px-Semi-protection-shackle.svg.
png" srcset="//upload.wikimedia.org/wikipedia/en/thumb/1/1b/Semi-protection-s
hackle.svg/30px-Semi-protection-shackle.svg.png 1.5x, //upload.wikimedia.org/
wikipedia/en/thumb/1/1b/Semi-protection-shackle.svg/40px-Semi-protection-shac
kle.svg.png 2x" width="20"/></a>,
 <a class="mw-jump-link" href="#mw-head">Jump to navigation</a>,
 <a class="mw-jump-link" href="#searchInput">Jump to search</a>,
 <a href="/wiki/Forbes" title="Forbes">Forbes</a>,
 <a href="/wiki/United_States" title="United States">United States</a>,
 <a href="/wiki/China" title="China">China</a>,
 <a href="#cite_note-1">[1]</a>,
 <a href="/wiki/Mukesh_Ambani" title="Mukesh Ambani">Mukesh Ambani</a>,
 <a href="#cite_note-2">[2]</a>,
 <a href="#cite_note-3">[3]</a>,
 <a href="/wiki/Savitri_Jindal" title="Savitri Jindal">Savitri Jindal</a>,
```

In [13]:
```
all_table =soup.find_all('table')
print(all_table)
```

```
[<table class="wikitable sortable">
<tbody><tr>
<th>Rank
</th>
<th>Name
</th>
<th>Wealth
<p>Change
</p>
</th>
<th><a href="/wiki/Net_worth" title="Net worth">Net worth</a> (<a class="mw-r
edirect" href="/wiki/USD" title="USD">USD</a>)
</th>
<th>Company
</th>
<th>Sources of wealth
</th></tr>
<tr>
<td>1
```

In [14]:
```
our_table=soup.find('table', class_= "wikitable sortable")
print(our_table)
```

```
<table class="wikitable sortable">
<tbody><tr>
<th>Rank
</th>
<th>Name
</th>
<th>Wealth
<p>Change
</p>
</th>
<th><a href="/wiki/Net_worth" title="Net worth">Net worth</a> (<a class="mw-r
edirect" href="/wiki/USD" title="USD">USD</a>)
</th>
<th>Company
</th>
<th>Sources of wealth
</th></tr>
<tr>
<td>1
```

In [15]:
```
table_links =our_table.find_all('a')
print(table_links)
```

