Reference:

https://www.tutorialspoint.com/python_data_structure/python_matrix.htm (https://www.tutorialspoint.com/python_data_structure/python_matrix.htm)

for pandas

https://www.tutorialspoint.com/python_pandas/python_pandas_missing_data.htm (https://www.tutorialspoint.com/python_pandas/python_pandas_missing_data.htm)

https://stackabuse.com/beginners-tutorial-on-the-pandas-python-library/ (https://stackabuse.com/beginners-tutorial-on-the-pandas-python-library/)

https://www.journaldev.com/29055/python-pandas-module-tutorial (https://www.journaldev.com/29055/python-pandas-module-tutorial)

https://www.learnpython.org/en/Pandas_Basics (https://www.learnpython.org/en/Pandas_Basics)

In [ ]:

In [ ]:

Sequence

print a sequence from 0 to 100?

In [1]:
```
lst=list(range(101))
for i in range(101):
     print (lst[i])
```

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57

```
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

In [2]:
```
 #Suppose we want to better formet
lst=list(range(100))
for i in range(100):
    print (lst[i],',',end='')
```

```
0 ,1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,10 ,11 ,12 ,13 ,14 ,15 ,16 ,17 ,18 ,19 ,20 ,21
,22 ,23 ,24 ,25 ,26 ,27 ,28 ,29 ,30 ,31 ,32 ,33 ,34 ,35 ,36 ,37 ,38 ,39 ,40
,41 ,42 ,43 ,44 ,45 ,46 ,47 ,48 ,49 ,50 ,51 ,52 ,53 ,54 ,55 ,56 ,57 ,58 ,59
,60 ,61 ,62 ,63 ,64 ,65 ,66 ,67 ,68 ,69 ,70 ,71 ,72 ,73 ,74 ,75 ,76 ,77 ,78
,79 ,80 ,81 ,82 ,83 ,84 ,85 ,86 ,87 ,88 ,89 ,90 ,91 ,92 ,93 ,94 ,95 ,96 ,97
,98 ,99 ,
```

In [ ]:
```
lst=list(range(101))
print(lst)
```

In [ ]:
```
#suppose we want to print sequence from 10 t0 20
ls=list(range(10,20+1))
print(ls)
```

Importing Modules

A module is a collection of function ,classes and data structures,statements etc. written as a python code.We can think a module as a library from where we can import various built-in function.Example:in a math module, there is a collection of large number of built in mathematical function-trigornometric,exponential,logarithamic etc. and also some mathematical constant example: pi etc.

Import math library

In [ ]:
```
import math
math.sqrt(100)
```

In [ ]:
```
math.pi
```

Other ways to import math module by a nick name

In [ ]:
```
import math as suman
suman.pi
```

Import specfic function and constants from math module

In [ ]:
```
from math import sin,pi
sin(pi/4)
```

Import all function from math module

In [ ]:
```
from math import *
sin(pi/4)
```

In [ ]:
```
e
```

In [ ]:
```
pow(3,4)
```

import "cmath" module for complex number operation

In [ ]:
```
from cmath import *  # import all function from cmath module
sqrt(-1)
```

For help

In [ ]:
```
help(math)
```

Time

python time is measured units of seconds(floating point numbers) and the clock ticks the number of seconds since 12.00 A.M ,January 1,1990.To know time,we have to import the time module.

In [ ]:
```
import time
time.time()
```

In [ ]:
```
time.time()
```

We can notice that the time() function registers two different time units in seconds,i.e, time is ever changing

For calender

In [ ]:
```
import calendar
calendar.month(2020,12)
```

In [ ]:
```
_  #The '_' variable can be used to refer the last thing that was evaluated
```

if else statements

In [3]:
```
#take a variable x ,cheek it even or not
x=6
if(x%2==0):
    print("even")
else:
    print("odd")
```

```
even
```

In [ ]:

In [4]:
```
#here, we will input value of x
x=int(input("please enter an integer number?\n"))
if(x%2==0):
    print(x,"is even")
else:
    print(x,"is odd")
```

```
please enter an integer number?
6
6 is even
```

In [5]:
```
#TAKE A VARIABLE Y,PRINT-GRADE A,IF Y>90,GRADE B,IF 60<Y<90,GRADE F OTHERWISE
y=float(input("enter the marks?"))
if(y>90):
    print("grade A")
elif(y>60):
    print("grade B")
else:
    print("grade F")
```

```
enter the marks?67
grade B
```

In [6]:
```
y=int(input("enter the marks( integer)?\n"))
if(y>90):
    print("grade A")
elif(y>60 & y<90):
    print("grade B")
else:
    print("grade F")
```

```
enter the marks( integer)?
56
grade F
```

In [7]:
```
#we can not use if posotion in sceond posotion such as "C","R". if we use "if
" condition then we get some wrong output when y>90.
y=int(input("enter the marks( integer)?\n"))            #####wrong output ##
##
if(y>90):
    print("grade A")
if(y>60 & y<90):
    print("grade B")
else:
    print("grade F")
```

```
enter the marks( integer)?
78
grade B
```

In [8]:
```
## cheek some condition

if(5>10):
    print("A")
elif(8!=9):
    print("B")
else:
    print("C")
```

```
B
```

break statement

In [9]:
```
a=[10,12,8,7,16,4]
for i in a:
    if(i%2==1):
        break
    print(i) # here, we stop the loop if we get odd number
```

```
10
12
8
```

In [10]:
```
a=[10,12,8,7,16,4]
for i in a:
    if(i%2==1):
        break
print(i) # here, we stop the loop if we get odd number and we will print this
odd number
```

```
7
```

loop in python

In [ ]:
```

```

In [11]:
```
#in python dex start at i=0 continue (n-1), here n=5, loop start at n=0, and
 continue upto 6-1=5
for i in range(6):
    print(i)
```

```
0
1
2
3
4
5
```

In [12]:
```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

```
apple
banana
cherry
```

In [13]:
```
#Loop through the letters in the word "banana":
for i in "bananna":
   print(i)
```

```
b
a
n
a
n
n
a
```

In [14]:
```
#print all the numbers from 11 to 50.
for i in range(11,51):
    print(i)
```

```
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
```

In [15]:
```
#print all the numbers from 11 to 50 with interval 3?
for i in range(11,51,3):
    print(i)
```

```
11
14
17
20
23
26
29
32
35
38
41
44
47
50
```

In [16]:
```
l=list(range(1,25+1)) #create a list of squre of  numbers 1 to 25.
n=len(l)
s1=0
s2=0
for i in range(n):
    y=i**2
    print(y)
```

```
0
1
4
9
16
25
36
49
64
81
100
121
144
169
196
225
256
289
324
361
400
441
484
529
576
```

In [17]:
```
l=[5,8,9,1] #create a list thats elements are squre of a given list l.
n=len(l)
y=[]
for i in l:
    y=i**2
    print(y)
```

```
25
64
81
1
```

In [18]:
```
#alternative way create a list thats elements are squre of a given list l.
l=[5,8,9,1]
m=[i**2 for i in l]
print(m)
```

```
[25, 64, 81, 1]
```

In [19]:
```
p=[5,8,9]   #alternative way
x=[0]*len(p) #define a  null list x
n=len(p)
for i in range(n):
    x[i]=p[i]**2

print(x)
```

```
[25, 64, 81]
```

In [ ]:

creating function in python:

Write a function in python for calculating area of a circle?

In [20]:
```
def area(r):
    a=3.14*r*r
    return(a)
area(2)  # calling function for calculating area of redius 2.
```

Out[20]:
```
12.56
```

In [21]:
```
#create a function for calculating which number is large between a &b?
def g(a,b):
    if(a>b):
        grater=a
    else:
        grater=b
    return grater
```

In [22]:
```
g(9,7)
```

Out[22]: 9

In [23]:
```
g(6,23)
```

Out[23]: 23

list

In [24]:
```
# integer list
l1=[1,5,8,9]
l1
```

Out[24]: [1, 5, 8, 9]

Create a list whose elements are 1,3,...,25?

In [25]:
```
lis=list(range(1,25,2))
print(lis)
```

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23]
```

In [26]:
```
l=list(range(1,25,2))
print(l)
```

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23]
```

In [27]:
```
l2=["suman","ram","A","B"]
l2
```

Out[27]: ['suman', 'ram', 'A', 'B']

In [28]:
```
l3=["A",2,"B",3,"C"]
l3
```

Out[28]: ['A', 2, 'B', 3, 'C']

In [29]:
```
l3[0:2]
```

Out[29]:
```
['A', 2]
```

In [30]:
```
l3[1:2]
```

Out[30]:
```
[2]
```

In [31]:
```
l3[0]
```

Out[31]:
```
'A'
```

In [32]:
```
l3[-1] #print the last element by negative index
```

Out[32]:
```
'C'
```

In [33]:
```
l3[4] #print the last element
```

Out[33]:
```
'C'
```

In [34]:
```
l = range(10)   #create a list of elements 0 to 9
for i in l:
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

Adding element in list

In [35]:
```
l4=["A",3,"B",6,"p"]
l4.append(7)  #adding an element 7
l4
```

Out[35]:
```
['A', 3, 'B', 6, 'p', 7]
```

In [36]:
```
l5=["A",3,"C"]
l5.extend([5,"A"])
l5
```

Out[36]:
```
['A', 3, 'C', 5, 'A']
```

Adding list to a list

In [37]:
```
l6=[1,7,4,"A","r"]
l4.append([5,9])
l4
```

Out[37]:
```
['A', 3, 'B', 6, 'p', 7, [5, 9]]
```

In [38]:
```
l4[-1] # print an element
```

Out[38]:
```
[5, 9]
```

Deleting an element of a list

In [39]:
```
l3
```

Out[39]:
```
['A', 2, 'B', 3, 'C']
```

In [40]:
```
l3.remove('B') # removing  an element of a list
```

In [41]:
```
l3
```

Out[41]:
```
['A', 2, 3, 'C']
```

In [42]:
```
l4
```

Out[42]:
```
['A', 3, 'B', 6, 'p', 7, [5, 9]]
```

In [43]:
```
del l4[3]  # removing the element whose index is 0
```

In [44]:
```
l4
```

Out[44]:
```
['A', 3, 'B', 'p', 7, [5, 9]]
```

printing elements in a list

In [45]:
```
l4
```

Out[45]:
```
['A', 3, 'B', 'p', 7, [5, 9]]
```

In [46]:
```
l4=[9,9.9,2,7]
for i in l4:
    print(i)
```

```
9
9.9
2
7
```

Create A list y thats elements are same of x?

In [42]:
```
x = [-1.1, 0.0, 3.6, -7.2]
y = x.copy()
print(y)
```

```
[-1.1, 0.0, 3.6, -7.2]
```

Cheek x and y these two list are equal or not?

In [43]:
```
x = [-1.1, 0.0, 3.6, -7.2]
y = x.copy()
x==y
```

Out[43]:
```
True
```

As the output is true so, the list x and y are equal.

Scalars versus 1-vectors.

In the mathematical notations of VMLS, 1-vector is considered as a scalar. However, in Python, 1-vectors are not the same as scalars. For list structure, Python distinguishes 1-vector (list with only one element) [2.4] and the number 2.4.

In [45]:
```
x = 2.4
y = [2.4]
x == y
```

Out[45]:
```
False
```

Block and stacked vectors

In Python, we can construct a block vector using the numpy function concatenate(). Remember you need an extra set of parentheses over the vectors that you want to concatenate. The use of numpy array or list structure does not create a huge difference here.

In [46]:
```
import numpy as np
x = np.array([1, -2])
y = np.array([1,1,0])
z = np.concatenate((x,y))
print(z)
```

```
[ 1 -2  1  1  0]
```

Some common mistakes

There are few Python operations that appear to be able to construct a block or stacked vector but do not. For example, z = (x,y) creates a tuple of two vectors; z = [x,y] creates an array of the two vectors. Both of these are valid Python expression but neither of them is the stacked vector.

In [ ]:

How to define a list with all elements are 0 for any fixed size n?

In [47]:
```
n=5
l=[0]*5
print(l)
```

```
[0, 0, 0, 0, 0]
```

Write a function for creating a list of order n with all elements are 0?

In [15]:
```
def null_list(n):
    L=[0]*n
    return(L)
null_list(6)  #call the function for creating a function of order n with all
 elements are 0.
```

Out[15]:
```
[0, 0, 0, 0, 0, 0]
```

Nested List(list inside a list)

A nested list is simply a list that occurs as an element of another list (which may of course itself be an element of another list, etc.). Common reasons nested lists arise are: They're matrices (a list of rows, where each row is itself a list, or a list of columns where each column is itself a list).

In [ ]:

In [6]:
```
L=[[1,2,9],[4,7,8]]
L[1] # call the sceond list
```

Out[6]:
```
[4, 7, 8]
```

In [8]:
```
L[1][2] #call the third element of sceond list
```

Out[8]:
```
8
```

Crete a nested list has six number of element and each of the list inside the list has four numbers of elements these are 1,2,3,4?

In [17]:
```
# Nested list comprehension
L = [[j for j in range(1,5)] for i in range(6)]
print(L)
```

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

It can be treated as a matrix with order 6×4.

Write a matrix of 4×3 order ?

Alternative question: Create a nested list of four elements such that every list inside the list have 3 numbers of elments all arev equal to 0?

In [111]:
```
list=[[0]*3]*4
print(list)
```

```
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

In [112]:
```
list[2]
```

Out[112]:
```
[0, 0, 0]
```

Write a matrix of $m \times n$ order ?

Alternative question: Create a nested list of m elements such that every list inside the list have n numbers of elments all arev equal to 0?

In [118]:
```
def null_matrix(m,n):
    mat=[[0]*n]*m
    return(mat)
null_matrix(3,4)
```

Out[118]:
```
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
```

In [119]:
```
z=null_matrix(3,4)
z[2][3]
```

Out[119]:
```
0
```

Dictionary

In [48]:
```
marks={"geography":100,"English":56}  #creating a dictionary
marks
```

Out[48]:
```
{'geography': 100, 'English': 56}
```

In [49]:
```
#access value
marks["geography"]
```

Out[49]:
```
100
```

In [50]:
```
#adding a dictionary
marks["math"]=80
marks
```

Out[50]:
```
{'geography': 100, 'English': 56, 'math': 80}
```

In [51]:
```
#adding multiple dictionary
marks.update({"hist":34,"bios":56})
marks
```

Out[51]:
```
{'geography': 100, 'English': 56, 'math': 80, 'hist': 34, 'bios': 56}
```

In [52]:
```
marks.update({"bengali":78})
marks
```

Out[52]:
```
{'geography': 100,
 'English': 56,
 'math': 80,
 'hist': 34,
 'bios': 56,
 'bengali': 78}
```

In [53]:
```
#delecting of a dictionary
del marks["bengali"]
marks
```

Out[53]:
```
{'geography': 100, 'English': 56, 'math': 80, 'hist': 34, 'bios': 56}
```

Write a function for creatingn a list of n order with all elements are 0?

In [12]:
```
m=3
n=2
x=[[0]*n]*m
x
```

Out[12]:
```
[[0, 0], [0, 0], [0, 0]]
```

Reading csv and excell file

In [54]:
```
import pandas as pd #import panndas data set
pd
```

Out[54]:
```
<module 'pandas' from 'C:\\Users\\SUMAN GHOSH\\Anaconda3\\lib\\site-packages
\\pandas\\__init__.py'>
```

In [55]:
```
#READ EXCELL FILE IN PYTHON
#name=dataset.read_excell(r'location\filename.elsx)
#for detalis please go to the link:    https://datatofish.com/read_excel/
df = pd.read_excel (r'C:\Users\SUMAN GHOSH\Desktop\datapython.xlsx')
print(df) #to see the file

#for csv file we can ony chane csv in stead of excell-df1= pd.read_csv
 (r'C:\Users\SUMAN GHOSH\Desktop\datapython.csv')
# where, datapython is our csv file
```

```
            Product   Price  number of items  warienty  defect probabilty  \
0  Desktop Computer   700.0                1         4                0.1
1            Tablet   250.0                3         3                0.4
2            iPhone   800.0                1         6                0.7
3            Laptop  1200.0                1         1                0.5
4               NaN     NaN                1         8                0.0
5         headphone   150.0                1         9                0.1
6             mouse   300.0                2         4                0.0
7           keybord   600.0                1        21                0.0

   available shop
0              34
1               5
2              67
3              22
4            3564
5             123
6             875
7            1234
```

In [56]:
```
df.head()  #to print first few rows
```

Out[56]:

	Product	Price	number of items	warienty	defect probabilty	available shop
0	Desktop Computer	700.0	1	4	0.1	34
1	Tablet	250.0	3	3	0.4	5
2	iPhone	800.0	1	6	0.7	67
3	Laptop	1200.0	1	1	0.5	22
4	NaN	NaN	1	8	0.0	3564

In [ ]:
```
import pandas as pd
df1= pd.read_csv (r'C:\Users\SUMAN GHOSH\Desktop\datapython.csv')
df1
```

Dataframe and basic operation in pandas library

In [57]:
```
import pandas as pd #inport pandas library as pd
# inpython we import data(example:excell,csv.e.t.c) as 'Data frame'
df = pd.read_excel (r'C:\Users\SUMAN GHOSH\Desktop\datapython.xlsx')
print(df) #to print the df data frame
```

```
            Product   Price  number of items  warienty  defect probabilty  \
0  Desktop Computer   700.0                1         4                0.1
1            Tablet   250.0                3         3                0.4
2            iPhone   800.0                1         6                0.7
3            Laptop  1200.0                1         1                0.5
4               NaN     NaN                1         8                0.0
5         headphone   150.0                1         9                0.1
6             mouse   300.0                2         4                0.0
7           keybord   600.0                1        21                0.0

   available shop
0              34
1               5
2              67
3              22
4            3564
5             123
6             875
7            1234
```

In [58]:
```
df.shape  #to see the numbers to rows and column in df data frame
```

Out[58]: (8, 6)

In [59]:
```
df.head() # to print the top 5 rows
```

Out[59]:

	Product	Price	number of items	warienty	defect probabilty	available shop
0	Desktop Computer	700.0	1	4	0.1	34
1	Tablet	250.0	3	3	0.4	5
2	iPhone	800.0	1	6	0.7	67
3	Laptop	1200.0	1	1	0.5	22
4	NaN	NaN	1	8	0.0	3564

In [60]:
```
df.tail() #to print last 5 rows
```

Out[60]:

	Product	Price	number of items	warienty	defect probabilty	available shop
3	Laptop	1200.0	1	1	0.5	22
4	NaN	NaN	1	8	0.0	3564
5	headphone	150.0	1	9	0.1	123
6	mouse	300.0	2	4	0.0	875
7	keybord	600.0	1	21	0.0	1234

In [61]:
```
df.tail(3) #to print last 3 rows
```

Out[61]:

	Product	Price	number of items	warienty	defect probabilty	available shop
5	headphone	150.0	1	9	0.1	123
6	mouse	300.0	2	4	0.0	875
7	keybord	600.0	1	21	0.0	1234

In [62]:
```
df.head(2) #to print first 2 rows
```

Out[62]:

	Product	Price	number of items	warienty	defect probabilty	available shop
0	Desktop Computer	700.0	1	4	0.1	34
1	Tablet	250.0	3	3	0.4	5

In [63]:
```
df.columns #to see all the column names
```

Out[63]:
```
Index(['Product', 'Price', 'number of items', 'warienty', 'defect probabilt
y',
       'available shop'],
      dtype='object')
```