```
[<a href="/wiki/Net_worth" title="Net worth">Net worth</a>, <a class="mw-redire
ct" href="/wiki/USD" title="USD">USD</a>, <a href="/wiki/Mukesh_Ambani" title
="Mukesh Ambani">Mukesh Ambani</a>, <a href="/wiki/Reliance_Industries" title
="Reliance Industries">Reliance Industries</a>, <a href="/wiki/Gautam_Adani" ti
tle="Gautam Adani">Gautam Adani</a>, <a href="/wiki/Adani_Group" title="Adani G
roup">Adani Group</a>, <a href="/wiki/Shiv_Nadar" title="Shiv Nadar">Shiv Nadar
</a>, <a href="/wiki/HCL_Technologies" title="HCL Technologies">HCL Technologie
s</a>, <a href="/wiki/Lakshmi_Mittal" title="Lakshmi Mittal">Lakshmi Mittal</a
>, <a href="/wiki/ArcelorMittal" title="ArcelorMittal">ArcelorMittal</a>, <a hr
ef="/wiki/Radhakishan_Damani" title="Radhakishan Damani">Radhakishan Damani</a
>, <a href="/wiki/DMart" title="DMart">Avenue Supermarts</a>, <a href="/wiki/DM
art" title="DMart">DMart</a>, <a href="/wiki/Pallonji_Mistry" title="Pallonji M
istry">Pallonji Mistry</a>, <a href="/wiki/Shapoorji_Pallonji_Group" title="Sha
poorji Pallonji Group">Shapoorji Pallonji Group</a>, <a class="mw-redirect" hre
f="/wiki/Hinduja_brothers" title="Hinduja brothers">Hinduja brothers</a>, <a hr
ef="/wiki/Hinduja_Group" title="Hinduja Group">Hinduja Group</a>, <a href="/wik
i/Uday_Kotak" title="Uday Kotak">Uday Kotak</a>, <a href="/wiki/Kotak_Mahindra_
Bank" title="Kotak Mahindra Bank">Kotak Mahindra Bank</a>, <a href="/wiki/Savit
ri_Jindal" title="Savitri Jindal">Savitri Jindal</a>, <a href="/wiki/JSW_Group"
title="JSW Group">JSW Group</a>, <a href="/wiki/Jindal_Steel_and_Power" title
="Jindal Steel and Power">Jindal Steel &amp;amp; Power</a>, <a class="mw-redirect"
href="/wiki/Cyrus_Poonawalla" title="Cyrus Poonawalla">Cyrus Poonawalla</a>, <a
href="/wiki/Serum_Institute_of_India" title="Serum Institute of India">Serum In
stitute of India</a>, <a href="/wiki/Kumar_Mangalam_Birla" title="Kumar Mangala
m Birla">Kumar Mangalam Birla</a>, <a href="/wiki/Aditya_Birla_Group" title="Ad
itya Birla Group">Aditya Birla Group</a>, <a href="/wiki/Dilip_Shanghvi" title
="Dilip Shanghvi">Dilip Shanghvi</a>, <a href="/wiki/Sun_Pharma" title="Sun Pha
rma">Sun Pharmaceutical Industries</a>, <a href="/wiki/Sunil_Mittal" title="Sun
il Mittal">Sunil Mittal</a>, <a href="/wiki/Bharti_Enterprises" title="Bharti E
nterprises">Bharti Enterprises</a>, <a href="/wiki/Godrej_family" title="Godrej
family">Godrej family</a>, <a href="/wiki/Godrej_Group" title="Godrej Group">Go
drej Group</a>, <a href="/wiki/Anand_Burman" title="Anand Burman">Burman family
</a>, <a href="/wiki/Dabur" title="Dabur">Dabur India</a>, <a href="/wiki/Azim_
Premji" title="Azim Premji">Azim Premji</a>, <a href="/wiki/Wipro" title="Wipr
o">Wipro Group</a>, <a href="/wiki/Kuldip_Singh_Dhingra" title="Kuldip Singh Dh
ingra">Kuldip</a>, <a href="/wiki/Gurbachan_Singh_Dhingra" title="Gurbachan Sin
gh Dhingra">Gurbachan Singh Dhingra</a>, <a href="/wiki/Berger_Paints" title="B
erger Paints">Berger Paints</a>, <a href="/wiki/Benu_Gopal_Bangur" title="Benu
Gopal Bangur">Benu Gopal Bangur</a>, <a href="/wiki/Shree_Cement" title="Shree
Cement">Shree Cement</a>, <a href="/wiki/Divi%27s_Laboratories" title="Divi's L
aboratories">Divi's Laboratories</a>, <a href="/wiki/Ashwin_Dani" title="Ashwin
Dani">Ashwin Dani</a>, <a href="/wiki/Asian_Paints" title="Asian Paints">Asian
Paints</a>, <a href="/wiki/Madhukar_Parekh" title="Madhukar Parekh">Madhukar Pa
rekh</a>, <a href="/wiki/Pidilite_Industries" title="Pidilite Industries">Pidil
ite Industries</a>, <a href="/wiki/Pankaj_Patel" title="Pankaj Patel">Pankaj Pa
tel</a>, <a href="/wiki/Cadila_Healthcare" title="Cadila Healthcare">Cadila Hea
lthcare</a>, <a href="/wiki/Rahul_Bajaj" title="Rahul Bajaj">Rahul Bajaj</a>, <
a href="/wiki/Bajaj_Group" title="Bajaj Group">Bajaj Group</a>, <a href="/wiki/
Sudhir_Mehta" title="Sudhir Mehta">Sudhir</a>, <a href="/wiki/Samir_Mehta" titl
e="Samir Mehta">Samir Mehta</a>, <a href="/wiki/Torrent_Group" title="Torrent G
roup">Torrent Group</a>, <a class="mw-redirect" href="/wiki/Intas_Biopharmaceut
icals" title="Intas Biopharmaceuticals">Intas Biopharmaceuticals</a>]
```

In [16]:
```
net_worth =[]
for links in table_links:
    net_worth.append(links.get('title'))
print(net_worth)
```

```
['Net worth', 'USD', 'Mukesh Ambani', 'Reliance Industries', 'Gautam Adani', 'A
dani Group', 'Shiv Nadar', 'HCL Technologies', 'Lakshmi Mittal', 'ArcelorMitta
l', 'Radhakishan Damani', 'DMart', 'DMart', 'Pallonji Mistry', 'Shapoorji Pallo
nji Group', 'Hinduja brothers', 'Hinduja Group', 'Uday Kotak', 'Kotak Mahindra
Bank', 'Savitri Jindal', 'JSW Group', 'Jindal Steel and Power', 'Cyrus Poonawal
la', 'Serum Institute of India', 'Kumar Mangalam Birla', 'Aditya Birla Group',
'Dilip Shanghvi', 'Sun Pharma', 'Sunil Mittal', 'Bharti Enterprises', 'Godrej f
amily', 'Godrej Group', 'Anand Burman', 'Dabur', 'Azim Premji', 'Wipro', 'Kuldi
p Singh Dhingra', 'Gurbachan Singh Dhingra', 'Berger Paints', 'Benu Gopal Bangu
r', 'Shree Cement', "Divi's Laboratories", 'Ashwin Dani', 'Asian Paints', 'Madh
ukar Parekh', 'Pidilite Industries', 'Pankaj Patel', 'Cadila Healthcare', 'Rahu
l Bajaj', 'Bajaj Group', 'Sudhir Mehta', 'Samir Mehta', 'Torrent Group', 'Intas
Biopharmaceuticals']
```

In [17]:
```
import pandas as pd
df=pd.DataFrame(net_worth)
print(df)
```