In [64]:
```
 df['Price']  #to see all the value of a single column name'Price'
```

Out[64]:
```
0     700.0
1     250.0
2     800.0
3    1200.0
4       NaN
5     150.0
6     300.0
7     600.0
Name: Price, dtype: float64
```

In [65]:
```
df[['Price','Product']] # to see the multiple columns values together
```

Out[65]:

	Price	Product
0	700.0	Desktop Computer
1	250.0	Tablet
2	800.0	iPhone
3	1200.0	Laptop
4	NaN	NaN
5	150.0	headphone
6	300.0	mouse
7	600.0	keybord

Indexing,selecting e.t.c

In [66]:
```
import pandas as pd #import pandas library
df = pd.read_excel (r'C:\Users\SUMAN GHOSH\Desktop\datapython.xlsx')
print(df)
```

```
             Product   Price  number of items  warienty  defect probabilty  \
0  Desktop Computer   700.0                1         4                0.1
1            Tablet   250.0                3         3                0.4
2            iPhone   800.0                1         6                0.7
3            Laptop  1200.0                1         1                0.5
4               NaN     NaN                1         8                0.0
5         headphone   150.0                1         9                0.1
6             mouse   300.0                2         4                0.0
7           keybord   600.0                1        21                0.0

   available shop
0              34
1               5
2              67
3              22
4            3564
5             123
6             875
7            1234
```

In [67]:
```
#selecting rows by their posotion
df.iloc[:5] #selecting first 5 rows of their posotions-it wiil start from 0 a
nd continue upto 5-1=4
```

Out[67]:

	Product	Price	number of items	warienty	defect probabilty	available shop
0	Desktop Computer	700.0	1	4	0.1	34
1	Tablet	250.0	3	3	0.4	5
2	iPhone	800.0	1	6	0.7	67
3	Laptop	1200.0	1	1	0.5	22
4	NaN	NaN	1	8	0.0	3564

In [68]:
```
#selecting column by their posotion
df.iloc[:,:3]  #selecting 3 columns by their posotion .it will start and cont
inue upto 3-1=2
```

Out[68]:

	Product	Price	number of items
0	Desktop Computer	700.0	1
1	Tablet	250.0	3
2	iPhone	800.0	1
3	Laptop	1200.0	1
4	NaN	NaN	1
5	headphone	150.0	1
6	mouse	300.0	2
7	keybord	600.0	1

In [69]:
```
#selecting rows by specific conditions
df[df['number of items']==1]    #prints all values whose number of items are
equal to 1.
```

Out[69]:

	Product	Price	number of items	warienty	defect probabilty	available shop
0	Desktop Computer	700.0	1	4	0.1	34
2	iPhone	800.0	1	6	0.7	67
3	Laptop	1200.0	1	1	0.5	22
4	NaN	NaN	1	8	0.0	3564
5	headphone	150.0	1	9	0.1	123
7	keybord	600.0	1	21	0.0	1234

In [70]:
```
import pandas as pd
index=pd.read_csv(r'F:\Python\index.csv')  # to read csv file'index' in pyth
on. it store in python as a dataframe.
print(index) #to print the dataframe index
```

```
     A  B   C   D  E  F      G      H       I     J  K  L
0    C  A   8   0  S  0   COIL  0.700   610.0     0  0  3
1    C  R   0   0  S  0   COIL  3.200   610.0     0  0  3
2    C  R   0   0  S  0  SHEET  0.700  1300.0   762  0  3
3    C  A   0  60  ?  0   COIL  2.801   385.1     0  0  3
4    C  A   0  60  ?  0  SHEET  0.801   255.0   269  0  3
..  .. ..  ..  .. .. ..    ...    ...     ...   ... .. ..
252  C  A   0  60  ?  0   COIL  0.799   609.0     0  0  3
253  C  R   6   0  ?  0  SHEET  1.000   610.0  4880  0  3
254  C  A   0  50  ?  0   COIL  0.600    20.0     0  0  3
255  C  K  45   0  ?  0   COIL  0.600   900.0     0  0  3
256  C  R   6   0  ?  0  SHEET  1.000   610.0   762  0  3

[257 rows x 12 columns]
```

In [71]:
```
#Access the third column of the index dataframe?
third_column=index. columns[2]  # to know the third column name of the index
data set.
third_column
index[third_column]  #print the third column
```

Out[71]:
```
0       8
1       0
2       0
3       0
4       0
       ..
252     0
253     6
254     0
255    45
256     6
Name: C, Length: 257, dtype: int64
```

In [72]:
```
#Access the last two column of the index dataset.
index.iloc[:,[-1,-2]]  # last columns means=-1.
```

Out[72]:

	L	K
0	3	0
1	3	0
2	3	0
3	3	0
4	3	0
...	...	...
252	3	0
253	3	0
254	3	0
255	3	0
256	3	0

257 rows × 2 columns

In [73]:
```
#Access the last 10 rows
index.tail(10)
```

Out[73]:

	A	B	C	D	E	F	G	H	I	J	K	L
247	C	R	0	0	S	0	COIL	0.901	966.0	0	0	3
248	C	A	0	0	S	0	SHEET	0.700	610.0	762	0	3
249	C	A	0	0	S	0	COIL	0.800	75.0	0	0	3
250	C	R	0	0	S	0	SHEET	0.700	1320.0	762	0	3
251	C	A	0	50	?	0	SHEET	0.601	1250.0	762	0	3
252	C	A	0	60	?	0	COIL	0.799	609.0	0	0	3
253	C	R	6	0	?	0	SHEET	1.000	610.0	4880	0	3
254	C	A	0	50	?	0	COIL	0.600	20.0	0	0	3
255	C	K	45	0	?	0	COIL	0.600	900.0	0	0	3
256	C	R	6	0	?	0	SHEET	1.000	610.0	762	0	3

In [74]:
```
#print first two columns of the index datasets
index.iloc[:,[0,1]]
```

Out[74]:

	A	B
0	C	A
1	C	R
2	C	R
3	C	A
4	C	A
...	...	...
252	C	A
253	C	R
254	C	A
255	C	K
256	C	R

257 rows × 2 columns

Sorting dataset

In [75]:
```
import pandas as pd #import panndas data set
pd
```

Out[75]:
```
<module 'pandas' from 'C:\\Users\\SUMAN GHOSH\\Anaconda3\\lib\\site-packages
\\pandas\\__init__.py'>
```

In [76]:
```
import numpy as np
np
df = pd.read_excel (r'C:\Users\SUMAN GHOSH\Desktop\datapython.xlsx')
df
```

Out[76]:

	Product	Price	number of items	warienty	defect probabilty	available shop
0	Desktop Computer	700.0	1	4	0.1	34
1	Tablet	250.0	3	3	0.4	5
2	iPhone	800.0	1	6	0.7	67
3	Laptop	1200.0	1	1	0.5	22
4	NaN	NaN	1	8	0.0	3564
5	headphone	150.0	1	9	0.1	123
6	mouse	300.0	2	4	0.0	875
7	keybord	600.0	1	21	0.0	1234

In [77]:
```
pdata=pd.read_csv(r'C:\Users\SUMAN GHOSH\Desktop\python_data\bigmart_data.csv') #read data frm csv file to python
pdata
```

Out[77]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Out
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092	
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692	
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180	
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614	
...	...	...	...	...	...	...	...
8518	FDF22	6.865	Low Fat	0.056783	Snack Foods	214.5218	
8519	FDS36	8.380	Regular	0.046982	Baking Goods	108.1570	
8520	NCJ29	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	
8521	FDN46	7.210	Regular	0.145221	Snack Foods	103.1332	
8522	DRG01	14.800	Low Fat	0.044878	Soft Drinks	75.4670	

8523 rows × 12 columns

In [ ]:

pdata=pdata.dropna(how='any') #he dropna() function is used to remove missing values. Determine if rows or columns which contain missing values are removed. 0,

pdata # link:https://www.kaggle.com/aliendev/example-of-pandas-dropna (https://www.kaggle.com/aliendev/example-of-pandas-dropna)

In [78]:
```
import pandas as pd #import pandas library for dataframe data processing, CSV file I/O (e.g. pd.read_csv)
import numpy as np #import numpy library for linear algebra
np
dataframe = pd.DataFrame([[np.nan, 2,np.nan, 0], [3, 4, np.nan, 1], [np.nan, np.nan, np.nan, 5],[3, 4, np.nan, 1], [3, 4, 0, 1]], columns=list('ABCD'))
dataframe
```

Out[78]:

	A	B	C	D
0	NaN	2.0	NaN	0
1	3.0	4.0	NaN	1
2	NaN	NaN	NaN	5
3	3.0	4.0	NaN	1
4	3.0	4.0	0.0	1

In [79]:
```
dataframe.drop_duplicates() #It will remove index 3 since it is dublicate to 1
```

Out[79]:

	A	B	C	D
0	NaN	2.0	NaN	0
1	3.0	4.0	NaN	1
2	NaN	NaN	NaN	5
4	3.0	4.0	0.0	1

sortnig a dataframe in pundas

In [80]:
```
import pandas as pd
```

Out[80]:
```
<module 'pandas' from 'C:\\Users\\SUMAN GHOSH\\Anaconda3\\lib\\site-packages\\pandas\\__init__.py'>
```

In [81]:
```
unsorted_df = pd.DataFrame({'col1':[2,1,1,1],'col2':[1,3,2,4]})
unsorted_df
```

Out[81]:

	col1	col2
0	2	1
1	1	3
2	1	2
3	1	4

In [82]:
```
sorted_df = unsorted_df.sort_values(by='col1',kind='heapsort')
sorted_df
```

Out[82]:

	col1	col2
1	1	3
2	1	2
3	1	4
0	2	1

In [83]:
```
for i in range(-6,10):
    print(i)
```

```
-6
-5
-4
-3
-2
-1
0
1
2
3
4
5
6
7
8
9
```

In [84]:
```
for i in range(1,10):
   print(i)
```

```
1
2
3
4
5
6
7
8
9
```

In [85]:
```
a=[4,5]
a
```

Out[85]:
```
[4, 5]
```

In [ ]:
```

```

In [86]:
```
b=[7,8]
b
```

Out[86]:
```
[7, 8]
```

In [87]:
```
n=a.append(b)
n
```

In [88]:
```
s={1,2,3}
s
```

Out[88]:
```
{1, 2, 3}
```

In [89]:
```
s.add([4,1])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-89-5c50ae90f142> in <module>
----> 1 s.add([4,1])

TypeError: unhashable type: 'list'
```

In [ ]:
```
set1={1,TRUE}
set1
```

In [ ]:
```
def fun(x):
    x[0] = 5
    return x
g = [10,11,12]
print fun(g) ,g
```

In [ ]:
```
L1 = [10, 20, 30, 40]

L2 = L1

L3 = L1.copy()

L4 = list(L1)

L1[0] = [50]

print(L1, L2, L3, L4)
```

In [ ]:
```
print(bool(0), bool(1.11), bool(-5))
```

In [ ]:
```
for i in range(1,10):
    if i==5:
        pass
    else:
        print(i,end=" ")
```

In [ ]:
```
list1=[11,12,13]
list1.append([14,15])
print(list1)
```

In [ ]:
```
import pandas as pd
s = pd.Series(data= [10,5,6,2,3,1,4,8,9,7], index=[49,48,47,46,45, 1, 2, 3, 4, 5])
s
```

In [ ]:
```
s.iloc[:3]
```

In [ ]:
```
v={1,2,3}
v.add([4,1])
print(v)
```

In [ ]:
```
a=[3,7,9]
sum(a)
```

In [ ]:
```
a=[4,2,7,5,1,1]
sum(a)
```

for loop

find the sum of the numbers 0 to 100?

In [ ]:
```
sum=0
for i in range(0,101):
  sum=sum+i
print('sum is',sum)
```

In [ ]:
```
#sumof a list
l =[1,2,3,4,5]
sum=0
for element in l:
    sum=sum+element
print(sum)
```

While loop

print the number 0 to 11?

In [ ]:
```
n=1
while n<11:
    print(n)
    n=n+1
```

sum of the number from 0 to 11 using while loop

In [ ]:
```
s=0
n=1
while n<11:
    s=s+n
    n=n+1
print(s)
```

In [ ]:
```
# using inbuilt function sum of these number
```

In [ ]:
```
# creating a list
list1 = [11, 5, 17, 18, 23]            #sum() is not present in this verson

# using sum() function
total =sum(list1)

# printing total value
print("Sum of all elements in given list: ", total)
```

In [ ]:
```
from platform import python_version

print(python_version())
```

In [ ]:
```
import platform
print(platform.python_version())
```

In [ ]:
```
x=[4,8,7]
sum(x)
```

Python Scripts for elementry compution

Area of a circle

In [ ]:
```
from math import pi
r=int(input("enter radius?\n"))
area=pi*r**2
print(area)
```

Creating a function for round off a number?

In [1]:
```
def round_to_places(num,n):
    a=round(num,n)
    return(a)
round_to_places(3.14159,2) # call the function
```

Out[1]: 3.14

In [ ]:
```
def round_to_places(num,n):
    a=round(num,n)
    return(a)
round_to_places(3.14159,2) ## find
```

Print of index of a list :In Python index start from 0 and end length-1

In [ ]:
```
L=[5,8,3]
n=len(L)
n
```

In [ ]:
```
for i in range(n):
    print(i)
```

Sorting of a list

In [ ]:
```
#increasing order
L=[6,2,9,4,-8,-6,0,6,1,15,12]
t=[]
n=len(L)
for i in range(n):
    for j in range(n-1):
        if(L[j]>L[j+1]):
            L[j],L[j+1]=L[j+1],L[j]

print(L)
```

In [ ]:
```
#decreasing order
L=[6,2,9,4,-8,-6,0,6,1,15,12]
t=[]
n=len(L)
for i in range(n):
    for j in range(n-1):
        if(L[j]<L[j+1]):
            L[j],L[j+1]=L[j+1],L[j]

print(L)
```

Find maximum value?

In [ ]:
```
#for finding maximum value at first we sort the data in increasing order .The
n we print the last element of the list for maximum
# and print the first element of the List
L=[6,2,9,78,-0.24,89.99,4,-8,-6,0,6,1,15,12]

n=len(L)
for i in range(n):
    for j in range(n-1):
        if(L[j]>L[j+1]):
            L[j],L[j+1]=L[j+1],L[j]

n=len(L)
max=L[n-1]  #print the last element of the list(since, index start from 0 so
 we take n-1)
min=L[0]
print("increasing order of the list:",L ,"\n","maximum value of the list i
s:",max,"\n","minimum value is: ",min)


#in this programme we print sorting data, maximum value and minimum value.
```

Create a function for finding sorting list,maximum and minimum value of a list?

In [ ]:
```
#quit()
def sort(L):
    n=len(L)
    for i in range(n):
         for j in range(n-1):
                if(L[j]>L[j+1]):
                    L[j],L[j+1]=L[j+1],L[j]
    max=L[n-1]
    min=L[0]
    return(print("increasing order of the list:",L ,"\n","maximum value of t
he list is:",max,"\n","minimum value is: ",min))

k=[0,3,1,56,-78,-231,-1258.974,678.34,678.95]  ##callthe function
sort(k)
```

By inbuilt function maximum ,minimum,increasing and decreasing order of a list

In [ ]:
```
p=[6,-2,8,9,0,45,-33,90.45]
max(p)
```

In [ ]:
```
min(p)
```

In [ ]:
```
sorted(p,reverse=True) #decreasing order
```

In [ ]:
```
#alternative
p=[6,-2,8,9,0,45,-33,90.45]
p.sort(reverse=True)
p
```

In [ ]:
```
sorted(p,reverse=False)   #ascending order
```

In [ ]:
```
#Alternative

p=[6,-2,8,9,0,45,-33,90.45]
p.sort(reverse=False)
p
```

Roots of quadratic equation

In [ ]:
```
#creating a complex number
complex(4,8)
```

In [ ]:
```
complex(6)
```

In [ ]:
```
complex(0)
```

Solution of the quadratic equation $ax^2 + bx + c = 0$ is given by Sreedhar Acharaya's formula $x_1, x_2 = \frac{-b \pm \sqrt{(b^2 - 4ac)}}{2a}$ So,depending on the argument inside the squre root , the solutions is either real and distinct or ewual root or complex conjugate roots. At first we will import some function from math library.

In [ ]:
```
from math import sqrt
a=int(input('Enter the values of a\n?'))
b=int(input('Enter the values of b\n?'))
c=int(input("enter the value of c?"))
arg=(b**2)-4*a*c
d=2*a
s=abs(arg)
if(arg>0):
    print("Two distinct real rotts are=",(-b+sqrt(s))/d,(-b-sqrt(s))/d)
if(arg==0):
    print("equals roots are",-b/d,-b/d)
if(arg<0):
    print("Two complex roots are",complex(-b/d,sqrt(s)/d),complex(-b/d,-sqrt
(s)/d))
```

In [ ]:
```
quit()
import math
#from  math import *
#from math import sqrt
math.sqrt(9)
```

Statistical calculations

Mean ,variance and standard deviation:$mean = \frac{\sum x}{n}$ and
$variiance = \frac{\sum x^2}{n} - (\frac{\sum x}{n})^2 \; and \; standard \; deviation \; = \sqrt{variance}$

First we import math library as we will use sqrt function

Finding mean of a list

In [ ]:
```
#For finding mean it is neccessary to know the following code .suppose, we wa
nt to print the index 0,1,2,....10.
for i in range(0,11):
    print(i)
```

In [ ]:
```
d=[]
```

In [ ]:
```
import math
L=[5,9,1,0,5]
sum=0
n=len(L)
for i in range(n):
    sum=sum+L[i]
mean=sum/n
print("Mean =",mean)
```

Write a function for finding mean?

In [ ]:
```
def suman(L):
    sum=0
    n=len(L)
    for i in range(n):
        sum=sum+L[i]
    mean=sum/n
    return(mean)
l=[7,9,5]  #we want to find the mean of the list
suman(l)  #call the function
```

Find variance of a list?

In [ ]:
```
x=[7,9,2]
n=len(x)
y=[0]*n #create an empty list of same size
s1=0
s2=0
for i in range(n):
    s1=s1+x[i]
    y[i]=x[i]**2
    s2=s2+y[i]
mean=s1/n
msq=s2/n
var=msq-(mean**2)
print(var)
```

Find mean, variance and standard deviation simultaneously ?

For finding standard deviation we will import math library as we will use sqrt() function under math library.

In [ ]:
```
import math
x=[7,9,2]
n=len(x)
y=[0]*n #create an empty list of same size
s1=0
s2=0
for i in range(n):
    s1=s1+x[i]
    y[i]=x[i]**2
    s2=s2+y[i]
mean=s1/n
msq=s2/n
var=msq-(mean**2)
print("mean=",mean,"\n","variance=",var,"\n","Standard deviation=",math.sqrt
(var))
```

Calculate mean,variance and standard deviation of x which takes the values 1,3,5,...,25?

In [ ]:
```
x=list(range(1,27,2)) # creating the list
import math
n=len(x)
y=[0]*n #create an empty list of same size
s1=0
s2=0
for i in range(n):
    s1=s1+x[i]
    y[i]=x[i]**2
    s2=s2+y[i]
mean=s1/n
msq=s2/n
var=msq-(mean**2)
print("mean=",mean,"\n","variance=",var,"\n","Standard deviation=",math.sqrt
(var))
```

In [ ]:
```
Write a function for finding a mean ,standard deviation and variance of a li
st?
import math
def mvs(x):
    n=len(x)
    y=[0]*n #create an empty list of same size
    s1=0
    s2=0
    for i in range(n):
        s1=s1+x[i]
        y[i]=x[i]**2
        s2=s2+y[i]
    mean=s1/n
    msq=s2/n
    var=msq-(mean**2)
    return(print("mean=",mean,"\n","variance=",var,"\n","Standard deviation
=",math.sqrt(var)))
z=list(range(1,27,2)) # creating the list of numbers 1,3,...,25
mvs(z)
```

In [ ]:
```
# correlation page-82
```

In [ ]:
```

```

In [ ]:
```

```

For data outside

Now we will input data as a and b list.The can either be given as lists inside the python scriptor can be read from outside.THen we can apply different treatment like sum(),min() etc.

In [ ]:
```

```

In [3]:
```
x=[]
y=[]
n=int(input("how many numbers?\n"))
print("Enter the numbers a and b")
for i in range(n):
    a=input()
    b=input()
    x.append(a)
    y.append(b)
    print("a=",a,",","b=",b)
```

```
how many numbers?
3
Enter the numbers a and b
78
56
a= 78 , b= 56
89
2
a= 89 , b= 2
0
56
a= 0 , b= 56
```

Zip() function

The zip() function a combination list by collecting pairs of respective elements from the two list x and y.

In [6]:
```
######zip() not working ################
x=[5,9,7,9]
y=[0,1,2,3]
zip(x,y)
```

Out[6]:
```
<zip at 0x246cad40508>
```

Numpy function

see google:https://numpy.org/doc/stable/reference/routines.statistics.html (https://numpy.org/doc/stable/reference/routines.statistics.html)

Factorial

Write a function for finding factorial of n?

In [12]:
```
def fact(n):
    fact=1
    for i in range(2,n+1):
        fact=fact*i
    return(fact)
n=4
fact(4) #call the function
```

Out[12]:
```
24
```

Finding factorial by built-in function $factorial()$ under math library

In [14]:
```
import math  #import math library
math.factorial(4)
```

Out[14]:
```
24
```

In [ ]:

Due page:67 to 108 .book-Python:Abhijit kar gupta

In [1]:

Out[1]:
```
7.79
```

In [8]:

Out[8]:
```
3.14
```

Matrix in python

A matrix is a two-dimensional data structure where numbers are arranged into rows and columns.Python doesn't have a built-in type for matrices. However, we can treat list of a list as a matrix. For example: In this matrix "A" has 3 rows and 4 columns.

In [8]:
```
A = [[1, 4, 5, 12],
    [-5, 8, 9, 0],
    [-6, 7, 11, 19]]

print("A =", A)
print("A[1] =", A[1])      # 2nd row
print("A[1][2] =", A[1][2])   # 3rd element of 2nd row
print("A[0][-1] =", A[0][-1])   # Last element of 1st Row

print("\n","\n")  # for new line

#print third column
column = [];        # empty list
for row in A:
    column.append(row[2])

print("3rd column =", column)
```

```
A = [[1, 4, 5, 12], [-5, 8, 9, 0], [-6, 7, 11, 19]]
A[1] = [-5, 8, 9, 0]
A[1][2] = 9
A[0][-1] = 12



3rd column = [5, 9, 11]
```

Matrix addition

In Python, we can implement a matrix as a nested list (list inside a list). We can treat each element as a row of the matrix. For example X = [[1, 2], [4, 5], [3, 6]] would represent a 3x2 matrix. First row can be selected as X[0] and the element in first row, first column can be selected as X[0][0].We can perform matrix addition in various ways in Python. For addition of two matrix ,the rows and column are to be equal.The (i,j) element of the matrix is added with the same (i,j)th element of another to obtain the (i,j)th element of the new matrix:
$C_{i,j} = A_{i,j} + B_{i,j}$.
Here,To create the resuilting matrix,we have initialized matrix C before the for loop starts.We do that by creating the list C with elements all 0.

In [26]:
```
# Program to add two matrices using nested loop

A = [[12,7,3],
    [4 ,5,6],
    [7 ,8,9],
     [1,2,3]]

B = [[5,8,1],
    [6,7,3],
    [4,5,9],
     [4,5,6]]

C = [[0,0,0],  #creating null matrix of same order with
         [0,0,0],
         [0,0,0],
    [0,0,0]]

for i in range(len(A)):
    for j in range(len(A[0])):
        C[i][j]=A[i][j]+B[i][j]
print("Sum of these two matrix is") #print some comment
for k in C:
    print(k)
```

```
Sum of these two matrix is
[17, 15, 4]
[10, 12, 9]
[11, 13, 18]
[5, 7, 9]
```

List comprehension

List cpmprehension is nothing but some manipulation inside a list with logical statements,loops,function inside the list .Basically ,the entire expression to calculate matrix C is a kind of implied for loop inside the list! In this method did does not have to initialize the resulting matrix C.

In [27]:
```
A = [[12,7,3],
    [4 ,5,6],
    [7 ,8,9],[6,9,0]]

B = [[5,8,1],
    [6,7,3],
    [4,5,9],[8,2,1]]
C=[[A[i][j]+B[i][j] for j in range(len(A[0]))]for i in range(len(A))]
for k in C:
    print(k)
```

```
[17, 15, 4]
[10, 12, 9]
[11, 13, 18]
[14, 11, 1]
```

Matrix multiplication

open the link: https://www.programiz.com/python-programming/matrix (https://www.programiz.com/python-programming/matrix)

& https://www.tutorialspoint.com/python_data_structure/python_matrix.htm (https://www.tutorialspoint.com/python_data_structure/python_matrix.htm)

$C_{ij} = (AB)_i = \sum_{k=1}^{m} A_{ik} B_{kj}$,for the product of the two matrix the number of column of the first matrix has to be equal to the number of the column of the sceond matrix.
In Python, we can implement a matrix as nested list (list inside a list).We can treat each element as a row of the matrix.For example A = [[1, 2], [4, 5], [3, 6]] would represent a 3x2 matrix.The first row can be selected as A[0]. And, the element in first row, first column can be selected as B[0][0].Multiplication of two matrices A and B is defined only if the number of columns in A is equal to the number of rows B. If A is a n x m matrix and B is a m x l matrix then, AB is defined and has the dimension n x l (but BA is not defined). Here are a couple of ways to implement matrix multiplication in Python.

Algoritham

```
\begin{enumerate}
   \item Input:Two matrix in the form of a list.
   \item Input:Dimentions of matrix,matrix elements.
\end{enumerate}
```

In [6]:
```
# 3x3 matrix
A= [[12,7,3],[4 ,5,6],[7 ,8,9]]
# 3x4 matrix
B= [[5,8,1,2],
    [6,7,3,0],
    [4,5,9,1]]
# C is 3x4
C= [[0,0,0,0],
        [0,0,0,0],
        [0,0,0,0]]

# iterate through rows of A
for i in range(len(A)):
   # iterate through columns of B
   for j in range(len(B[0])):
       # iterate through rows of B
       for k in range(len(B)):
            C[i][j] += A[i][k] * B[k][j]

for r in C:
    print(r)
```

```
[114, 160, 60, 27]
[74, 97, 73, 14]
[119, 157, 112, 23]
```

In [132]:
```
def null_matrix(m,n):
    mat=[[0]*n]*m
    return(mat)
A=[[12,7,3],[4,5,6],[7,8,9]]  #create a 3*3 matrix
B=[[5,8,1,2],[6,7,3,0],[4,5,9,1]] #create B (3*4) matrix
m=len(A)
n=len(B[0])
p=len([B])
C=null_matrix(m,n)
print(C)
#C= [[0,0,0,0],[0,0,0,0],[0,0,0,0]]
#C=[[0]*len(B[0])]*len(A)

    # iterate through rows of A
for i in range(len(A)):
    for j in range(len(B[0])):
        for k in range(len(B)):
            C[i][j] += A[i][k] * B[k][j]
for r in C:
    print(r)
```

```
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
[307, 414, 245, 64]
[307, 414, 245, 64]
[307, 414, 245, 64]
```

??output is wrong

Matrix Multiplication Using Nested List Comprehension

In [2]:
```
#quit()
A=[[12,7,3],[4,5,6],[7,8,9]]  #create a 3*3 matrix
B=[[5,8,1,2],[6,7,3,0],[4,5,9,1]] #create B (3*4) matrix
m,p,n=len(A),len(B),len(B[0])
c=[[sum([A[i][k]*B[k][j] for k in range(p)])for j in range(n)]for i in range
(m)]
for row in c:
    print(row)
```

```
[114, 160, 60, 27]
[74, 97, 73, 14]
[119, 157, 112, 23]
```

Create a function for multiolication of two matrix?

In [3]:
```
 def matm(A,B):
        m,p,n=len(A),len(B),len(B[0])
        c=[[sum([A[i][k]*B[k][j] for k in range(p)])for j in range(n)]for i
in range(m)]
        for row in c:
            print(row)
#Multiply two matrix A and B
A=[[12,7,3],[4,5,6],[7,8,9]]  #create a 3*3 matrix
B=[[5,8,1,2],[6,7,3,0],[4,5,9,1]] #create B (3*4) matrix
matm(A,B)
```

```
[114, 160, 60, 27]
[74, 97, 73, 14]
[119, 157, 112, 23]
```

Tranpose of a matrix

For example X = [[1, 2], [4, 5], [3, 6]] would represent a 3x2 matrix. The first row can be selected as X[0]. And, the element in the first-row first column can be selected as X[0][0]. Transpose of a matrix is the interchanging of rows and columns. It is denoted as X'. The element at ith row and jth column in X will be placed at jth row and ith column in X'. So if X is a 3x2 matrix, X' will be a 2x3 matrix.

In [3]:
```
X = [[12,7],
    [4 ,5],
    [3 ,8]]
#B=[[0]*2]*3
#print(B)
B=[[0,0,0],[0,0,0]]
for i in range(len(X)):
    for j in range(len(X[0])):
        B[j][i]=X[i][j]
for k in B:
    print(k)
```

```
[12, 4, 3]
[7, 5, 8]
```

Transpose of a matrix by List comprehension

In [5]:
```
A=[[1,2,3],[4,5,6],[7,8,9],[9,34,78]]
B=[[A[j][i] for j in range(len(A))]for i in range(len(B[0]))]
for k in B:
    print(k)
```

```
[1, 4, 7, 9]
[2, 5, 8, 34]
[3, 6, 9, 78]
```

In [ ]:

DOUT IN MATRIX:HERE ALTHOUGH OUTPUT OF 'B' AND 'C' ARE EQUAL BUT THEY ARE NOT SAME

In [34]:
```
B=[[0]*2]*3
print(B)
C=[[0,0],[0,0],[0,0]]
C
```

```
[[0, 0], [0, 0], [0, 0]]
```

Out[34]:
```
[[0, 0], [0, 0], [0, 0]]
```

In [17]:

Out[17]:
```
0
```

Matrix operation by numpy package

$'numpy'$ is numerical python package for python to do numerical calculation >This is very usefull to multidimentional array management.Using numpy package we easily complete matrix addition,multiplication,matrix inverse etc.Once numpy package is installed in your computer, you cal import the package by'import numpy' and start using the modules with the reference to numpy.
To import numpy, and refer array() function ,we may write
$import\ numpy\ as\ np$
$np.array([\,])$

We can have array of many dimentions.Simple lists are one dimentional arrays. Example [9,7,8,0] is one dimention array.Nested lists are multidimentional array(List inside one list).For example X=[[1,2,3],[4,6,9]] is a two dimentional array(two lists inside one list). If we think of this as a matrix form then it can be written as $\begin{bmatrix}1 & 2 & 3\\4 & 6 & 9\end{bmatrix}$ Order of the matrix is 2×3 ,i.e, the dimention of the array X is 2×3.In other words, the 'shape' of the array is(2,3) ,i.e, an list of two elements,each of which is a list of three elements.
In the same way if we have,y=[[[1,2],[3,4]],[[5,6],[7,8]]], then y is a 3 dimensional array.The shape of Y is (2,2,2).

Writen an array X of dimention (2,3) or, matrix of 2×3 order? cheek it shape?

In [11]:
```
import numpy as np #install numpy library
X=np.array([[1,2,3],[4,6,9]])  # create the array
print("Array or the matrix is:",X,"\n")      ## "\n" for new line
print("Shape of rhe array or dimension of the matrix is:")
X.shape
```

```
Array or the matrix is: [[1 2 3]
 [4 6 9]]

Shape of rhe array or dimension of the matrix is:
```

Out[11]:
```
(2, 3)
```

Write a matrix Y of same order as X?

In [38]:
```
import numpy as np #install numpy library
Y=np.array([[11,12,13],[14,16,19]])  # create the array
print("Array or the matrix is:",Y)      ## "\n" for new line
```

```
Array or the matrix is: [[11 12 13]
 [14 16 19]]
```

In [36]:
```

```

```
[[11 12 13]
 [14 16 19]]
```

Matrix or array with comples numbers

In [18]:
```
A = np.array([[1, 2, 3], [3, 4, 5]], dtype = complex) # Array of complex num
bers
print(A)
```

```
[[1.+0.j 2.+0.j 3.+0.j]
 [3.+0.j 4.+0.j 5.+0.j]]
```

Array of dimension(m,n) with all elements are 0

In [19]:
```
import numpy as np
zeors_array = np.zeros( (2, 3) )  # here m=2 and n=3
print(zeors_array)
```

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

Array of dimension(m,n) with all elements are 1

In [21]:
```
import numpy as np
one_array = np.ones( (4, 3) )  # here m=4 and n=3
print(one_array)
```

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

Unit vector

We can create ei,the ith unit vector of length n using index.

import numpy as np i = 2 n = 4 x = np.zeros(n) x[i] = 1 print(x)

In [49]:
```
import numpy as np
i = 2
n = 4
x = np.zeros(n)
x[i] = 1
print(x)
```

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

Algebra with arrays

In [10]:
```
x+Y  #sum of the resprictive elements of arrays x and y, these is equal to matrix addition of X and Y.
```

Out[10]:
```
array([[12, 14, 16],
       [18, 22, 28]])
```

In [12]:
```
X*Y # product of the respective elements. these does to equakl to matrix product.
```

Out[12]:
```
array([[ 11,  24,  39],
       [ 56,  96, 171]])
```

Note: $X * Y$ is not a matrix product.The rules of matrix multiplication are different and we wiil use other function for matrix multiplication from numpy library.

In [14]:
```
1/X   #find an arrayv whose  each element are the receprocal of X.
```

Out[14]:
```
array([[1.        , 0.5       , 0.33333333],
       [0.25      , 0.16666667, 0.11111111]])
```

create a array H that is copy from Y?

In [39]:
```
import numpy as np #install numpy library
Y=np.array([[11,12,13],[14,16,19]])  # create the array
H=Y.copy()   #for coping we use copy() function

print(H)
```

```
[[11 12 13]
 [14 16 19]]
```

Vector equality

Equality of vectors is checked using the relational operator == (doubleequal signs). The Python expression evaluates to True if the expression on the left and right-hand side of the relational operator is equal, and to False otherwise.

In [40]:
```
import numpy as np
x = np.array([-1.1, 0.0, 3.6, -7.2])
y = x.copy()
x == y
```

Out[40]:
```
array([ True,  True,  True,  True])
```

Reshaping an array

In [15]:
```
import numpy as np
A=np.array([1,2,8.5,3,87,34,8,4])
A.shape
```

Out[15]: (8,)

Single element in the tuple(8,) implies thev array is one dimentional and contain 8 elements.Numbers of the elements in the tupple is equal to the dimension of the array .Next we reshaped the array into a two dimentional array with(2,4) by $reshape()$ function.

In [17]:
```
import numpy as np
A=np.array([1,2,8.5,3,87,34,8,4])
np.reshape(A,(2,4))
```

Out[17]:
```
array([[ 1. ,  2. ,  8.5,  3. ],
       [87. , 34. ,  8. ,  4. ]])
```

Change datatype of Array

In [42]:
```
# Create Float Matrix
A = np.array([3.0, 4.0, 5.0])
# Convert to Int
B = A.astype(np.int)
print("integer array is :",B)
#Convert to Float
C = A.astype(np.float)
print("float array is",C)
```

```
[3. 4. 5.]
integer array is : [3 4 5]
float array is [3. 4. 5.]
```

In [ ]:

arange() function in numpy library

In [25]:
```
import numpy as np
A = np.arange(4)
print('A =', A)
```

```
A = [0 1 2 3]
```

In [26]:
```
import numpy as np
B=np.arange(10)
B
```

Out[26]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

Write a function for creating an one dimension array of elements that are started $0, 1, 2...n$?
Using these array find an array of dimension (2,5)?

In [34]:
```
def arr(n):
    import numpy as np
    B=np.arange(n+1)
    return(B)
print(arr(9))
print("\n") #for new line
print("(2*5) dimension array is")
np.reshape(arr(9),(2,5))
```

```
[0 1 2 3 4 5 6 7 8 9]


(2*5) dimension array is
```

Out[34]:
```
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
```

1). Matrix operation Under numpy package(Any form of data numpy array)

i).Matrix addition

In [2]:
```
import numpy as np
X=np.array([[1,2,3],[4,6,9]])  # create a matrix X
Y=np.array([[11,12,13],[14,16,19]])  # create another matrix Y of same order
X+Y
```

Out[2]:
```
array([[12, 14, 16],
       [18, 22, 28]])
```

ii).Multiplication of Two Matrices

To multiply two matrices, we use dot() method. Learn more about how numpy.dot works. Note: * is used for array multiplication (multiplication of corresponding elements of two arrays) not matrix multiplication.

a). matrix multiplication by array() function

In [6]:
```
import numpy as np
A=np.array([[1,2,3],[4,6,9]])  # create a matrix A of order (2*3)
B=np.array([[11,12,13],[14,16,19],[10,12,14]])  # create another matrix B of order (3*2)
C=np.dot(A,B)
print(C)
```

```
[[ 69  80  93]
 [218 252 292]]
```

In [7]:
```
#Alternative code for multiplication of two matrix
A=np.array([[1,2,3],[4,6,9]])  # create a matrix A of order (2*3)
B=np.array([[11,12,13],[14,16,19],[10,12,14]])  # create another matrix B of order (3*2)
C=A.dot(B)
print(C)
```

```
[[ 69  80  93]
 [218 252 292]]
```

***Access rows of a Matrix**

In [18]:
```
 A = np.array([[1, 4, 5, 12],
    [-5, 8, 9, 0],
    [-6, 7, 11, 19]])

print("first row =", A[0]) # First Row
print("Sceond row =", A[2]) # Third Row
print("third row =", A[-1]) # Last Row (3rd row in this case)
```

```
first row = [ 1  4  5 12]
Sceond row = [-6  7 11 19]
third row = [-6  7 11 19]
```

***Access columns of a Matrix**

In [20]:
```
import numpy as np
A = np.array([[1, 4, 5, 12],
    [-5, 8, 9, 0],
    [-6, 7, 11, 19]])
print("first column =",A[:,0]) # First Column
print("Sceond column=", A[:,3]) # Fourth Column
print("Third column =", A[:,-1]) # Last Column (4th column in this case)
```

```
first column = [ 1 -5 -6]
Sceond column= [12  0 19]
Third column = [12  0 19]
```

b).Matrix multiplication by matrix() function under numpy package

There are seperate objects,called $matrix()$ in numpy.We may directly use this define a matrix and preceed for matrix operations as we write mathematically.In the following,the demonstration is done on python interpreter in order to understand the $matrix()$ array operations.