```
                          0
0                 Net worth
1                       USD
2              Mukesh Ambani
3        Reliance Industries
4               Gautam Adani
5                Adani Group
6                 Shiv Nadar
7           HCL Technologies
8             Lakshmi Mittal
9              ArcelorMittal
10        Radhakishan Damani
11                    DMart
12                    DMart
13           Pallonji Mistry
14  Shapoorji Pallonji Group
15          Hinduja brothers
16             Hinduja Group
17                Uday Kotak
18       Kotak Mahindra Bank
19             Savitri Jindal
20                  JSW Group
21    Jindal Steel and Power
22          Cyrus Poonawalla
23  Serum Institute of India
24      Kumar Mangalam Birla
25        Aditya Birla Group
26             Dilip Shanghvi
27                 Sun Pharma
28               Sunil Mittal
29        Bharti Enterprises
30              Godrej family
31               Godrej Group
32               Anand Burman
33                      Dabur
34                Azim Premji
35                      Wipro
36      Kuldip Singh Dhingra
37   Gurbachan Singh Dhingra
38              Berger Paints
39         Benu Gopal Bangur
40               Shree Cement
41       Divi's Laboratories
42                Ashwin Dani
43               Asian Paints
44           Madhukar Parekh
45       Pidilite Industries
46                Pankaj Patel
47          Cadila Healthcare
48                Rahul Bajaj
49                Bajaj Group
50               Sudhir Mehta
51                Samir Mehta
```

```
52             Torrent Group
53  Intas Biopharmaceuticals
```

In [18]:
```
writer=pd.ExcelWriter('networth.xlsx', engine='xlsxwriter')
df.to_excel(writer,sheet_name='List')
writer.save()
```

In [19]:
```
df1=pd.read_excel('networth.xlsx')
df1
```

Out[19]:

	Unnamed: 0	0
0	0	Net worth
1	1	USD
2	2	Mukesh Ambani
3	3	Reliance Industries
4	4	Gautam Adani
5	5	Adani Group
6	6	Shiv Nadar
7	7	HCL Technologies
8	8	Lakshmi Mittal
9	9	ArcelorMittal
10	10	Radhakishan Damani
11	11	DMart
12	12	DMart
13	13	Pallonji Mistry
14	14	Shapoorji Pallonji Group
15	15	Hinduja brothers
16	16	Hinduja Group
17	17	Uday Kotak
18	18	Kotak Mahindra Bank
19	19	Savitri Jindal
20	20	JSW Group
21	21	Jindal Steel and Power
22	22	Cyrus Poonawalla
23	23	Serum Institute of India
24	24	Kumar Mangalam Birla
25	25	Aditya Birla Group
26	26	Dilip Shanghvi
27	27	Sun Pharma
28	28	Sunil Mittal
29	29	Bharti Enterprises
30	30	Godrej family
31	31	Godrej Group
32	32	Anand Burman
33	33	Dabur

	Unnamed: 0	0
34	34	Azim Premji
35	35	Wipro
36	36	Kuldip Singh Dhingra
37	37	Gurbachan Singh Dhingra
38	38	Berger Paints
39	39	Benu Gopal Bangur
40	40	Shree Cement
41	41	Divi's Laboratories
42	42	Ashwin Dani
43	43	Asian Paints
44	44	Madhukar Parekh
45	45	Pidilite Industries
46	46	Pankaj Patel
47	47	Cadila Healthcare
48	48	Rahul Bajaj
49	49	Bajaj Group
50	50	Sudhir Mehta
51	51	Samir Mehta
52	52	Torrent Group
53	53	Intas Biopharmaceuticals

we have sucessfully created our first web scraping program.

6.Create a 3*3 matrix with values ranging from 2 to 10 using numpy

Solution:

In [20]:
```
import numpy as np
x = np.arange(2,11).reshape(3,3)
print(x)
```

```
[[ 2  3  4]
 [ 5  6  7]
 [ 8  9 10]]
```

7.Write a python program to convert a list of numeric value into a one dimensional Numpy array

Solution:

In [21]:
```
import numpy as np
l=[12.23,13.32,100.67,36.32]
print("Original List:",l)
a=np.array(l)
print("One-Dimensional Numpy array:",a)
```

```
Original List: [12.23, 13.32, 100.67, 36.32]
One-Dimensional Numpy array: [ 12.23  13.32 100.67  36.32]
```

8.Write a python program to create a null vector of size 10 and update sixth value to 11

Solution:

In [22]:
```
import numpy as np
x= np.zeros(10)
print(x)
print("Update Sixth value to 11")
x[6] = 11
print(x)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Update Sixth value to 11
[ 0.  0.  0.  0.  0.  0. 11.  0.  0.  0.]
```

==========

In [ ]:
```

```