In [16]:
```
import numpy as np # import numpy library
A=np.matrix([[1,2,3],[4,6,9]])  # create a matrix A of order (2*3)
print("Our first matrix is","\n",A)
print("\n")
B=np.matrix([[11,12,13],[14,16,19],[10,12,14]])  # create another matrix B o
f  order (3*2)
print("Our sceond matrix is:","\n",B)
print("\n")
product=A*B
print("Product of A and B is","\n",product)
```

```
Our first matrix is
 [[1 2 3]
 [4 6 9]]


Our sceond matrix is:
 [[11 12 13]
 [14 16 19]
 [10 12 14]]


Product of A and B is
 [[ 69  80  93]
 [218 252 292]]
```

iii).Slicing of a Matrix

Slicing of a one-dimensional NumPy array is similar to a list. If you don't know how slicing for a list works. Let us see the scaling of one dimentional array.

In [21]:
```
import numpy as np
letters = np.array([1, 3, 5, 7, 9, 7, 5])

# 3rd to 5th elements
print(letters[2:5])        # Output: [5, 7, 9]

# 1st to 4th elements
print(letters[:-5])        # Output: [1, 3]

# 6th to last elements
print(letters[5:])         # Output:[7, 5]

# 1st to last elements
print(letters[:])          # Output:[1, 3, 5, 7, 9, 7, 5]

# reversing a list
print(letters[::-1])          # Output:[5, 7, 9, 7, 5, 3, 1] that is the rev
erse of the list or one domentional array.
```

```
[5 7 9]
[1 3]
[7 5]
[1 3 5 7 9 7 5]
[5 7 9 7 5 3 1]
```

Now, let's see how we can slice a matrix or more than one dimention array.

In [27]:
```
import numpy as np
A = np.array([[1, 4, 5, 12, 14],
    [-5, 8, 9, 0, 17],
    [-6, 7, 11, 19, 21]])
print("two rows(1st and 2nd) and four columns are","\n",A[:2, :4])  # two ro
ws, four columns\
print("\n")
print("first row and all column is:",A[:1,])  # first row, all columns
print("\n")
print("all rows and sceond column",A[:,2])  # all rows, second column
print("\n")
print("all rows ,3rd to 5th column:","\n",A[:, 2:5])  # all rows, third to t
he fifth column
```

```
two rows(1st and 2nd) and four columns are
 [[ 1  4  5 12]
 [-5  8  9  0]]


first row and all column is: [[ 1  4  5 12 14]]


all rows and sceond column [ 5  9 11]


all rows ,3rd to 5th column:
 [[ 5 12 14]
 [ 9  0 17]
 [11 19 21]]
```

iv).Tranpose of matrix

In [32]:
```
import numpy as np # import library
A = np.array([[1, 4, 5, 12, 14],
    [-5, 8, 9, 0, 17],
    [-6, 7, 11, 19, 21]])
print("Our matrix is","\n",A)
B=np.transpose(A)
print("Tranpose of the matrix is:","\n",B)
```

```
Our matrix is
 [[ 1  4  5 12 14]
 [-5  8  9  0 17]
 [-6  7 11 19 21]]
Tranpose of the matrix is:
 [[ 1 -5 -6]
 [ 4  8  7]
 [ 5  9 11]
 [12  0 19]
 [14 17 21]]
```

v).Inverse of matrix

In [35]:
```
import numpy as np
A = np.array([[1, 4, 5],
    [-5, 8, 9],
    [-6, 7, 11]])
print("Our matrix is","\n",A)
D = np.linalg.inv(A)
print("Inverse of the matrix is:","\n",D)
```

```
Our matrix is
 [[ 1  4  5]
 [-5  8  9]
 [-6  7 11]]
Inverse of the matrix is:
 [[ 0.26595745 -0.09574468 -0.04255319]
 [ 0.0106383   0.43617021 -0.36170213]
 [ 0.13829787 -0.32978723  0.29787234]]
```

Cheek D is Inverse of A .We know, $AA^{-1} = I_n$

In [38]:
```
import numpy as np
A = np.array([[1, 4, 5], [-5, 8, 9],[-6, 7, 11]])
print("Our matrix is","\n",A,"\n")
D = np.linalg.inv(A)
E=np.dot(A,D)
print("Inverse of the matrix is:","\n",D,"\n")
print("Multiplication of A and Of its inverse:","\n",E)
```

```
Our matrix is
 [[ 1  4  5]
 [-5  8  9]
 [-6  7 11]]

Inverse of the matrix is:
 [[ 0.26595745 -0.09574468 -0.04255319]
 [ 0.0106383   0.43617021 -0.36170213]
 [ 0.13829787 -0.32978723  0.29787234]]

Multiplication of A and Of its inverse:
 [[ 1.00000000e+00 -1.11022302e-16  3.33066907e-16]
 [ 2.77555756e-17  1.00000000e+00 -1.11022302e-16]
 [ 8.32667268e-17 -1.11022302e-16  1.00000000e+00]]
```

Here , we see that $AA^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = I_3$

Hence , D is Inverse of A

vi).System of linear equation :see the link below

http://nebomusic.net/perception/Matrix_Operations_Python_Numpy.pdf
(http://nebomusic.net/perception/Matrix_Operations_Python_Numpy.pdf)

In [45]:

Out[45]:
```
array([[4., 9., 0., 9.],
       [3., 3., 9., 3.],
       [5., 5., 1., 5.]])
```

vii).Basic operation:Eigen value and eigen vector

For finding eigen values and vector recuire two packages:
$numpy\ packages \& \ linalg from numpy package$

In [59]:
```
A = np.array([[1, 4, 5], [-5, 8, 9],[-6, 7, 11]])
print("Our matrix is","\n",A,"\n")
eigvals, eigvecs = np.linalg.eig(A)
print ("Eigen values are:",eigvals,"\n")
print("Eigen vectors corresponding to the eigen values are","\n",eigvecs)
```

```
Our matrix is
 [[ 1  4  5]
 [-5  8  9]
 [-6  7 11]]

Eigen values are: [13.59373746  5.03209301  1.37416954]

Eigen vectors corresponding to the eigen values are
 [[ 0.45145779  0.83661458  0.10258363]
 [ 0.62348353  0.44632316 -0.77299039]
 [ 0.63832135  0.31760303  0.62606905]]
```

One can also use the following code :

Here,We will directly imprt linalg as Lg from numpy package,i.e from numpy package install linalg:these line implies:LA=np.linalg,so we use LA instead of np.linalg

In [7]:
```
import numpy as np #import numpy package
A = np.array([[1, 4, 5], [-5, 8, 9],[-6, 7, 11]])
print("Our matrix is","\n",A,"\n")
from numpy import linalg as LA
eigvals, eigvecs = LA.eig(A)
print ("Eigen values are:",eigvals,"\n")
print("Eigen vectors corresponding to the eigen values are","\n",eigvecs)
```

```
Our matrix is
 [[ 1  4  5]
 [-5  8  9]
 [-6  7 11]]

Eigen values are: [13.59373746  5.03209301  1.37416954]

Eigen vectors corresponding to the eigen values are
 [[ 0.45145779  0.83661458  0.10258363]
 [ 0.62348353  0.44632316 -0.77299039]
 [ 0.63832135  0.31760303  0.62606905]]
```

In [55]:
```
eigvals[0] # print the first eigen value
```

Out[55]:
```
5.032093008023611
```

In [56]:
```
eigvals[1] #print the sceond eigen value
```

Out[56]:
```
5.032093008023611
```

In [57]:
```
eigvals[2] #print the third eigen value
```

Out[57]:
```
1.3741695350624064
```

In [8]:
```
#print the first eigen vector.
import numpy as np
A = np.array([[1, 4, 5], [-5, 8, 9],[-6, 7, 11]])
v1 = eigvecs[:,0] # First column is the first eigenvector
print(v1)
```

```
[0.45145779 0.62348353 0.63832135]
```

for find all the eigen values and eigen vector of the squre matix A?

In [67]:
```
for i in range(len(A)):  #len(A) is the number rows or column in the matrix
 A.
    print("the eigen vector is",eigvecs[:,i],"corresponding to the eigen val
ue",eigvals[i])
```

```
the eigen vector is [0.45145779 0.62348353 0.63832135] corresponding to the
eigen value 13.593737456913974
the eigen vector is [0.83661458 0.44632316 0.31760303] corresponding to the
eigen value 5.032093008023611
the eigen vector is [ 0.10258363 -0.77299039  0.62606905] corresponding to t
he eigen value 1.3741695350624064
```

Create a function for finding eigen values and eigen vector of a squre matix? using these function find the eigen values and eigen vactors of a matrix B?

In [9]:
```
def eiv(matrix):
    n=len(A)
    import numpy as np
    eigvals, eigvecs = np.linalg.eig(matrix)  # linalag is a package for fin
ding eigen value and eigen vector
    for i in range(n):
        print("the eigen vector is",eigvecs[:,i],"corresponding to the eigen
 value",eigvals[i])
import numpy as np
B= np.array([[2, 0, 0], [-1, 3, 1],[-1, 1, 3]])
eiv(B)
```

```
the eigen vector is [0.         0.70710678 0.70710678] corresponding to the
eigen value 4.0
the eigen vector is [ 0.          0.70710678 -0.70710678] corresponding to t
he eigen value 2.0
the eigen vector is [0.81649658 0.40824829 0.40824829] corresponding to the
eigen value 2.0
```

One can use the code which is given below aslo.These two code are almost same .

In [10]:
```
def eiv(matrix):
    n=len(matrix)
    import numpy as np
    from numpy import linalg as LA  #from numpy package  install linalg:thes
e line means:LA=np.linalg
    eigvals, eigvecs = LA.eig(matrix)  # eigen value and eigen vector
    for i in range(n):
        print("the eigen vector is",eigvecs[:,i],"corresponding to the eigen
 value",eigvals[i])
import numpy as np
B= np.array([[2, 0, 0], [-1, 3, 1],[-1, 1, 3]])
eiv(B)
```

```
the eigen vector is [0.         0.70710678 0.70710678] corresponding to the
eigen value 4.0
the eigen vector is [ 0.          0.70710678 -0.70710678] corresponding to t
he eigen value 2.0
the eigen vector is [0.81649658 0.40824829 0.40824829] corresponding to the
eigen value 2.0
```

PANDAS MODULE IN PYTHON

Reference:Python Data Analytics_ Data Analysis and Science Using Pandas, matplotlib, and the Python Programming Language (PDFDrive) book in my laptop link:https://www.journaldev.com/29055/python-pandas-module-tutorial (https://www.journaldev.com/29055/python-pandas-module-tutorial)

https://stackabuse.com/beginners-tutorial-on-the-pandas-python-library/ (https://stackabuse.com/beginners-tutorial-on-the-pandas-python-library/)

Pandas is an open source library in Python. It provides ready to use high-performance data structures and data analysis tools. There are 3 data structures provided by the Pandas module, which are as follows:

1. Series:

It is a 1-D size-immutable array like structure having homogeneous data.

2. DataFrames:

It is a 2-D size-mutable tabular structure with heterogeneously typed columns.

3.Panel:

It is a 3-D, size-mutable array.

Series

The Series is the object of the pandas library designed to represent one-dimensional data structures of any data types(integer, string, float, python objects, etc.),similarly to an array but with some additional features. The axis labels are collectively called index. Pandas Series is nothing but a column in an excel sheet.
To create a Pandas Series, we must first import the Pandas package via the Python's import command:
$import\ pandas\ as\ pd$

To create the Series, we wii use the following code $pd.Series()$ m, as shown below:
It is very carefull that the first letter "S" in series function is capital.

In [4]:
```
import pandas as pd #import pandas library
series=pd.Series([1,2,3,4])
print(series)
```

```
0    1
1    2
2    3
3    4
dtype: int64
```

You can see that we have two columns, the first one with numbers starting from index 0 and the second one with the elements that were added to the series.

The first column denotes the indexes for the elements.

In [5]:
```
# Include index
s = pd.Series([12,-4,7,9], index=['a','b','c','d'])
print(s)
```

```
a    12
b    -4
c     7
d     9
dtype: int64
```

If you want to individually see the two arrays that make up this data structure you can call the two attributes of the Series as follows: index and values.

In [4]:
```
s.values
```

Out[4]:
```
array([12, -4,  7,  9], dtype=int64)
```

In [6]:
```
s.index
```

Out[6]:
```
Index(['a', 'b', 'c', 'd'], dtype='object')
```

Selecting internal element

Suppose,we want to know the values of the array by their index then we will use the following code that is given below

In [7]:
```
s[0]#first element of the array
```

Out[7]:
```
12
```

Or you can specify the label corresponding to the position of the index.

In [8]:
```
s['a']
```

Out[8]:
```
12
```

In the same way you select multiple items

In [9]:
```
s[0:2]
```

Out[9]:
```
a    12
b    -4
dtype: int64
```

or even in this case, use the corresponding labels, but specifying the list of labels within an array.

In [10]:
```
s[['b','c']]
```

Out[10]:
```
b   -4
c    7
dtype: int64
```

Assigning Values to the Elements

Now that you understand how to select individual elements, you also know how to assign new values to them. In fact, you can select the value by index or label.

In [16]:
```
import pandas as pd #import library
s = pd.Series([12,-4,7,9], index=['a','b','c','d']) #create an array s
print(s,"\n")
#replacing first element in the array by 0
s[1]=0
print(s)
```

```
a    12
b    -4
c     7
d     9
dtype: int64

a    12
b     0
c     7
d     9
dtype: int64
```

Or you can specify the label corresponding to the position of the index.

In [17]:
```
s['b']=0
print(s)
```

```
a    12
b     0
c     7
d     9
dtype: int64
```

Filtering Values

For example, if you need to know which elements within the series have value greater than 8, you will write the following:

In [32]:
```
import pandas as pd #import library
s = pd.Series([12,-4,7,9], index=['a','b','c','d']) #create an array s
print(s,"\n")
#print the elements within the series s have value greater than 8,
print(s[s>8])
```

```
a    12
b    -4
c     7
d     9
dtype: int64

a    12
d     9
dtype: int64
```

if we want answer these question by logical vector then we will use the code that is given below-

In [33]:
```
print(s>8)
```

```
a     True
b    False
c    False
d     True
dtype: bool
```

copy from an array

Create an array y that is copy from x?

In [29]:
```
y=s3.copy()
y
```

Out[29]:
```
0    1
1    2
2    3
3    4
dtype: int32
```

Defining Series from NumPy Arrays

In [37]:
```
import numpy as np #import numpy library for define an arry
ar=np.array([1,2,3,4])
import pandas as pd
a=pd.Series(ar)
print(a)
```

```
0    1
1    2
2    3
3    4
dtype: int32
```

Operations and Mathematical Functions

Other operation such as +,-.*,/ are possible for these series

Create an series w whose elements are half of elements s?

In [44]:
```
import pandas as pd #import library
s = pd.Series([12,34,7,9], index=['a','b','c','d']) #create an array s
print(s,"\n")  # to see the array s
#Create an series w whose elements are half of elements s
w=s/2
print("Answer is:","\n",w)
```

```
a    12
b    34
c     7
d     9
dtype: int64

Answer is:
 a     6.0
b    17.0
c     3.5
d     4.5
dtype: float64
```

Create a series x whose elements are logarithem values of the series s?

For calculating logarithametic value we import numpy library

In [47]:
```
import pandas as pd #import library
s = pd.Series([12,34,7,9], index=['a','b','c','d']) #create an array s
print("Our given series is","\n")  # to see the array s
#Create a series x whose elements are logarithem values of the series s
import numpy as np
np.log(s)
```

```
Our given series is


a    2.484907
b    3.526361
c    1.945910
d    2.197225
dtype: float64
```

In [45]:
```
import numpy as np
np.log(s)
```

```
a    2.484907
b    3.526361
c    1.945910
d    2.197225
dtype: float64
```

Unique values within a series

To know all the values contained within the Series excluding duplicates, you can use the unique( ) function. The return value is an array containing the unique values in the Series, though not necessarily in order.

In [7]:
```
import pandas as pd
serd = pd.Series([1,0,2,1,2,3], index=['white','white','blue','green','gree
n','yellow']) #create a series serd
print("The series is\n",serd,"\n")
#unique values of the series
uni=serd.unique()
print("unique values of the series is:",uni)
```

```
The series is
 white     1
white     0
blue      2
green     1
green     2
yellow    3
dtype: int64

unique values of the series is: [1 0 2 3]
```

A function similar to unique( ) is the value_counts( ) function, which not only returns the unique values that is given below but calculates occurrences within a Series.

```
In [11]: serd.value_counts()
```

```
Out[11]: 2    2
         1    2
         3    1
         0    1
         dtype: int64
```

by value_counts() function we calculate 2 present 2 times,1 present 2 times, 3 present 1 times,0 present 1 times in our gevien series s.

Evaluating Values

Finally, isin( ) is a function that evaluates the membership, that is, given a list of values, this function lets you know if these values are contained within the data structure. Boolean values that are returned can be very useful during the filtering of data within a series or in a column of a DataFrame.

Find the index of the array "serd" whoose values are 0 and 3?

```
In [12]: serd.isin([0,3])
```

```
Out[12]: white     False
         white      True
         blue      False
         green     False
         green     False
         yellow     True
         dtype: bool
```

```
In [14]: # more preciseouly if we want to better formet ins stead of ligical output.
         serd[serd.isin([0,3])]
```

```
Out[14]: white     0
         yellow    3
         dtype: int64
```

Nan values

Suppose we define a series s that is given below:

In [15]:
```
import pandas as pd #import library
s = pd.Series([12,-4,7,9], index=['a','b','c','d']) #create an array s
print(s)
```

```
a    12
b    -4
c     7
d     9
dtype: int64
```

Now we will create a series 't' whose element are logaritham values of the series 's' that is given below:

In [16]:
```
import numpy as np #for logaritham function we import library numpy
t=np.log(s)
print(t)
```

```
C:\Users\SUMAN GHOSH\Anaconda3\lib\site-packages\pandas\core\series.py:853:
RuntimeWarning: invalid value encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Out[16]:
```
a    2.484907
b         NaN
c    1.945910
d    2.197225
dtype: float64
```

Here ,we see that the values of the in 'b' is missing values(or Nan values) within the series 't' as we know that logarithm of a negative number is undefined.This specific value NaN (Not a Number) is used within pandas data structures to indicate the presence of an empty field or not definable numerically.

Despite their problematic nature, however, pandas allows to explicitly define and add this value in a data structure, such as Series. Within the array containing the values you enter $np.NaN$wherever we want to define a missing value.

Now we will define a series s2 with nan values

In [17]:
```
import pandas as pd
s2 = pd.Series([5,-3,np.NaN,14])
print(s2)
```

```
0     5.0
1    -3.0
2     NaN
3    14.0
dtype: float64
```

Finding the index of NaN values

The $isnull()$ and$notnull()$functions are very useful to identify the indexes without a value.these two functions return the Series with Boolean values that contains the ‘True’ and ‘False’ valuesdepending on whether the item is a NaN value or less. The isnull( ) function returns ‘True’ at NaN values inthe Series; inversely, the notnull( ) function returns ‘True’ if they are not NaN.

```
In [18]: s2.isnull( )
```

```
Out[18]: 0    False
         1    False
         2     True
         3    False
         dtype: bool
```

```
In [19]: s2.notnull()
```

```
Out[19]: 0     True
         1     True
         2    False
         3     True
         dtype: bool
```

or, more preciseously

```
In [20]: s2[s2.isnull( )]
```

```
Out[20]: 2   NaN
         dtype: float64
```

```
In [21]: s2[s2.notnull()]
```

```
Out[21]: 0     5.0
         1    -3.0
         3    14.0
         dtype: float64
```

Series as a dictionary : see book- Python Data Analytics_ Data Analysis and Science Using Pandas, matplotlib, and the Python Programming Language (PDFDrive) in my laptop

Operations between Series:

We have seen how to perform arithmetic operations between Series and scalar values. Here, we wiil create two series 's' and 't' then we will sum of these two series.

In [23]:
```
#create a series 's'
import pandas as pd
s=pd.Series([200,100,70,30,50],index=["red","yellow","black","blue","green"
])
print(s)
```

```
red       200
yellow    100
black      70
blue       30
green      50
dtype: int64
```

In [24]:
```
#create s series 't'
import pandas as pd
t=pd.Series([60,345,90,78],index=["pink","orrange","red","yellow",])
print(t)
```

```
pink        60
orrange    345
red         90
yellow      78
dtype: int64
```

In [26]:
```
#sum of these two series
print("sum is:",s+t,"\n")
```

```
sum is: black        NaN
blue         NaN
green        NaN
orrange      NaN
pink         NaN
red        290.0
yellow     178.0
dtype: float64
```

Here, we see that the sum of the two series returns numerical valus if both of the series have same lebels or index. And all other label present in one of the two series are still added to the result but have a NaN value.

Here, both the series s and t have same index "red" and "yellow".These are added respectively

Dataframe

The Pandas DataFrame can be seen as a table. It organizes data into rows and columns, making it a two-dimensional data structure. Potentially, the columns are of a different types and the size of the DataFrame is mutable, and hence can be modified.

1. Method for creating dataframe

the common method for creating a data frame is given below-

data = {'First Column Name': ['First value', 'Second value',...],
'Second Column Name': ['First value', 'Second value',...], ...
}

df = pd.DataFrame (data)

print (df)

https://datatofish.com/create-pandas-dataframe/ (https://datatofish.com/create-pandas-dataframe/) Let us see the example below

In [3]:
```
data = {'color' : ['blue','green','yellow','red','white'],
'object' : ['ball','pen','pencil','paper','mug'],
'price' : [1.2,1.0,0.6,0.9,1.7]}
import pandas as pd
df=pd.DataFrame(data)
print(df)
```

```
    color  object  price
0    blue    ball    1.2
1   green     pen    1.0
2  yellow  pencil    0.6
3     red   paper    0.9
4   white     mug    1.7
```

Here, we can not define index names . pandas automatically assigns a numeric sequence starting from 0. Instead, if you want to assign labels to the indexes of a DataFrame, you have to use the index option assigning it an array containing the labels. That is given below:

In [6]:
```
import pandas as pd
data = {'color' : ['blue','green','yellow','red','white'],
'object' : ['ball','pen','pencil','paper','mug'],
'price' : [1.2,1.0,0.6,0.9,1.7]}
frame=pd.DataFrame(data,index=["one","two","three","four","five"])
print(frame)
```

```
        color  object  price
one      blue    ball    1.2
two     green     pen    1.0
three  yellow  pencil    0.6
four      red   paper    0.9
five    white     mug    1.7
```

Create a dataframe from a list.

It is possible for us to create a DataFrame from a list or even a set of lists. We only have to call the function pd.DataFrame().Consider the following example:

In [8]:
```
l=["ram","shyam","jodu","madhu"] #creating a list
import pandas as pd
d=pd.DataFrame(l,index=["r1","r2","r3","r4"])
print(d)
```

```
        0
r1    ram
r2  shyam
r3   jodu
r4  madhu
```

Create a dataframe from list of Dicts

List of Dictionaries can be passed as input data to create a DataFrame. The dictionary keys are by default taken as column names. Let see the following example below-

- Example 1 :Create dataframe without index

In [10]:
```
import pandas as pd
data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
df1 = pd.DataFrame(data)
print(df1)
```

```
   a   b     c
0  1   2   NaN
1  5  10  20.0
```

Here, we notice that in this dataframe column "c" does not have any values by default it will take NaN (not a number).

- Example 2:The following example shows how to create a DataFrame by passing a list of dictionaries and the row indices.

In [15]:
```
import pandas as pd
data2 = [{'a': 1, 'b': 2,'c':67},{'a': 5, 'b': 10, 'c': 20}]
df2= pd.DataFrame(data2, index=['first', 'second'])
print(df2)
```

```
        a   b   c
first   1   2  67
second  5  10  20
```

Create a DataFrame from Dict of Series

Dictionary of Series can be passed to form a DataFrame. The resultant index is the union of all the series indexes passed.

In [16]:
```
#Example
import pandas as pd

d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
   'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}

df3 = pd.DataFrame(d)
print(df3)
```

```
   one  two
a  1.0    1
b  2.0    2
c  3.0    3
d  NaN    4
```

$Note$ − Observe, for the series one, there is no label ‘d’ passed, but in the result, for the d label, NaN is appended with NaN.

Selecting Elements of data frame by pandas library

1). Name of the columns of a data frame .

To know name of the all columns of a dataframe we will use $columns$ function. Look at the example below- In this example at first we will create a data frame 'df'.The we will determine the names of all columns.

In [5]:
```
data = {'color' : ['blue','green','yellow','red','white'],
'object' : ['ball','pen','pencil','paper','mug'],
'price' : [1.2,1.0,0.6,0.9,1.7]}
import pandas as pd
df=pd.DataFrame(data,index=['one','two','three','four','five'])
print(df,"\n") #"\n" for new line
columns_names=df.columns
print("the columns names are:",columns_names)
```

```
        color  object  price
one      blue    ball    1.2
two     green     pen    1.0
three  yellow  pencil    0.6
four      red   paper    0.9
five    white     mug    1.7

the columns names are: Index(['color', 'object', 'price'], dtype='object')
```

2).Name of the rows or index

Similarly, we can know the index or rows using the function $index$

In [26]:
```
df.index # in a data frame index are called as rows
```

Out[26]:
```
Index(['one', 'two', 'three', 'four', 'five'], dtype='object')
```

3).how to know Dimention or order of numbers of rows and column of a dataframe?

we will use $shape$ function that is given below.....

In [9]:
```
#create a dataframe
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    "per_marks":[60,70,35,90,81,77,79],
    "attendence":[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
print(std,"\n \n")
std.shape
```

```
        roll number  registraction no  age  doc_id no  per_marks  attendence
ram               2                 1   22          12         60          45
shyam             4                 3   21           3         70          78
jodu              6                 5   20           5         35          67
madhu             8                 7   23           6         90          89
laxyam            7                12   27           2         81          75
rohit             9                23   32          67         77          78
rimpa            13                14   25          34         79          82
```

Out[9]:
```
(7, 6)
```

these function return numbers of rows and column in a data frame

4).Display the table of values by row wise

In [11]:
```
std.values
```

Out[11]:
```
array([[ 2,  1, 22, 12, 60, 45],
       [ 4,  3, 21,  3, 70, 78],
       [ 6,  5, 20,  5, 35, 67],
       [ 8,  7, 23,  6, 90, 89],
       [ 7, 12, 27,  2, 81, 75],
       [ 9, 23, 32, 67, 77, 78],
       [13, 14, 25, 34, 79, 82]], dtype=int64)
```

5).How we know a single value(single value) within a data frame?

To know a single value within a data frame first you have use the name ofthe column and then the index or the label of the row. There are two method to access a scaler value of a dataframe these are-

- at method: It needs to the lebels of row and column. and
- iat method: It needs to the index of row and column.

a).at method:

Function notation :$nane\ of\ dataframe.\ at[”\ row\ lables, column\ lables]$. If you sure about lebels the you apply iat function.
Example: find roll number of ram of std dataframe?

In [16]:
```
#creating a data frame std
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    "per_marks":[60,70,35,90,81,77,79],
    "attendence":[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
print(std,"\n \n")
#what is the roll number of the student ram?
std.at["ram","roll number"]
```

	roll number	registraction no	age	doc_id no	per_marks	attendence
ram	2	1	22	12	60	45
shyam	4	3	21	3	70	78
jodu	6	5	20	5	35	67
madhu	8	7	23	6	90	89
laxyam	7	12	27	2	81	75
rohit	9	23	32	67	77	78
rimpa	13	14	25	34	79	82

Out[16]: 2

One can use the following function $name\ of\ dataframe[\text{" }column\ lables\text{ "}][\text{" }row\ lables\text{ "}]$ also

In [14]:
```
std['roll number']["shyam"]
```

Out[14]: 4

b).iat method:

Function notation :$nane\ of\ dataframe.\ at[\text{" }row\ lables, column\ lables]$. If you sure about index the you apply iat function.
Example: find roll number of ram of std dataframe?

In [17]:
```
std
std.iat[0,0]
```

Out[17]: 2

In [ ]:
```

```

6). Display the unique values of a column

For finding unique value of a dataframe we wii use $name\ of\ dataframe[\text{" }column\ name\text{ "}].\ unique()$ function.

- find the unique value of age column of std dataframe?

In [15]:
```
#creating a data frame std
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,22,32,22],
    'doc_id no':[12,3,5,6,2,67,34],
    "per_marks":[60,70,35,90,81,77,79],
    "attendence":[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
print("our given datagrame is:\n",std,"\n \n")
#unique values of age column
print("unique value of age column is:")
std['age'].unique()
```

```
our given datagrame is:
         roll number  registraction no  age  doc_id no  per_marks  attendenc
e
ram                2                 1   22         12         60          45
shyam              4                 3   21          3         70          78
jodu               6                 5   20          5         35          67
madhu              8                 7   23          6         90          89
laxyam             7                12   22          2         81          75
rohit              9                23   32         67         77          78
rimpa             13                14   22         34         79          82


unique value of age column is:
```

Out[15]:
```
array([22, 21, 20, 23, 32], dtype=int64)
```

7). Column Selection

- Selection of a single column :
  determine the value of the 'price' column of 'df' data frame?

In [33]:
```
std["roll number"]
```

Out[33]:
```
ram        2
shyam      4
jodu       6
madhu      8
laxyam     7
rohit      9
rimpa     13
Name: roll number, dtype: int64
```

As you can see, the return value is a Series object. Another way is to use the column name as an attribute of the instance of the DataFrame

In [29]:
```
std.attendence  # if we run the comment std.roll number  then we get an error as .we space is not alloud.
```

Out[29]:
```
ram       45
shyam     78
jodu      67
madhu     89
laxyam    75
rohit     78
rimpa     82
Name: attendence, dtype: int64
```

- b).Selection of multiple columns simultineously

In [ ]:
```

```

8).Selecting rows

Regarding the rows within a data frame, it is possible to use the $ix[]$ attribute with the index value of the row that you want to extract

a).Selecting a particular row

In [62]:
```
import pandas as pd
d = {'roll number' :[2,4,6,8],
   'registraction no' :[1,3,5,7]}
student = pd.DataFrame(d,index=["ram","shyam","jodu","madhu"])
print(student)
```

```
       roll number  registraction no
ram              2                 1
shyam            4                 3
jodu             6                 5
madhu            8                 7
```

Suppose we want to know registraction and roll number of the student jodu. then we will use ix[ ] function

In [45]:
```
student.ix["jodu"]
```

```
C:\Users\SUMAN GHOSH\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: Fu
tureWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ix-inde
xer-is-deprecated
  """Entry point for launching an IPython kernel.
```

Out[45]:
```
roll number         6
registraction no    5
Name: jodu, dtype: int64
```

- selection by index

In [63]:
```
student.iloc[2]
```

Out[63]:
```
roll number         6
registraction no    5
Name: jodu, dtype: int64
```

b).Selection more than one rows

here, for problem in ix function we use $iloc[\ ]$ function.

In [68]:
```
student.iloc[0:2]  # select rirst and sceond rows.
```

Out[68]:

	roll number	registraction no
ram	2	1
shyam	4	3

######dout loc function??

https://stackabuse.com/beginners-tutorial-on-the-pandas-python-library/ (https://stackabuse.com/beginners-tutorial-on-the-pandas-python-library/)

d).select first few rows by $head(numbers\ of\ rows)$ ***function***

print only the first few rows on the console rather than printing all the rows by using head() function.

In [86]:
```
#creating a data frame std
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    "per_marks":[60,70,35,90,81,77,79],
    "attendence":[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
print(std)
```

```
        roll number  registraction no  age  doc_id no  per_marks  attendence
ram               2                 1   22          12         60          45
shyam             4                 3   21           3         70          78
jodu              6                 5   20           5         35          67
madhu             8                 7   23           6         90          89
laxyam            7                12   27           2         81          75
rohit             9                23   32          67         77          78
rimpa            13                14   25          34         79          82
```

- *print the first 3 rows of std dataframe?*
  for printing the first 3 rows of std datafram we will use head() function that is given below.....

In [82]:
```
z=std.head(3)
print("first three rows are:\n",z)
```

```
first three rows are:
        roll number  registraction no  age
ram               2                 1   22
shyam             4                 3   21
jodu              6                 5   20
```

- note: if we will define number of rows then it will take 5. head() function return first 5 rows. that is given below

In [83]:
```
std.head()
```

Out[83]:

	roll number	registraction no	age
ram	2	1	22
shyam	4	3	21
jodu	6	5	20
madhu	8	7	23
laxyam	7	12	27

e).select last few rows by $tail(numbers\ of\ rows)$ function

print only the first few rows on the console rather than printing all the rows by using head() function.

- find last 4 rows of std dataframe?

```
In [90]: print("last 4 rows of std dataframe are:\n",std.tail(4))
```

```
last 4 rows of std dataframe are:
         roll number  registraction no  age  doc_id no  per_marks  attendenc
e
madhu              8                 7   23          6         90          89
laxyam             7                12   27          2         81          75
rohit              9                23   32         67         77          78
rimpa             13                14   25         34         79          82
```

- Note:note: if we will define number of rows then it will take 5. tail() function return last 5 rows that is given below.....

```
In [13]: std.tail()
```

Out[13]:

	roll number	registraction no	age	doc_id no	per_marks	attendence
jodu	6	5	20	5	35	67
madhu	8	7	23	6	90	89
laxyam	7	12	27	2	81	75
rohit	9	23	32	67	77	78
rimpa	13	14	25	34	79	82

9). Display first and last n values of a column in a dataframe:

we know head(n) and tail(n) function returns the first and last n rows of a data frame respectively .Now we will print first and last n rows corresponding to a particular column . taht is given below....

- determine first 4 values of "age" columns of std dataframe.

```
In [7]: std['age'].head(4)
```

```
Out[7]: ram      22
shyam    21
jodu     20
madhu    23
Name: age, dtype: int64
```

- Determine last 3 values of "attendence" column.

```
In [8]: std["attendence"].tail()
```

```
Out[8]: jodu      67
        madhu     89
        laxyam    75
        rohit     78
        rimpa     82
        Name: attendence, dtype: int64
```

- Note that: if we do not define n in head() or tail function by deafault it takes n=5.Let us see the two examples below------

```
In [11]: std['age'].head()
```

```
Out[11]: ram       22
         shyam     21
         jodu      20
         madhu     23
         laxyam    27
         Name: age, dtype: int64
```

```
In [48]: std['age'].tail()
```

```
Out[48]: jodu      20
         madhu     23
         laxyam    27
         rohit     32
         rimpa     25
         Name: age, dtype: int64
```

10). slection rows and columns by at(),loc(),iloc() function:

- Dataframe.loc: Access group of values of rows and columns of a series or dataframe.
- DataFrame.xs: turns a cross-section (row(s) or column(s)) from the Series/DataFrame.
- DataFrame.at: ccess a single value for a row/column label pair.
- DataFrame.iloc: “iloc” in pandas is used to select rows and columns by number, in the order that they appear in the data frame.

a). Dataframe.loc[ ] function: Selecting rows by label

In [76]:
```
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    "per_marks":[60,70,35,90,81,77,79],
    "attendence":[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
print(std)
```

	roll number	registraction no	age	doc_id no	per_marks	attendence
ram	2	1	22	12	60	45
shyam	4	3	21	3	70	78
jodu	6	5	20	5	35	67
madhu	8	7	23	6	90	89
laxyam	7	12	27	2	81	75
rohit	9	23	32	67	77	78
rimpa	13	14	25	34	79	82

- selection of a paricular column 'roll number'

In [65]:
```
z=std.loc[:,'roll number'] #selection of particular column. we can not use th
e function std.loc[:,0] for these selection

print(" roll number column is:\n",z,"\n \n")
```

```
 roll number column is:
 ram        2
shyam      4
jodu       6
madhu      8
laxyam     7
rohit      9
rimpa     13
Name: roll number, dtype: int64
```

- Selection of multiple column simultaneously :selection of the columns 'roll number','age','attendence' of the dataframe student?

In [71]:
```
t=std.loc[:,['roll number','attendence','age']] #multiple columns #selection of multiple columns
print("roll no. and attendence columns are:\n",t,"\n \n \n")
```

```
roll no. and attendence columns are:
         roll number  attendence  age
ram                2          45   22
shyam              4          78   21
jodu               6          67   20
madhu              8          89   23
laxyam             7          75   27
rohit              9          78   32
rimpa             13          82   25
```

- Selection of a particular row "madhu"

In [72]:
```
# find all the details of the student madhu
print(std.loc["madhu"]) # selection of particular row.
print("\n \n \n")
```

```
roll number          8
registraction no     7
age                 23
doc_id no            6
per_marks           90
attendence          89
Name: madhu, dtype: int64
```

- selection multiple rows simnultaneously

In [92]:
```
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    "per_marks":[60,70,35,90,81,77,79],
    "attendence":[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
#select both the rows 'madhu','jodu'
y=std.loc[["madhu","jodu"]]
print("these two rows are:\n",y,"\n \n")
```

```
these two rows are:
        roll number  registraction no  age  doc_id no  per_marks  attendence
madhu            8                 7   23          6         90          89
jodu             6                 5   20          5         35          67
```

- selection of combination of rows and columns:
  Find the roll number and registraction number of two student ram and shyam?

In [99]:
```
s=std.loc[['ram','jodu'],['roll number','registraction no']] # selection of
 rows and columns if we take [0,2] instead of
                                                              #['ram','jodu']
then we get an error
print(s,"\n \n \n ")
```

```
      roll number  registraction no
ram             2                 1
jodu            6                 5
```

In [36]:
```
import pandas as pd
d = {'roll number' :[2,4,6,8],
   'registraction no' :[1,3,5,7]}
student = pd.DataFrame(d,index=["ram","shyam","jodu","madhu"])
#print(student)

student.loc[:,'roll number'] #selection of particular column.
#student.loc[[0,1],['roll number']]
student.loc["ram"] # selection of particular row.
#student.loc['roll number']
student.loc[:,['roll number','registraction no']] #multiple columns
student.loc[['ram','jodu'],['roll number','registraction no']] # selection o
f rows and columns if we take [0,2] instead of
                                                              #['ram','jodu']
then we get an error
```

Out[36]:

	roll number	registraction no
ram	2	1
jodu	6	5

c). .at function :

Function notation : $naneofdataframe.at[" rowlables ", " columnlables "]$. If you sure about lebels the you apply iat function

Example: find roll number of ram of std dataframe?

In [14]:
```
#creating a data frame std
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    "per_marks":[60,70,35,90,81,77,79],
    "attendence":[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
#what is the roll number of the student ram?
std.at["ram","roll number"]
```

Out[14]: 2

d). iloc() function:

“iloc” in pandas is used to select rows and columns by number, in the order that they appear in the data frame.

- select a single row

  find the details of the student ram? or print first row?

  we know the index of the first row is 0 . so we will use loc[0]

In [114]:
```
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    "per_marks":[60,70,35,90,81,77,79],
    "attendence":[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
print ("here the data frame is :\n",std,"\n")
#select first row  or find the details of the student ram
a=std.iloc[0]
print("first row of std data frame is \n:",a,"\n \n \n")
```

```
here the data frame is :
         roll number  registraction no  age  doc_id no  per_marks  attendenc
e
ram                2                 1   22         12         60          45
shyam              4                 3   21          3         70          78
jodu               6                 5   20          5         35          67
madhu              8                 7   23          6         90          89
laxyam             7                12   27          2         81          75
rohit              9                23   32         67         77          78
rimpa             13                14   25         34         79          82

first row of std data frame is
: roll number          2
registraction no     1
age                 22
doc_id no           12
per_marks           60
attendence          45
Name: ram, dtype: int64
```

- Select multiple rows simultanously:
  print 2nd ,4th, first rows or find details of the student shyam madhu ram?

In [106]:
```
b=std.iloc[[1,3,0]]
print(b,"\n \n \n ")
```

```
       roll number  registraction no  age  doc_id no  per_marks  attendence
shyam            4                 3   21          3         70          78
madhu            8                 7   23          6         90          89
ram              2                 1   22         12         60          45
```

In [123]:
```
std.iloc[0:4] # first four rows of dataframe std
```

Out[123]:

	roll number	registraction no	age	doc_id no	per_marks	attendence
ram	2	1	22	12	60	45
shyam	4	3	21	3	70	78
jodu	6	5	20	5	35	67
madhu	8	7	23	6	90	89

- Select a particular column
  print 3rd column or find the age of all student?

In [108]:
```
print(std.iloc[:,2],"\n \n \n")
```

```
ram       22
shyam     21
jodu      20
madhu     23
laxyam    27
rohit     32
rimpa     25
Name: age, dtype: int64
```

- Select multiple columns simultaneously:
  print 3rd,5th,1st column of std data frame?

In [110]:
```
print(std.iloc[:,[2,4,0]],"\n \n \n")
```

```
        age  per_marks  roll number
ram      22         60            2
shyam    21         70            4
jodu     20         35            6
madhu    23         90            8
laxyam   27         81            7
rohit    32         77            9
rimpa    25         79           13
```

In [121]:
```
std.iloc[:, 0:2] # print first two columns of data frame with all rows
```

Out[121]:

	roll number	registraction no
ram	2	1
shyam	4	3
jodu	6	5
madhu	8	7
laxyam	7	12
rohit	9	23
rimpa	13	14

In [ ]:
```

```

- Select Multiple columns and rows can be selected together using the .iloc indexer:

  example 1: print combination of 2nd,4th rows and 3rd, 5th column of std dataframe?

In [111]:
```
z=std.iloc[[1,3],[2,4]]
print(z,"\n \n \n")
```

```
       age  per_marks
shyam   21         70
madhu   23         90
```

Example 2: select first 4 rows and 2nd,3rd, 4thcolumns of data frame std?

In [118]:
```
s=std.iloc[0:3,3:6]
print(s)
```

```
       doc_id no  per_marks  attendence
ram           12         60          45
shyam          3         70          78
jodu           5         35          67
```

In [117]:
```
std.shape
```

Out[117]:
```
(7, 6)
```

- Summary
  There are two “arguments” to iloc – a row selector, and a column selector. For example:

Single selections using iloc and DataFrame # Rows: data.iloc[0] # first row of data frame (Aleshia Tomkiewicz) - Note a Series data type output. data.iloc[1] # second row of data frame (Evan Zigomalas) data.iloc[-1] # last row of data frame (Mi Richan) # Columns: data.iloc[:,0] # first column of data frame (first_name) data.iloc[:,1] # second column of data frame (last_name) data.iloc[:,-1] # last column of data frame (id)

Multiple columns and rows can be selected together using the .iloc indexer.

In [ ]:
```
# Multiple row and column selections using iloc and DataFrame
data.iloc[0:5] # first five rows of dataframe
data.iloc[:, 0:2] # first two columns of data frame with all rows
data.iloc[[0,3,6,24], [0,5,6]] # 1st, 4th, 7th, 25th row + 1st 6th 7th colum
ns.
data.iloc[0:5, 5:8] # first 5 rows and 5th, 6th, 7th columns of data frame
 (county -> phone1).
```

Here, data= name of the data frame

Reference: https://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.loc.html (https://pandas.pydata.org/pandas-docs/version/0.23/generated/pandas.DataFrame.loc.html) https://www.shanelynn.ie/select-pandas-dataframe-rows-and-columns-using-iloc-loc-and-ix/ (https://www.shanelynn.ie/select-pandas-dataframe-rows-and-columns-using-iloc-loc-and-ix/) https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.loc.html (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.loc.html)

11). view datatypes of a dataframe

We view the types of dataframe according to row by using $name\ of\ dataframe.dtypes$ function.Let see the example below. here, we will create a dataframe df2 then find whats are the data types of every row.

In [32]:
```
d2 = {'name': ['Jimmy', 'Monty'],
      'score': [10.5, 9],
      'employed': [False, True],
      'kids': [0, 0]}
df2 = pd.DataFrame(data=d2)
df2
```

Out[32]:

	name	score	employed	kids
0	Jimmy	10.5	False	0
1	Monty	9.0	True	0

In [36]:
```
#to see the datatupes of every row
df2.dtypes
```

Out[36]:
```
name         object
score       float64
employed       bool
kids          int64
dtype: object
```

In [ ]:

Assigning Values

Once you understand how to access the various elements that make up a DataFrame, just follow the same logic to add or change the values in it.

For example, you have already seen that within the DataFrame structure an array of indexes is specified by the index attribute, and the row containing the name of the columns is specified with the columns attribute. Well, you can also assign a label, using the name attribute, to these two substructures for identifying them. That is given below...

In [2]:
```
# create a data frame std
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    "per_marks":[60,70,35,90,81,77,79],
    "attendence":[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
print(std)
```

```
        roll number  registraction no  age  doc_id no  per_marks  attendence
ram               2                 1   22          12         60          45
shyam             4                 3   21           3         70          78
jodu              6                 5   20           5         35          67
madhu             8                 7   23           6         90          89
laxyam            7                12   27           2         81          75
rohit             9                23   32          67         77          78
rimpa            13                14   25          34         79          82
```

- assine to label "student name" and "details" corresponding to the rows and columns?

In [3]:
```
std.index.name = 'student name' # add lebel of rows
std.columns.name = 'details' #add lebel of column
print(std)
```

```
details       roll number  registraction no  age  doc_id no  per_marks  \
student name
ram                     2                 1   22          12         60
shyam                   4                 3   21           3         70
jodu                    6                 5   20           5         35
madhu                   8                 7   23           6         90
laxyam                  7                12   27           2         81
rohit                   9                23   32          67         77
rimpa                  13                14   25          34         79

details       attendence
student name
ram                   45
shyam                 78
jodu                  67
madhu                 89
laxyam                75
rohit                 78
rimpa                 82
```

).How to change a single value of a dataframe?

simply select the item and give it the new value. these can be done by these ways. describes below-

- at method: we apply these method when we know lables of rows and column.
  suppose we scrutinize the data frame and find that roll number of the student 'jodu' is 10. How I will correct it?

In [105]:
```
#change roll number ram to 10
std.at["ram","roll number"]=10
print(std,"\n \n")
```

```
        roll number  registraction no  age  doc_id no  per_marks  attendence
ram              10                 1   22         12         60          45
shyam             4                56   26          3         70          78
jodu              6                 5   20          5         35          67
madhu             8                 7   23          6         90          89
laxyam            7                12   27          2         81          75
rohit             9                23   32         67         77          78
rimpa            13                14   25         34         79          82
```

- iat method: We use these method when we know the index of row and column.
  suppose we scrutinize the data frame and find that age number of the student 'shyam' is 26. How I will correct it?

In [104]:
```
std.iat[1,2]=26
std
```

Out[104]:

	roll number	registraction no	age	doc_id no	per_marks	attendence
ram	10	1	22	12	60	45
shyam	4	56	26	3	70	78
jodu	6	5	20	5	35	67
madhu	8	7	23	6	90	89
laxyam	7	12	27	2	81	75
rohit	9	23	32	67	77	78
rimpa	13	14	25	34	79	82

- Alternative method using the comment $name\ of\ dataframe\backslash['column\ lables']['row\ index'] = c.$
  These comments returns the dataframe whose values of the ['column\ lables']['row\ index'] chage to c. Let see the example below.
  Suppose we chhek these dataframe and find that attendence of rohit is 50 .How to update these value?

In [6]:
```
std['attendence']['rohit']=50
std
```

Out[6]:

details	roll number	registraction no	age	doc_id no	per_marks	attendence
student name						
ram	2	1	22	12	60	45
shyam	4	3	21	3	70	78
jodu	6	5	20	5	35	67
madhu	8	7	23	6	90	89
laxyam	7	12	27	2	81	75
rohit	9	23	89	67	77	50
rimpa	13	14	25	34	79	82

- Change multiple values of a dataframe
  Creating a dataframe dt and replacing elements 1000 and 2000 by 110 and 210 respectively of these dataframe?

In [31]:
```
import pandas as pd
dt = pd.DataFrame({'one':[10,20,30,40,50,2000], 'two':[1000,0,30,40,50,60]})
print("our given dataframe is:\n",dt)
print("the resulting data frame after replacing the elements is:")
dt.replace({1000:110,2000:210})
```

```
our given dataframe is:
     one   two
0    10  1000
1    20     0
2    30    30
3    40    40
4    50    50
5  2000    60
the resulting data frame after replacing the elements is:
```

Out[31]:

	one	two
0	10	110
1	20	0
2	30	30
3	40	40
4	50	50
5	210	60

a).Column Addition

Additional columns can be added after defining a DataFrame as below,

In [95]:
```
#create a dataframe student
import pandas as pd
d = {'roll number' :[2,4,6,8],
   'registraction no' :[1,3,5,7]}
student = pd.DataFrame(d,index=["ram","shyam","jodu","madhu"])
print(student)
```

```
       roll number  registraction no
ram              2                 1
shyam            4                 3
jodu             6                 5
madhu            8                 7
```

- add a column 'number of columns issues by the students"?

In [37]:
```
import pandas as pd
d = {'roll number' :[2,4,6,8],
   'registraction no' :[1,3,5,7]}
student = pd.DataFrame(d,index=["ram","shyam","jodu","madhu"])
student["number of columns issues by the students"]=[6,9,7,12]
print(student)
```

```
       roll number  registraction no  number of columns issues by the studen
ts
ram              2                 1
6
shyam            4                 3
9
jodu             6                 5
7
madhu            8                 7
12
```

suppose we want to add a new column called ‘new’ with the value within 12 replicated for each of its elements. then we will use the following function given below

In [96]:
```
import pandas as pd
d = {'roll number' :[2,4,6,8],
   'registraction no' :[1,3,5,7]}
student = pd.DataFrame(d,index=["ram","shyam","jodu","madhu"])
student["new"]=12
print(student)
```

```
       roll number  registraction no  new
ram              2                 1   12
shyam            4                 3   12
jodu             6                 5   12
madhu            8                 7   12
```

In [ ]:
```

```

- add a column 'question_id' that is the sum of the columns 'roll number' and 'registraction number' ?

In [98]:
```
import pandas as pd
d = {'roll number' :[2,4,6,8],
   'registraction no' :[1,3,5,7]}
student = pd.DataFrame(d,index=["ram","shyam","jodu","madhu"])
student["question_id"]=student["roll number"]+student["registraction no"]
print(student)
```

```
       roll number  registraction no  question_id
ram              2                 1            3
shyam            4                 3            7
jodu             6                 5           11
madhu            8                 7           15
```

Update a column of a dataframe

It is possible to update a column after creating a dataframe. This process is describe below...

In [1]:
```
# creating a data frame student
import pandas as pd
d = {'roll number' :[2,4,6,8],
   'registraction no' :[1,3,5,7]}
student = pd.DataFrame(d,index=["ram","shyam","jodu","madhu"])
student["question_id"]=student["roll number"]+student["registraction no"]
print(student)
```

```
       roll number  registraction no  question_id
ram              2                 1            3
shyam            4                 3            7
jodu             6                 5           11
madhu            8                 7           15
```

In [121]:
```
#update the column "registraction no "

c=[5,8,9,3]
student["registraction no "]=c
print(student)
##########??????????????????????????????????????wrong output
```

```
       roll number  registraction no  question_id  registraction no
ram              2                 1            3                NaN
shyam            4                 3            7                NaN
jodu             6                 5           11                NaN
madhu            8                 7           15                NaN
```

b). Rename of rows and columns or Change column names and row indexes in Pandas DataFrame:

Pandas Dataframe type has two attributes called ‘columns’ and ‘index’ which can be used to change the column names as well as the row indexes.

i).Rename a single column:

Using rename() function with dictionary to change a single column

In [39]:
```
# create a data frame std
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    "per_marks":[60,70,35,90,81,77,79],
    "attendence":[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
print("our given dataframe is:\n",std,"\n \n ")
# from "Age" to "column 3" using rename() function
z=std.rename(columns = {"age":"column 3"})
print("new dataframe after rename column is:\n",z)
```

```
our given dataframe is:
         roll number  registraction no  age  doc_id no  per_marks  attendenc
e
ram              2                 1   22         12         60          45
shyam            4                 3   21          3         70          78
jodu             6                 5   20          5         35          67
madhu            8                 7   23          6         90          89
laxyam           7                12   27          2         81          75
rohit            9                23   32         67         77          78
rimpa           13                14   25         34         79          82


new dataframe after rename column is:
         roll number  registraction no  column 3  doc_id no  per_marks  \
ram              2                 1        22         12         60
shyam            4                 3        21          3         70
jodu             6                 5        20          5         35
madhu            8                 7        23          6         90
laxyam           7                12        27          2         81
rohit            9                23        32         67         77
rimpa           13                14        25         34         79

        attendence
ram             45
shyam           78
jodu            67
madhu           89
laxyam          75
rohit           78
rimpa           82
```

note that if we add an extra comment "inplace=True" in rename function then the dataframe wiil be changed permanently. that means suppose we run the comment $std.\,rename(columns = "\ age\ ":"\ column3\ "); print(std)$ then we will get the previous dataframa as a output. If we will run the comment $std.\,rename(columns = "\ age\ ":"\ column3\ ", inplace = True); print(std)$ then we will get the out put as a changed column . One most advantage of add an extra comment $inplace = True$ is it does not necessary to run an extra comment $z = std.\,rename(columns = "\ age\ ":"\ column3\ "); print(z)$. Let see the example below-

In [40]:
```
# create a data frame std
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    "per_marks":[60,70,35,90,81,77,79],
    "attendence":[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
# from "Age" to "column 3" using rename() function
z=std.rename(columns = {"age":"column 3"})
print("new dataframe after rename column is:\n",z)
```

```
new dataframe after rename column is:
        roll number  registraction no  column 3  doc_id no  per_marks  \
ram               2                 1        22         12         60
shyam             4                 3        21          3         70
jodu              6                 5        20          5         35
madhu             8                 7        23          6         90
laxyam            7                12        27          2         81
rohit             9                23        32         67         77
rimpa            13                14        25         34         79

        attendence
ram             45
shyam           78
jodu            67
madhu           89
laxyam          75
rohit           78
rimpa           82
```

We also get same output by using these comment below

In [41]:
```
std.rename(columns = {"age":"column 3"},inplace=True)
print(std)
```

```
        roll number  registraction no  column 3  doc_id no  per_marks  \
ram               2                 1        22         12         60
shyam             4                 3        21          3         70
jodu              6                 5        20          5         35
madhu             8                 7        23          6         90
laxyam            7                12        27          2         81
rohit             9                23        32         67         77
rimpa            13                14        25         34         79

        attendence
ram             45
shyam           78
jodu            67
madhu           89
laxyam          75
rohit           78
rimpa           82
```

- Rename of colum by column index: Using values attribute to rename the columns.We can use values attribute directly on the column whose name we want to change. Let see the example below: rename of column 5 "per_mark" by "percentage of marks".

In [42]:
```
# create a data frame std
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    "per_marks":[60,70,35,90,81,77,79],
    "attendence":[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
std.columns.values[4] = 'Percentages of marks'
print(std)
```

```
        roll number  registraction no  column 3  doc_id no  \
ram               2                 1        22         12
shyam             4                 3        21          3
jodu              6                 5        20          5
madhu             8                 7        23          6
laxyam            7                12        27          2
rohit             9                23        32         67
rimpa            13                14        25         34

        Percentages of marks  attendence
ram                       60          45
shyam                     70          78
jodu                      35          67
madhu                     90          89
laxyam                    81          75
rohit                     77          78
rimpa                     79          82
```

ii). Change multiple column names simultaneously –

We can change multiple column names by passing a dictionary of old names and new names, to the rename() function. Let see the example below
Change the name first and sceond column by "first column" and "sceond column" of std dataframe?

In [55]:
```
# create a data frame std
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    "per_marks":[60,70,35,90,81,77,79],
    "attendence":[45,78,67,89,75,78,82]}
std.rename({"roll number":"first column","registraction no":"sceond column"
}, axis='columns')
```

Out[55]:

	first column	sceond column	age	doc_id no	per_marks	attendence
ram	2	1	22	12	60	45
shyam	4	3	21	3	70	78
jodu	6	5	20	5	35	67
madhu	8	7	23	6	90	89
laxyam	7	12	27	2	81	75
rohit	9	23	32	67	77	78
rimpa	13	14	25	34	79	82

If one can run the comment
$std.rename(" rollnumber ":" firstcolumn ","registractionno ":" sceondcolumn ", axis = 1)$
then he also get same output

iii). Rename of a single row:

For changing rows we use the previous comment same as columns but we change the comment axis =columns, instead of axis =0. Let see the example below
Rename the thid row 'jodu' by 'suman'?

In [48]:
```
# create a data frame std
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    "per_marks":[60,70,35,90,81,77,79],
    "attendence":[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
r=std.rename({"jodu":"suman"}, axis=0)
print("new dataframe after rename row is:\n")
r
```

```
new dataframe after rename row is:
```

Out[48]:

	roll number	registraction no	age	doc_id no	per_marks	attendence
ram	2	1	22	12	60	45
shyam	4	3	21	3	70	78
suman	6	5	20	5	35	67
madhu	8	7	23	6	90	89
laxyam	7	12	27	2	81	75
rohit	9	23	32	67	77	78
rimpa	13	14	25	34	79	82

- row change by row index:
  Change 4th row by "debanjon"?

In [56]:
```
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    "per_marks":[60,70,35,90,81,77,79],
    "attendence":[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
std.index.values[4] = 'debanjon'
std
```

Out[56]:

	roll number	registraction no	age	doc_id no	per_marks	attendence
ram	2	1	22	12	60	45
shyam	4	3	21	3	70	78
jodu	6	5	20	5	35	67
madhu	8	7	23	6	90	89
debanjon	7	12	27	2	81	75
rohit	9	23	32	67	77	78
rimpa	13	14	25	34	79	82

iv).Rename of multiple rows simultineously:

In [8]:
```
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    "per_marks":[60,70,35,90,81,77,79],
    "attendence":[45,78,67,89,75,78,82]}
# rename the first and last rows by"first_row" and "last_row"
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
std.rename({"ram":"first_row","rimpa":"last_row"}, axis=0)
```

Out[8]:

	roll number	registraction no	age	doc_id no	per_marks	attendence
first_row	2	1	22	12	60	45
shyam	4	3	21	3	70	78
jodu	6	5	20	5	35	67
madhu	8	7	23	6	90	89
laxyam	7	12	27	2	81	75
rohit	9	23	32	67	77	78
last_row	13	14	25	34	79	82

c).Delecting multiple(or a particular) rows and columns by drop() function:

We can use Pandas drop() function to drop multiple columns from a dataframe. Pandas drop() is versatile and it can be used to drop rows of a dataframe as well columns. In addition, we also need to specify axis=1 argument to tell the drop() function that we are dropping columns
With axis=0 drop() function drops rows of a dataframe

i).Delecting a particular column dataframe:

For delecting or droping a particular row of a
dataframe we wiil use the function $drop('select\ column, axis = 1)$
xample 1: delecting the column 'registraction number' of std dataframe?

In [23]:
```
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    "per_marks":[60,70,35,90,81,77,79],
    "attendence":[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
print("our data frame is \n",std,"\n")
# delecting the column"registraction number column of std data frame
new_frame=std.drop("registraction no",axis=1) # if we add ,inplace=True then
we can get always change dataframe
print("new data frame after delecting the column'registraction number  is:\n"
,new_frame,"\n \n \n")
```

```
our data frame is
         roll number  registraction no  age  doc_id no  per_marks  attendenc
e
ram              2                 1   22         12         60          45
shyam            4                 3   21          3         70          78
jodu             6                 5   20          5         35          67
madhu            8                 7   23          6         90          89
laxyam           7                12   27          2         81          75
rohit            9                23   32         67         77          78
rimpa           13                14   25         34         79          82

new data frame after delecting the column'registraction number  is:
         roll number  age  doc_id no  per_marks  attendence
ram              2   22         12         60          45
shyam            4   21          3         70          78
jodu             6   20          5         35          67
madhu            8   23          6         90          89
laxyam           7   27          2         81          75
rohit            9   32         67         77          78
rimpa           13   25         34         79          82
```

ii).Delecting a particular row of a dataframe: For delecting or droping a particular row of a

dataframe we wiil use the function $drop('select\ row' axis = 0)$
Example : delecting the row 'jodu' of std dataframe?

In [26]:
```
# create a data frame std
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    'per_marks':[60,70,35,90,81,77,79],
    'attendence':[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
#delecting row
new1=std.drop('jodu',axis=0)
print("new data frame after delcting the row 'jodu' is:\n",new1,"\n \n \n ")
```

```
new data frame after delcting the row 'jodu' is:
         roll number  registraction no  age  doc_id no  per_marks  attendenc
e
ram              2                 1   22         12         60          45
shyam            4                 3   21          3         70          78
madhu            8                 7   23          6         90          89
laxyam           7                12   27          2         81          75
rohit            9                23   32         67         77          78
rimpa           13                14   25         34         79          82
```

iii).Delete or drop multiple columns of a dataframe:To drop multiple column of a dataframe

it is neccessary to write the multiples columns in a list.Then we will use drop() function as previous. Function notation $drop([multiplecolumns], axis = 1)$.
Example:Delete the columns 'attendence' and 'age' of student dataframe?

In [27]:
```
#creating dataframe
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    'per_marks':[60,70,35,90,81,77,79],
    'attendence':[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
#delecting multiple columns 'attendence' and'age'
new2=std.drop(['attendence','age'],axis=1)
print("new data frame after delcting columns 'attendence' and 'age' is:\n",ne
w2,"\n \n \n ")
```

```
new data frame after delcting columns 'attendence' and 'age' is:
         roll number  registraction no  doc_id no  per_marks
ram                2                 1         12         60
shyam              4                 3          3         70
jodu               6                 5          5         35
madhu              8                 7          6         90
laxyam             7                12          2         81
rohit              9                23         67         77
rimpa             13                14         34         79
```

iv).Delete or drop multiple rows of a dataframe:To drop multiple rows of a dataframe

it is neccessary to write the multiples rows in a list.Then we will use drop() function as previous. Function notation $drop([multiple\ rows], axis = 0)$.
Example:Delete the columns 'rimpa' and 'madhu' of student dataframe?

In [28]:
```
#creating dataframe
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    'per_marks':[60,70,35,90,81,77,79],
    'attendence':[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
#delecting multiple rows 'rimpa' and 'madhu'
new3=std.drop(['rimpa','madhu'],axis=0)
print("new data frame after delcting rows 'rimpa' and 'madhu' is:\n",new3,"\n
 \n \n ")
```

```
new data frame after delcting rows 'rimpa' and 'madhu' is:
         roll number  registraction no  age  doc_id no  per_marks  attendenc
e
ram              2                 1   22         12         60          45
shyam            4                 3   21          3         70          78
jodu             6                 5   20          5         35          67
laxyam           7                12   27          2         81          75
rohit            9                23   32         67         77          78
```

- How To Drop Multiple Columns inplace in Pandas?
  We can also use Pandas drop() function to drop multiple columns or rows or a particular column or rows in place. This basically changes the original dataframe. To drop columns without creating a new dataframe we specify “inplace=True”. Let see the example below Example:

In [29]:
```
#creating dataframe
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    'per_marks':[60,70,35,90,81,77,79],
    'attendence':[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
#delecting multiple columns 'attendence' and'age' in place
std.drop(['attendence','age'],axis=1,inplace=True)
print(std)
```

```
        roll number  registraction no  doc_id no  per_marks
ram               2                 1         12         60
shyam             4                 3          3         70
jodu              6                 5          5         35
madhu             8                 7          6         90
laxyam            7                12          2         81
rohit             9                23         67         77
rimpa            13                14         34         79
```

Suppose we donot use "inplase=True" then the dataframe is not change.Let see the example below

In [30]:
```
#creating dataframe
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    'per_marks':[60,70,35,90,81,77,79],
    'attendence':[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
#delecting multiple columns 'attendence' and'age' in place
std.drop(['attendence','age'],axis=1)
print(std)
```

	roll number	registraction no	age	doc_id no	per_marks	attendence
ram	2	1	22	12	60	45
shyam	4	3	21	3	70	78
jodu	6	5	20	5	35	67
madhu	8	7	23	6	90	89
laxyam	7	12	27	2	81	75
rohit	9	23	32	67	77	78
rimpa	13	14	25	34	79	82

- alternatively, we can also delete a column by del coomend. let see the example below...

In [32]:
```
# create a data frame std
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    "per_marks":[60,70,35,90,81,77,79],
    "attendence":[45,78,67,89,75,78,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
print("our given dataframe is",std,"\n \n ")
# delete the column'Registraction number'
del std["registraction no"]
print("new data frame after delecting the column 'registrcation number' is:\n
",std)
```

```
our given dataframe is         roll number  registraction no  age  doc_id no
per_marks  attendence
ram                2                1   22         12         60          45
shyam              4                3   21          3         70          78
jodu               6                5   20          5         35          67
madhu              8                7   23          6         90          89
laxyam             7               12   27          2         81          75
rohit              9               23   32         67         77          78
rimpa             13               14   25         34         79          82


new data frame after delecting the column 'registrcation number' is:
         roll number  age  doc_id no  per_marks  attendence
ram                2   22         12         60          45
shyam              4   21          3         70          78
jodu               6   20          5         35          67
madhu              8   23          6         90          89
laxyam             7   27          2         81          75
rohit              9   32         67         77          78
rimpa             13   25         34         79          82
```

d). delete duplicate rows of a dataframe:

we drop duplicate rows by using $name\ of\ dataframe\.drop_duplicates()$ function.Let see the example below..

In [78]:
```
#create a dataframe dt
import pandas as pd
d1 = {'a' :[2,4,6,8,7,4,2],
   'b' :[1,3,5,7,12,3,1],
   'dc':[12,3,5,6,2,3,12],
    "d":[60,70,35,90,81,70,60],
    "e":[45,78,67,89,75,78,45]}
dt = pd.DataFrame(d1,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
print("our given dataframe is\n",dt,"\n \n ")
#delete duplicates rows
print("resulting dataframe after delecting duplicates rows is:")
dt.drop_duplicates()
```

```
our given dataframe is
         a   b  dc   d   e
ram      2   1  12  60  45
shyam    4   3   3  70  78
jodu     6   5   5  35  67
madhu    8   7   6  90  89
laxyam   7  12   2  81  75
rohit    4   3   3  70  78
rimpa    2   1  12  60  45


resulting dataframe after delecting duplicates rows is:
```

Out[78]:

	a	b	dc	d	e
ram	2	1	12	60	45
shyam	4	3	3	70	78
jodu	6	5	5	35	67
madhu	8	7	6	90	89
laxyam	7	12	2	81	75

From these example we see that ram and rimpa rows are same & shyam and rohit rows are same. by these function we delete the duplicates rows rohit and rimpa from the datafram dt.

Data Wrangling:

Data wrangling is the process of processing data to prepare it for use in the next step. Examples of data wrangling processes include merging,sums, grouping, and concatenation. This kind of manipulation is often needed in data science to get your data in to a form that works well with whatever analysis or algorithms that you're going to put it through.

1).Merging

The Pandas library allows us to join DataFrame objects via the merge() function. Let us create two DataFrames and demonstrate how to merge them.

- Example 1:

In [2]:
```
import pandas as pd
d1 = {
    'subject_id': [1,2,3,4,5],
    'student_name': ['John', 'Emily', 'Kate', 'Joseph', 'Dennis']
}
df1=pd.DataFrame(d1)
print(df1)
```

```
   subject_id student_name
0           1         John
1           2        Emily
2           3         Kate
3           4       Joseph
4           5       Dennis
```

In [3]:
```
#create another data frame df2
import pandas as pd
d2 = {
    'subject_id': [4, 5, 6, 7, 8],
    'student_name': ['Brian', 'William', 'Lilian', 'Grace', 'Caleb']
}
df2=pd.DataFrame(d2)
print(df2)
```

```
   subject_id student_name
0           4        Brian
1           5      William
2           6       Lilian
3           7        Grace
4           8        Caleb
```

We now need to merge the two DataFrames, that is, df1 and df2 along the particular column values of subject_id. We simply call the merge() function as shown below:

In [11]:
```
import pandas as pd
m=pd.merge(df1, df2, on='subject_id')
print("merge dataframe is\n",m)
```

```
merge dataframe is
    subject_id student_name_x student_name_y
0           4         Joseph          Brian
1           5         Dennis        William
```

.here, we merge two dataframe df1 and df2 according to column'subject_id' . student _name_x is the name of student of the dataframe df1 and student _name_y is the name of student of the dataframe df2

- Example 2: create two dataframe std and std1 .then merge of these two dataframe acccording to column 'attendence'?

In [14]:
```
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    'per_marks':[60,70,35,90,81,77,79],
    'attendence':[12,78,79,89,75,20,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
print(std,"\n")
d1 = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    'per_marks':[60,70,35,90,81,77,79],
    'attendence':[45,78,67,89,75,78,82]}
std1 = pd.DataFrame(d1,index=["rohim","shyam","kartik","madhu","laxyam","roh
it","rimpa"])
print(std1)
```

```
       roll number  registraction no  age  doc_id no  per_marks  attendence
ram              2                 1   22          12         60          12
shyam            4                 3   21           3         70          78
jodu             6                 5   20           5         35          79
madhu            8                 7   23           6         90          89
laxyam           7                12   27           2         81          75
rohit            9                23   32          67         77          20
rimpa           13                14   25          34         79          82

       roll number  registraction no  age  doc_id no  per_marks  attendence
rohim            2                 1   22          12         60          45
shyam            4                 3   21           3         70          78
kartik           6                 5   20           5         35          67
madhu            8                 7   23           6         90          89
laxyam           7                12   27           2         81          75
rohit            9                23   32          67         77          78
rimpa           13                14   25          34         79          82
```

In [18]:
```
# merge of these two data frame
print(std1)
m2=pd.merge(std,std1,on='attendence')
```

```
        roll number  registraction no  age  doc_id no  per_marks  attendence
rohim             2                 1   22          12         60          45
shyam             4                 3   21           3         70          78
kartik            6                 5   20           5         35          67
madhu             8                 7   23           6         90          89
laxyam            7                12   27           2         81          75
rohit             9                23   32          67         77          78
rimpa            13                14   25          34         79          82
```

Out[18]:

	roll number_x	registraction no_x	age_x	doc_id no_x	per_marks_x	attendence	roll number_y	registraction no_y
0	4	3	21	3	70	78	4	3
1	4	3	21	3	70	78	9	23
2	8	7	23	6	90	89	8	7
3	7	12	27	2	81	75	7	12
4	13	14	25	34	79	82	13	14

2. adding two dataframe

In [23]:
```
import pandas as pd
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    'per_marks':[60,70,35,90,81,77,79],
    }
std = pd.DataFrame(d,index=["shyam","jodu","madhu","laxyam","rohit","rimpa",
"ram"])
print("first dataframe is:\n",std,"\n")
d1 = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    'per_marks':[60,70,35,90,81,77,79],
    'attendence':[45,78,67,89,75,78,82]}
std1 = pd.DataFrame(d1,index=["rohim","shyam","kartik","madhu","laxyam","roh
it","rimpa"])
print("seceond dataframe is:\n",std1,"\n")
print("sum of these two dataframe is:")
std+std1
```

```
first dataframe is:
         roll number  registraction no  age  doc_id no  per_marks
shyam              2                 1   22         12         60
jodu               4                 3   21          3         70
madhu              6                 5   20          5         35
laxyam             8                 7   23          6         90
rohit              7                12   27          2         81
rimpa              9                23   32         67         77
ram               13                14   25         34         79

seceond dataframe is:
         roll number  registraction no  age  doc_id no  per_marks  attendenc
e
rohim              2                 1   22         12         60          45
shyam              4                 3   21          3         70          78
kartik             6                 5   20          5         35          67
madhu              8                 7   23          6         90          89
laxyam             7                12   27          2         81          75
rohit              9                23   32         67         77          78
rimpa             13                14   25         34         79          82

sum of these two dataframe is:
```

Out[23]:

	age	attendence	doc_id no	per_marks	registraction no	roll number
jodu	NaN	NaN	NaN	NaN	NaN	NaN
kartik	NaN	NaN	NaN	NaN	NaN	NaN
laxyam	50.0	NaN	8.0	171.0	19.0	15.0
madhu	43.0	NaN	11.0	125.0	12.0	14.0
ram	NaN	NaN	NaN	NaN	NaN	NaN
rimpa	57.0	NaN	101.0	156.0	37.0	22.0
rohim	NaN	NaN	NaN	NaN	NaN	NaN
rohit	59.0	NaN	69.0	158.0	35.0	16.0
shyam	43.0	NaN	15.0	130.0	4.0	6.0

3).Grouping :'

Using the get_group() method, we can select a single group.

In [62]:
```
# import pandas library
import pandas as pd

raw = {
    'Name': ['John', 'John', 'Grace', 'Grace', 'Benjamin', 'Benjamin', 'Benj
amin',
        'Benjamin', 'John', 'Alex', 'Alex', 'Alex'],
    'Position': [2, 1, 1, 4, 2, 4, 3, 1, 3, 2, 4, 3],
    'Year': [2009, 2010, 2009, 2010, 2010, 2010, 2011, 2012, 2011, 2013, 201
3, 2012],
    'Marks':[408, 398, 422, 376, 401, 380, 396, 388, 356, 402, 368, 378]
}
df = pd.DataFrame(raw)

group = df.groupby('Year')
print(group.get_group(2010))
```

```
       Name  Position  Year  Marks
1      John         1  2010    398
3     Grace         4  2010    376
4  Benjamin         2  2010    401
5  Benjamin         4  2010    380
```

4).Concatenation:

Concatenation of data, which basically means to add one set of data to another, can be done by calling the concat() function.

- Example 1:

In [41]:
```
#create a dataframe df1
import pandas as pd
d1 = {
    'subject_id': [1,2,3,4,5],
    'student_name': ['John', 'Emily', 'Kate', 'Joseph', 'Dennis']
}
df1=pd.DataFrame(d1)
print(df1)
#create another dataframe df2

#create another data frame df2
import pandas as pd
d2 = {
    'subject_id': [4, 5, 6, 7, 8],
    'student_name': ['Brian', 'William', 'Lilian', 'Grace', 'Caleb']
}
df2=pd.DataFrame(d2)
print(df2)
```

```
   subject_id student_name
0           1         John
1           2        Emily
2           3         Kate
3           4       Joseph
4           5       Dennis
   subject_id student_name
0           4        Brian
1           5      William
2           6       Lilian
3           7        Grace
4           8        Caleb
```

In [42]:
```
print(pd.concat([df1, df2]))
```

```
   subject_id student_name
0           1         John
1           2        Emily
2           3         Kate
3           4       Joseph
4           5       Dennis
0           4        Brian
1           5      William
2           6       Lilian
3           7        Grace
4           8        Caleb
```

- Example 2:

In [46]:
```
d = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    'per_marks':[60,70,35,90,81,77,79],
     'attendence':[45,78,67,89,75,78,82]
    }
std = pd.DataFrame(d,index=["shyam","jodu","madhu","laxyam","rohit","rimpa",
"ram"])
print("first dataframe is:\n",std,"\n")
d1 = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    'per_marks':[60,70,35,90,81,77,79],
    'attendence':[45,78,67,89,75,78,82]}
std1 = pd.DataFrame(d1,index=["rohim","shyam","kartik","madhu","laxyam","roh
it","rimpa"])
print("seceond dataframe is:\n",std1,"\n")
print("sum of these two dataframe is:")
print(pd.concat([std, std1]))
```

```
first dataframe is:
         roll number  registraction no  age  doc_id no  per_marks  attendenc
e
shyam              2                 1   22         12         60          45
jodu               4                 3   21          3         70          78
madhu              6                 5   20          5         35          67
laxyam             8                 7   23          6         90          89
rohit              7                12   27          2         81          75
rimpa              9                23   32         67         77          78
ram               13                14   25         34         79          82

seceond dataframe is:
         roll number  registraction no  age  doc_id no  per_marks  attendenc
e
rohim              2                 1   22         12         60          45
shyam              4                 3   21          3         70          78
kartik             6                 5   20          5         35          67
madhu              8                 7   23          6         90          89
laxyam             7                12   27          2         81          75
rohit              9                23   32         67         77          78
rimpa             13                14   25         34         79          82

sum of these two dataframe is:
        roll number  registraction no  age  doc_id no  per_marks  attendence
shyam              2                 1   22         12         60          45
jodu               4                 3   21          3         70          78
madhu              6                 5   20          5         35          67
laxyam             8                 7   23          6         90          89
rohit              7                12   27          2         81          75
rimpa              9                23   32         67         77          78
ram               13                14   25         34         79          82
rohim              2                 1   22         12         60          45
shyam              4                 3   21          3         70          78
kartik             6                 5   20          5         35          67
madhu              8                 7   23          6         90          89
laxyam             7                12   27          2         81          75
rohit              9                23   32         67         77          78
rimpa             13                14   25         34         79          82
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

missing value observation

link:https://www.tutorialspoint.com/python_pandas/python_pandas_missing_data.htm (https://www.tutorialspoint.com/python_pandas/python_pandas_missing_data.htm)

Missing data is always a problem in real life scenarios. Areas like machine learning and data mining face severe issues in the accuracy of their model predictions because of poor quality of data caused by missing values. In these areas, missing value treatment is a major point of focus to make their models more accurate and valid.

- When and Why Is Data Missed?
  Let us consider an online survey for a product. Many a times, people do not share all the information related to them. Few people share their experience, but not how long they are using the product; few people share how long they are using the product, their experience but not their contact information. Thus, in some or the other way a part of data is always missing, and this is very common in real time.

Let us now see how we can handle missing values (say NA or NaN) using Pandas.

In [61]:
```
# create a data frame std
import pandas as pd
import numpy as np # for define a NaN values we import numpy function then we
will use np.NaN() function.
d = {"roll number" :[2,4,np.NaN,8,7,np.NaN,13],
   "registraction no" :[1,3,9,7,12,23,14],
    "age":[22,np.NaN,20,23,np.NaN,32,np.NaN],
    "doc_id no":[12,3,5,12,2,67,34],
    "per_marks":[60,np.NaN,35,90,np.NaN,77,79],
    "attendence":[np.NaN,78,np.NaN,89,75,np.NaN,82]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
print("our given dataframe is:\n ",std,"\n \n ")
```

```
our given dataframe is:
          roll number  registraction no   age  doc_id no  per_marks  attende
nce
ram             2.0                 1  22.0         12       60.0         Na
N
shyam           4.0                 3   NaN          3        NaN        78.
0
jodu            NaN                 9  20.0          5       35.0         Na
N
madhu           8.0                 7  23.0         12       90.0        89.
0
laxyam          7.0                12   NaN          2        NaN        75.
0
rohit           NaN                23  32.0         67       77.0         Na
N
rimpa          13.0                14   NaN         34       79.0        82.
0
```

In the output, NaN means Not a Number.

i).Cheek missing values observation

The $isnull()$ and $notnull()$ functions are very useful to identify the indexes without a value.these two functions return the Dataframe(also Series) with Boolean values that contains the 'True' and 'False' valuesdepending on whether the item is a NaN value or less. The isnull( ) function returns 'True' at NaN values inthe Series; inversely, the notnull( ) function returns 'True' if they are not NaN

- find the missingvalues of the age 'column' of std dataframe?

```
In [62]: std['age'].isnull()
```

```
Out[62]: ram       False
         shyam      True
         jodu      False
         madhu     False
         laxyam     True
         rohit     False
         rimpa      True
         Name: age, dtype: bool
```

Alternatively one can to know all the values rows and columns corresponding to missing values of a particular column "age" by using these function below..

```
In [63]: print(std[std['age'].isnull()])
```

```
         roll number  registraction no  age  doc_id no  per_marks  attendence
shyam            4.0                 3  NaN          3        NaN        78.0
laxyam           7.0                12  NaN          2        NaN        75.0
rimpa           13.0                14  NaN         34       79.0        82.0
```

*ii). cheek for non missing values observation or display the non missing values or numeric values of a column of a dataframe-**

```
In [64]: std['age'].notnull()
```

```
Out[64]: ram        True
         shyam     False
         jodu       True
         madhu      True
         laxyam    False
         rohit      True
         rimpa     False
         Name: age, dtype: bool
```

Alternatively,

In [37]:
```
print(std[std['age'].notnull()])  # it returns the all numeric values of 'ag
e' column of std dataframe.
```

```
       roll number  registraction no   age  doc_id no  per_marks  attendence
ram            2.0                 1  22.0         12       60.0         NaN
jodu           NaN                 9  20.0          5       35.0         NaN
madhu          8.0                 7  23.0         12       90.0        89.0
rohit          9.0                23  32.0         67       77.0         NaN
```

*).Calculations with Missing Data

- When summing data, NA will be treated as Zero
- If the data are all NA, then the result will be NA

In [39]:
```
# create a data frame std
import pandas as pd
import numpy as np # for define a NaN values we import numpy function then we
will use np.NaN() function.
d = {"roll number" :[2,4,np.NaN,8,7,np.NaN,13],
   "registraction no" :[1,3,9,7,12,23,14],
    "age":[22,np.NaN,20,23,np.NaN,32,np.NaN],
    "doc_id no":[12,3,5,12,2,67,34],
    "per_marks":[60,np.NaN,35,90,np.NaN,77,79],
    "attendence":[np.NaN,np.NaN,np.NaN,np.NaN,np.NaN,np.NaN,np.NaN]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
print("our given dataframe is:\n ",std,"\n \n ")
```

```
our given dataframe is:
          roll number  registraction no   age  doc_id no  per_marks  attende
nce
ram             2.0                 1  22.0         12       60.0         Na
N
shyam           4.0                 3   NaN          3        NaN         Na
N
jodu            NaN                 9  20.0          5       35.0         Na
N
madhu           8.0                 7  23.0         12       90.0         Na
N
laxyam          7.0                12   NaN          2        NaN         Na
N
rohit           NaN                23  32.0         67       77.0         Na
N
rimpa          13.0                14   NaN         34       79.0         Na
N
```

- Example 1: sum of age column of std dataframe

```
In [40]: std['age'].sum()
```

```
Out[40]: 97.0
```

- Example 2:sum of attendence column of std dataframe

```
In [41]: std['attendence'].sum()
```

```
Out[41]: 0.0
```

Here,all the values of atendence column are NaN .We know NaN values are treated as 0. So sum of all NaN values is 0.

c).Cleaning / Filling Missing Data

Pandas provides various methods for cleaning the missing values. The fillna function can “fill in” NA values with non-null data in a couple of ways, which we have illustrated in the following sections.

https://www.journaldev.com/29055/python-pandas-module-tutorial (https://www.journaldev.com/29055/python-pandas-module-tutorial)

i). Replace missing value by a scaler .

Let see the following examples below...

- Example 1 : fill all the missing value by a scaler 0?

In [15]:
```
# create a data frame std
import pandas as pd
import numpy as np # for define a NaN values we import numpy function then we
will use np.NaN() function.
d = {"roll number" :[2,4,np.NaN,8,7,np.NaN,13],
   "registraction no" :[1,3,9,7,12,23,14],
    "age":[22,np.NaN,20,23,np.NaN,32,np.NaN],
    "doc_id no":[12,3,5,12,2,67,34],
    "per_marks":[60,np.NaN,35,90,np.NaN,77,79],
    "attendence":[np.NaN,np.NaN,np.NaN,np.NaN,np.NaN,np.NaN,np.NaN]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
print("our given dataframe is:\n ",std,"\n \n ")
print ("NaN replaced with '0':")
std.fillna(0)
```

```
our given dataframe is:
          roll number  registraction no   age  doc_id no  per_marks  attende
nce
ram             2.0                 1  22.0         12       60.0         Na
N
shyam           4.0                 3   NaN          3        NaN         Na
N
jodu            NaN                 9  20.0          5       35.0         Na
N
madhu           8.0                 7  23.0         12       90.0         Na
N
laxyam          7.0                12   NaN          2        NaN         Na
N
rohit           NaN                23  32.0         67       77.0         Na
N
rimpa          13.0                14   NaN         34       79.0         Na
N


NaN replaced with '0':
```

Out[15]:

	roll number	registraction no	age	doc_id no	per_marks	attendence
ram	2.0	1	22.0	12	60.0	0.0
shyam	4.0	3	0.0	3	0.0	0.0
jodu	0.0	9	20.0	5	35.0	0.0
madhu	8.0	7	23.0	12	90.0	0.0
laxyam	7.0	12	0.0	2	0.0	0.0
rohit	0.0	23	32.0	67	77.0	0.0
rimpa	13.0	14	0.0	34	79.0	0.0

- Example 2 : fill all the missing value by a scaler 55?

In [16]:
```
#replacing all the missing value by 55
std.fillna(55)
```

Out[16]:

	roll number	registraction no	age	doc_id no	per_marks	attendence
ram	2.0	1	22.0	12	60.0	55.0
shyam	4.0	3	55.0	3	55.0	55.0
jodu	55.0	9	20.0	5	35.0	55.0
madhu	8.0	7	23.0	12	90.0	55.0
laxyam	7.0	12	55.0	2	55.0	55.0
rohit	55.0	23	32.0	67	77.0	55.0
rimpa	13.0	14	55.0	34	79.0	55.0

ii).Drop Missing Values

If you want to simply exclude the missing values, then use the dropna function along with the axis argument. By default, axis=0, i.e., along row, which means that if any value within a row is NA then the whole row is excluded(or, the whole row will be removed). If we take axia=1 then if any value within a row is NaN then the whole column is removed. Let see the examples below.

- Example 1 : removing columns if we found any value within a column is NaN

In [39]:
```
# create a data frame std
import pandas as pd
import numpy as np # for define a NaN values we import numpy function then we
will use np.NaN() function.
d = {"roll number" :[2,4,np.NaN,8,7,np.NaN,13],
   "registraction no" :[1,3,9,7,12,23,14],
    "age":[22,np.NaN,20,23,np.NaN,32,np.NaN],
    "doc_id no":[12,3,5,12,2,67,34],
    "per_marks":[60,np.NaN,35,90,np.NaN,77,79],
    "attendence":[80,np.NaN,np.NaN,np.NaN,np.NaN,np.NaN,np.NaN]}
std = pd.DataFrame(d,index=["ram","shyam","jodu","madhu","laxyam","rohit","r
impa"])
print("our given dataframe is:\n ",std,"\n \n ")
# remove columns if we found any value within a column is NaN
print("the resuilting data frame after removing columns if we found any value
within a column is NaN is :")
std.dropna(axis=1)
```

```
our given dataframe is:
         roll number  registraction no   age  doc_id no  per_marks  attende
nce
ram              2.0                 1  22.0         12       60.0        80.
0
shyam            4.0                 3   NaN          3        NaN         Na
N
jodu             NaN                 9  20.0          5       35.0         Na
N
madhu            8.0                 7  23.0         12       90.0         Na
N
laxyam           7.0                12   NaN          2        NaN         Na
N
rohit            NaN                23  32.0         67       77.0         Na
N
rimpa           13.0                14   NaN         34       79.0         Na
N


the resuilting data frame after removing columns if we found any value withi
n a column is NaN is :
```

Out[39]:

	registraction no	doc_id no
ram	1	12
shyam	3	3
jodu	9	5
madhu	7	12
laxyam	12	2
rohit	23	67
rimpa	14	34

- Example 2 : removing rows if we found any value within a rows is NaN : for these situation we use the comment axis=0

In [26]:
```
print("the resuilting data frame after removing rows if we found any value within a row is NaN is :")
std.dropna(axis=0)
```

```
the resuilting data frame after removing rows if we found any value within a row is NaN is :
```

Out[26]:

	roll number	registraction no	age	doc_id no	per_marks	attendence
ram	2.0	1	22.0	12	60.0	80.0

In [ ]:
```
```

Transposition of a DataFrame

In [40]:
```
std.T
```

Out[40]:

	ram	shyam	jodu	madhu	laxyam	rohit	rimpa
roll number	2.0	4.0	NaN	8.0	7.0	NaN	13.0
registraction no	1.0	3.0	9.0	7.0	12.0	23.0	14.0
age	22.0	NaN	20.0	23.0	NaN	32.0	NaN
doc_id no	12.0	3.0	5.0	12.0	2.0	67.0	34.0
per_marks	60.0	NaN	35.0	90.0	NaN	77.0	79.0
attendence	80.0	NaN	NaN	NaN	NaN	NaN	NaN

In [41]:
```
std.transpose()
```

Out[41]:

	ram	shyam	jodu	madhu	laxyam	rohit	rimpa
roll number	2.0	4.0	NaN	8.0	7.0	NaN	13.0
registraction no	1.0	3.0	9.0	7.0	12.0	23.0	14.0
age	22.0	NaN	20.0	23.0	NaN	32.0	NaN
doc_id no	12.0	3.0	5.0	12.0	2.0	67.0	34.0
per_marks	60.0	NaN	35.0	90.0	NaN	77.0	79.0
attendence	80.0	NaN	NaN	NaN	NaN	NaN	NaN

To import dataset in python

Dataset from the text,csv and excell can be importated to python by using pandas library.

How to Import an Excel File into Python using Pandas?

Reference : https://datatofish.com/read_excel/ (https://datatofish.com/read_excel/) To import excell file into python at first it is neccessary to import panadas library as a nick name(say pd).i.e for importing panadas library we use the followings comment $import\ pandas\ as\ pd.$
Then we will run the comment
$pd.\ read\_excel(r'Path\ where\ the\ Excel\ file\ is\ stored\File\ name.\ xlsx').$

In [10]:
```
import pandas as pd #import panndas library
data3=pd.read_excel(r'D:\Books(statistics)\python\pythondata\datapython.xlsx'
)
data3

#pd.read_excel (r'C:\Users\SUMAN GHOSH\Desktop\datapython.xlsx')
```

Out[10]:

	Product	Price	number of items	warienty	defect probabilty	available shop
0	Desktop Computer	700.0	1	4	0.1	34
1	Tablet	250.0	3	3	0.4	5
2	iPhone	800.0	1	6	0.7	67
3	Laptop	1200.0	1	1	0.5	22
4	NaN	NaN	1	8	0.0	3564
5	headphone	150.0	1	9	0.1	123
6	mouse	300.0	2	4	0.0	875
7	keybord	600.0	1	21	0.0	1234

How to Import an Excel File into Python using Pandas?

The procedure of import csv file is same as import excell file. Only we will take 'csv' in stead of 'excell'.

In [11]:
```
data1=pd.read_csv(r'D:\Books(statistics)\python\pythondata\bigmartdata.csv')
data1
```

Out[11]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Out
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092	
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692	
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180	
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614	
...	...	...	...	...	...	...	...
8518	FDF22	6.865	Low Fat	0.056783	Snack Foods	214.5218	
8519	FDS36	8.380	Regular	0.046982	Baking Goods	108.1570	
8520	NCJ29	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	
8521	FDN46	7.210	Regular	0.145221	Snack Foods	103.1332	
8522	DRG01	14.800	Low Fat	0.044878	Soft Drinks	75.4670	

8523 rows × 12 columns

In [ ]:
```

```

a).sortnig a dataframe in pundas

we can compute shorting a particular column by using the function $name\ of\ dataframe.\ sort\_values(by =' \ name\ of\ the\ column', kind =' \ heapsort')$ .Let see the example below.....

In [81]:
```
d3 = {'roll number' :[2,4,6,8,7,9,13],
   'registraction no' :[1,3,5,7,12,23,14],
    'age':[22,21,20,23,27,32,25],
    'doc_id no':[12,3,5,6,2,67,34],
    'per_marks':[60,70,35,90,81,77,79],
    'attendence':[45,78,67,89,75,78,82]}
unsorteddf = pd.DataFrame(d3,index=["rohim","shyam","kartik","madhu","laxya
m","rohit","rimpa"])
print("our given dataframe is:\n",unsorteddf)
```

```
our given dataframe is:
         roll number  registraction no  age  doc_id no  per_marks  attendenc
e
rohim              2                 1   22          12         60          45
shyam              4                 3   21           3         70          78
kartik             6                 5   20           5         35          67
madhu              8                 7   23           6         90          89
laxyam             7                12   27           2         81          75
rohit              9                23   32          67         77          78
rimpa             13                14   25          34         79          82
```

Now we will sort these dataframe accoring to the column age by using the function sort_values() .

In [82]:
```
sorted_df = unsorteddf.sort_values(by='age',kind='heapsort')
sorted_df
```

Out[82]:

	roll number	registraction no	age	doc_id no	per_marks	attendence
kartik	6	5	20	5	35	67
shyam	4	3	21	3	70	78
rohim	2	1	22	12	60	45
madhu	8	7	23	6	90	89
rimpa	13	14	25	34	79	82
laxyam	7	12	27	2	81	75
rohit	9	23	32	67	77	78

b). Descriptive Statistics

As I briefly showed earlier, when we use the describe() function we get the descriptive statistics for numerical columns, but the character columns are excluded.

Let's first create a DataFrame showing student names and their scores in Math and English:

In [66]:
```
import pandas as pd

data = {
    'Name': ['John', 'Alice', 'Joseph', 'Alex'],
    'English': [64, 78, 68, 58],
    'Maths': [76, 54, 72, 64],
    'physics':[78,45,23,56]
}

df = pd.DataFrame(data)
print(df)
```

```
     Name  English  Maths  physics
0    John       64     76       78
1   Alice       78     54       45
2  Joseph       68     72       23
3    Alex       58     64       56
```

We only have to call the describe() function on the DataFrame and get the various measures like the mean, standard deviation, median, maximum element, minimum element, etc:
As you can see, the describe() method completely ignored the "Name" column since it is not numberical, which is what we want. This simplifies things for the caller since you don't need to worry about removing non-numerical columns before calculating the numerical stats you want.

- Statistical summery of all the columns of a dataframe

In [67]:
```
df.describe()
```

Out[67]:

	English	Maths	physics
count	4.000000	4.000000	4.000000
mean	67.000000	66.500000	50.500000
std	8.406347	9.712535	22.898326
min	58.000000	54.000000	23.000000
25%	62.500000	61.500000	39.500000
50%	66.000000	68.000000	50.500000
75%	70.500000	73.000000	61.500000
max	78.000000	76.000000	78.000000

- Statistical summery of an individual column Example: here, we will find the statistical summery of the column'english' by using describe() function

In [68]:
```
df['Maths'].describe()
```

Out[68]:
```
count     4.000000
mean     66.500000
std       9.712535
min      54.000000
25%      61.500000
50%      68.000000
75%      73.000000
max      76.000000
Name: Maths, dtype: float64
```

In [58]:
```
df['English'].describe()
```

Out[58]:
```
count     4.000000
mean     67.000000
std       8.406347
min      58.000000
25%      62.500000
50%      66.000000
75%      70.500000
max      78.000000
Name: English, dtype: float64
```

- statistical summery of multiple columns: For selecting multiple columns simultaneously we use the function $nameofthedataframe.loc[:,['first\ column',....,['last\ column']]$. then we apply $describe()$ function for calculating summary of statistics. let see the example below.
  calculate summary of statistics of the two columns 'math' and 'english' ?

In [71]:
```
df.loc[:,['Maths','English']].describe()
```

Out[71]:

	Maths	English
count	4.000000	4.000000
mean	66.500000	67.000000
std	9.712535	8.406347
min	54.000000	58.000000
25%	61.500000	62.500000
50%	68.000000	66.000000
75%	73.000000	70.500000
max	76.000000	78.000000

In [83]:
```
df.sum()
```

Out[83]:
```
Name       JohnAliceJosephAlex
English                    268
Maths                      266
physics                    202
dtype: object
```

In [84]:
```
df.mean()
```

Out[84]:
```
English    67.0
Maths      66.5
physics    50.5
dtype: float64
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```
sum=d=0
while d !=10:
    d+=0.1
    sum+=sum+d
```

In [1]:
```
d={"s":6,"y":7}
d["s"]
```

Out[1]:
```
6
```

In [3]:
```
L=list()
L.append([1,2,[3,4]])
L.extend([5,6,7])
print(L)
```

```
[[1, 2, [3, 4]], 5, 6, 7]
```

In [8]:
```
for i in range(1,6+1):
    for j in range(6,0,-1):
        print(j if j<=i else"",end ="")
    print()
```

```
1
21
321
4321
54321
654321
```

In [ ]: