import numpy as np
import pandas as pd

what is Numpy
- A package in python which stands for 'Number python'
- It is used for mathematical and scientific computation which contains multi-dimensional arrays & matrices
- Numpy also provides a module called 'linalg' which contains various functions (such as det, eig, norm) to apply linear algebra on Numpy array
- Numpy array is a central structure of the Numpy library it is an n-dimensional array object containing rows and columns

what is an array?
- A numpy array looks similar to a list
- It is used for mathematical and scientific computation which contains multi-dimensional arrays and matrices
- Numpy also provides a module called 'linalg' which contains various functions (such as det, eig, norm) to apply linear algebra on Numpy array.
- Numpy array is a central structure of the Numpy library. It is an n-dimensional array object containing rows and columns.
- A array is a grid of values, indexed by a tuple of positive integers
- It usually contains numeric values, However it can contain string values too
- They work faster than lists
- An array can be n-dimensional.

Creating array from list

np.array()

```
np.array([2,3,1,6,87])
array([2,3,1,6,87])

li = [3,2,1,6,0]
np.array(li)
array([3,2,1,0])
```

type()

```
weight = [12,13,14,45,15,18,19,20,23,45,67,89,90]
print(weight)

[12,13,45,15,18,19,20,23,45,67,89,90]
type(weight)
list : output
```

dtype()

```
li = ['hello', 78, 12.23, True, 8+5j]
ar = np.array(li)
ar.dtype
dtype('<U64')

li = [78,12,23, True, 8+5j]
ar = np.array
ar.dtype
dtype('complex128')
```

(random.random())

```
a1 = np.random.random(10)
a1
array([0.1729937 , 0.91665751, 0.77065805, 0.58270479,
       0.34893446, 0.61299962, 0.39009311, 0.48253307,
       0.13439108, 0.76823019])
```

= Normal distribution

gives uniformly distributed data excluding 1 = random.random

(random.randn())

```
a2 = np.random.randn(10)
a2
array([ 0.0721272 ,  0.50846854,  1.5359024 ,  0.47110143,
        0.07586959, -0.78129108, -0.67446499,  1.11045824,
        0.758977  , -0.51109021])
```

gives the random value from standard normal distribution, where n=0, sd=1

(random.randint())

```
a3 = np.random.randint(10,50,20)
a3
array([24, 47, 30, 48, 36, 27, 48, 17, 11, 29, 44, 31, 18, 42,
       38, 40, 14, 45, 38, 36])
```

(np.arrange())

functionname → Numpy.arrange(start, stop, step, dtype) ← datatype

stop: endofinterval

step: step between values

```
np.arrange(10,21)
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20])

np.arrange(0,101,5)
array([  0,   5,  10,  15,  20,  25,  30,  35,  40,  45,  50,  55,  60,
        65,  70,  75,  80,  85,  90,  95, 100])
```

```
arr.shape
(4,2)
a1.shape
(10,)
```

itemsize ✓

```
arr.itemsize
4
```

ndim ✓

```
arr3 = np.array([[[1,2,3],[4,5,6]],[[7,8,9],[10,11,12]]])
arr3
array([[[ 1, 2, 3 ],
        [ 4, 5, 6 ]],
       [[ 7, 8, 9 ],
        [ 10, 11, 12 ]]])
arr3.shape
(2,2,3)
arr3.ndim ✓
3
a3.dtype
dtype('int32')
arr3.itemsize
4
```

Creating Identity, zero, full & eye matrix

identity() ✓

```
np.identity(5,dtype=int)
array([[ 1, 0, 0, 0, 0 ],
       [ 0, 1, 0, 0, 0 ],
       [ 0, 0, 1, 0, 0 ],
       [ 0, 0, 0, 1, 0 ],
       [ 0, 0, 0, 0, 1 ]])
```

output =

Eye

```
np.eye(5, 4, dtype=int, k=-1)
array([[0, 0, 0, 0],
       [1, 0, 0, 0],
       [0, 1, 0, 0],
       [0, 0, 1, 0],
       [0, 0, 0, 1]])
```

2D Array, 3D Array and .reshape()

1D

```
a = np.random.randint(10, 50, 20)
array([43, 30, 46, 28, 10, 27, 25, 32, 24, 43, 37, 31,
       17, 28, 18, 37, 35, 45, 38, 24])
```

2D

```
np.random.randint(10, 50, 20).reshape(5, 4)
array([[13, 25, 33, 18],
       [40, 47, 22, 34],
       [18, 29, 39, 43],
       [29, 29, 43, 31],
       [46, 48, 45, 35]])
```

3-D

```
np.random.randint(10, 50, 60).reshape(3, 5, 4)
array([[[10, 29, 31, 41],
        [38, 44, 13, 11],
        [48, 38, 33, 34],
        [41, 33, 16, 30],
        [17, 39, 20, 25]],

       [[30, 27, 16, 39],
        [26, 21, 44, 49],
        [42, 26, 37, 38],
        [12, 49, 11, 43],
        [34, 26, 24, 44]],

       [[39, 35, 27, 24],
        [25, 38, 10, 16],
        [23, 29, 33, 34],
```

Indexing and Slicing of Array

Index - 1D Array
each element in the Array can be accessed by passing the positional index of the element. The index for an array starts at 0 from left. It starts at -1 from the right

Index - 2D Array
Element in 2D Array can be accessed by the row and column indices. We can also select a specific row or column by passing the respective index

```
a
array([[10, 45, 28, 27, 23, 15, 22, 48, 37, 47, 16, 31, 32, 46
        24, 27, 49, 46, 12, 15, 39, 43, 39, 35, 31, 27, 39, 3
        29, 12, 17, 36, 41, 10, 20, 37, 10, 10, 28, 35, 12,
        16, 28, 23, 37, 26, 26, 40, 30, 29]])
```

45, 11, 40,
34, 39, 41
25, 30, 35

```
a[4]
 23
a[-5]
 26
```

Index 2D array:
Element in 2D Array can be accessed by the row & column indices. We can also select a specific row or column by passing the respective index

```
a[:5]
array([[10, 45, 28, 27, 23]])
a[-7:]
array([[23, 37, 26, 26, 40, 30, 29]])
a[::-1]
```

> ये पूरे array को reverse कर देगा

print कर देगा।

```
mat[2]    array([17, 14, 13])
mat[1]    array([11, 13, 19])

mat[-2:]
  array([[11, 13, 19],
         [17, 14, 13]])

mat[:, 0]
 array([16, 11, 17])

mat[:, :2]
 array([[16, 12],
        [11, 13],
        [17, 14]])

mat[:, -2:]
 array([[12, 19],
        [13, 19],
        [14, 13]])
mat[1]
array([11, 13, 19])
mat[1].shape
(3,)

mat[1:2, :]
array([[11, 13, 19]])
 mat[1:2, :].shape
 (1, 3)
mat[1].ndim
1
mat[1:2, :]ndim
2
mat[0, -2:].shape
(2,)
mat[0:1, -2:].shape
(1, 2)
```

```
mat[:, 1::2]
array([[17, 16],
       [16, 17],
       [13, 10],
       [15, 11],
       [11, 11]])
mat[:, [0, -2, -1]]
array([[13, 10, 13],
       [12, 17, 12],
       [12, 10, 19],
       [13, 11, 16],
       [19, 11, 15]])
```

RISHI TANWAR

RISHI TAN

```
arr3[ ]
array([[[ 1, 2, 9],
        [ 4, 5, 6]],
       [[ 7, 8, 9],
        [10, 11, 12]]])
```

arr3[0:1] - 1 dimensional
~~arr3[0:1~~ array([1])
arr3[0:1, 0:1]
array([[1]]) - 2 dimens
arr3[0:1, 0:1, 0:1]
array([[[1]]]) 3 dimens

```
arr3[:, :, -2:]
array([[[ 2, 3]
        [ 5, 6]],
       [[ 8, 9],
        [11, 12]]])
```

COMPARISON OPERATIONS ON ARRAY

a
array([10, 45, 28 27 23 15 22, 48, 37 47 16
21, 27,
29, 12,
16, 28,

a>30

जहाँ भी a>30 होगा वहाँ पर True print हो जाएगा
और जहाँ a>30 नहीं होगा वहाँ False print हो जाएगा

a[a>30]
array([45,48,37,47,31,32,46,45,40,49,46,39,43,39,
34,39,41,38,41,37,39,35,31,36,39,37,

a[(a>30) & (a%2==0)]
array([48,32,46,40,46,34,34,36,36,40])

(a>30) & (a%2==0)
array([True, False print होगा])

a=np.array([2,3,5,6])
a[[True, False, False, True]] यहाँ True वो print करेगा.

ARITHMETIC OPERATIONS ON ARRAY

a+3
array([5,6,8,9])

a-5
array([-3,-2,0,1])

np.array([2,3,4,5]) + np.array([3,6,8,9])
array([5,9,12,14])

a=np.array([2,3,4,5])
b=np.array([3,6,8,9])
a+b
array([5,9,12,14])

a*b
array([6,18,32,45])

14 an array.

a.sum() ✓
14

```
mat1 = np.random.randint(1,11,9).reshape(3,3)
mat2 = np.random.randint(1,11,9).reshape(3,3)
mat1
    array([[1,5,5],
           [9,1,8],
           [9,6,3]])
mat2
array([[6,3,9],
       [1,7,6],
       [2,3,9]])
mat1+mat2 #Element wise
[
[
[
]
]
]
```
✓

```
mat1-mat2
array([[-5,2,-4],
       [8,-6,2],
       [7,3,-6]])
```
✓

np.sum(mat1) sum of all the elements ✓
50

np.sum(mat1, axis=1) → sum of rows.
array([21,13,16]) 21 = 8+7+6, 13 = 10+2+1, 16 = 5+10+1 ✓

```
mat1
array([[8,7,6],
       [10,2,1],
       [5,10,1]])
```

np.sum(mat1, axis=0) ✓
array([23,19,8])

Splitting of arrays

np.split and np.array_split()

```
a
array([2,3,4,5])
a1,a2 = np.split(a,2)
a1
array([2,3])
a2
array([4,5])

np.array_split(a,3)
[array([2,3]), array([4]), array([5])]

a = np.random.randint(20,30,8)
a
array([27,20,29,22,21,24,29,29])

np.array_split(a,3)
[array([27,20,29]), array([22,21,24]), array([29,29])]

8//3   2
5//2   2

mat4 = np.random.randint(1,11,20).reshape(5,4)
mat4
array([[ 4, 3, 7, 3],
       [10, 6, 10, 10],
       [ 9, 1, 7, 9],
       [ 1, 7, 7, 6],
       [ 9, 7, 10, 10]])

[array([[ 7  2  7  8  10],
        [ 5  1  10 4  10]]),
 array([[ 7  10 3  8],
        [ 2  2  2  6  4]])]
```

np.split(mat4,2)

SERIES AND DATAFRAME

```
age = [34, 32, 45, 21, 67, 55, 43]
pd.Series(age, name = 'AGE')
```

```
0 34
1 32
2 45
3 21
4 67
5 55
6 43
```

```
l = ['hello', 21, 34, 'my', 'friend', 11.2]
pd.series(l)
```

```
0 hello
1   21
2   34
3   my
4   friend
5   11.2
dtype : object
```

```
l = ['hello', 21, 34, 'my', 'friend', 11.2, None]
pd.series(l)
```

```
0 hello
1   21
2   34
3   my
4   friend
5   11.2
6   None
dtype : object
```

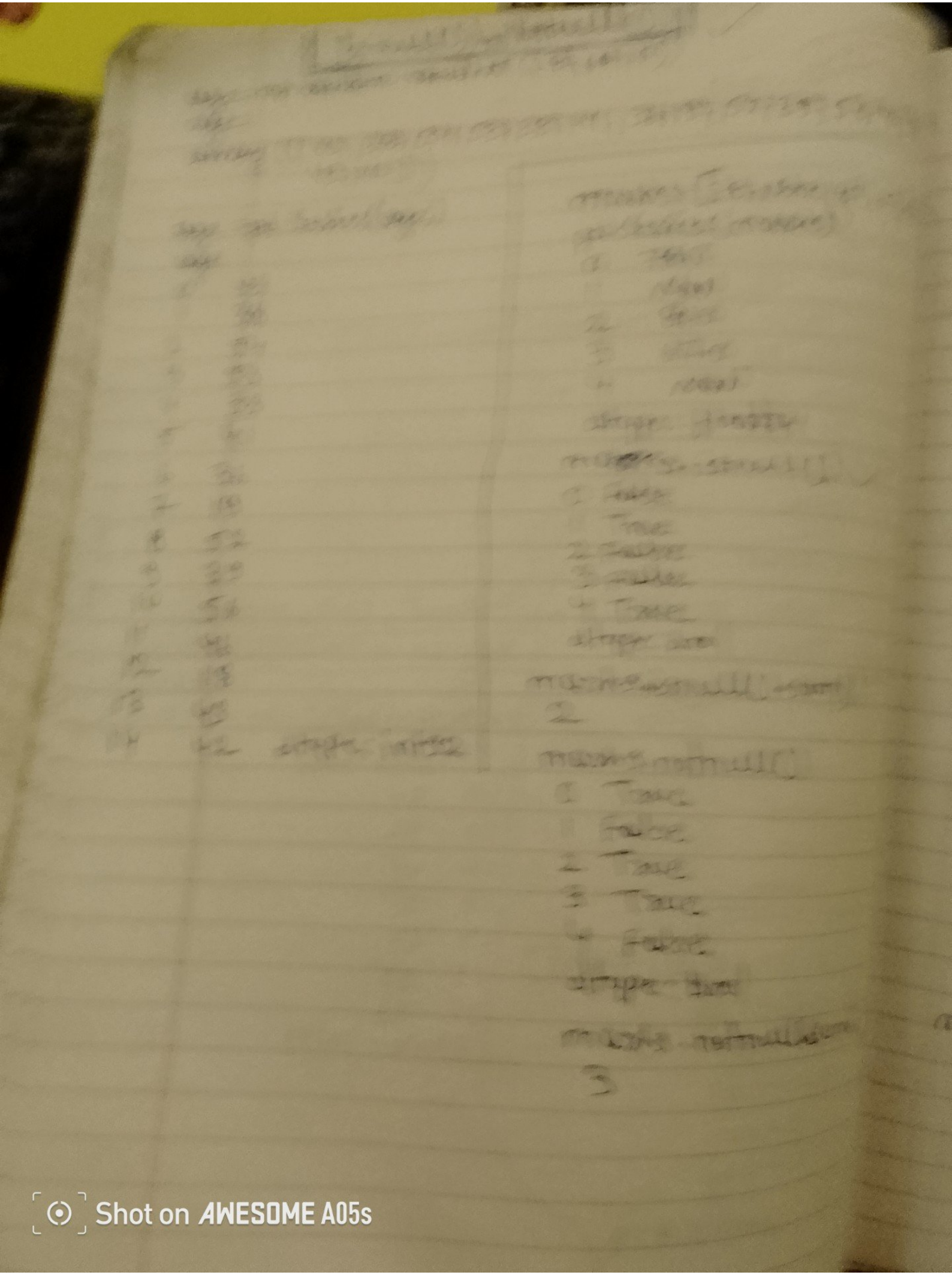
```
l = [21, 34, 11.2, 5+4j]
pd.series(l)
```

```
0 21.0+0.0j
1   34.0+0.0j
2   11.2+0.0j
3   5.0+4.0j
dtype : complex128
```

SERIES WITH USER DEFINED INDEXES

```
pd.Series(age, index = [10, 11, 12, 13,
   14, 15, 16])
```

```
10  34
11  32
12  45
13  21
14  67
15  55
16  43
dtype : int64
```



//RANKING AND SORTING// IN SERIES

```
age
0  43
1  38
2  54
3  53
4  33
5  41
6  36
7  19
8  52
9  39
10 56
11 46
12 18
13 43
14 42
dtype: int32
```

age.sort_value()

```
12  18
7   19
4   33
6   36
1   38
9   39
5   41
14  42
0   43
13  43
11  46
8   52
3   53
2   54
10  56
dtype: int32
```

age.sort_values(ignore_index=True)

```
0   18
1   19
2   33
3   36
4   38
5   39
6   41
7   42
8   43
9   43
10  46
11  52
12  53
13  54
14  56
dtype: int32
```

marks.sort_values(ascending=False, ignore_index=True)

```
0  81.0
1  78.0
2  67.0
3  NaN
4  NaN
dtype: float64.
```

marks.sort_values(ascending=False, ignore_index=True, na_position='first')

```
0  NaN
1  NaN
2  81.0
3  78.0
4  67.0
dtype: float64.
```

marks.sort_values(ascending=False, ignore_index=True, na_position='first', inplace=True)

```
1  3.0
2  4.0
3  2.0
4  2.0
5  6.0
6  4.0
7  1.0
8  5.0
9  1.0
10 dtype: float64
```

DATA FRAME

```
emp_id = pd.Series([1011,1012,1013,1014,1015])
name = pd.Series(['Ajay','Mukesh','Anil','Suman',
                  'Rajat'])
salary = pd.Series([78000,None,81000,None,
                    167000])
pd.DataFrame()
```

	0	1	2	3	4
0	1011	1012	1013	1014	1015
1	Ajay	Mukesh	Anil	Suman	Rajat
2	78000.0	NaN	81000.0	NaN	167000.0

```
pd.DataFrame({'EMPID': emp_id, 'EMP NAME':
              name, 'SALARY': salary})
```

	EMPID	EMPNAME	SALARY
0	1011	Ajay	78000.0
1	1012	Mukesh	NaN
2	1013	Anil	81000.0
3	1014	Suman	NaN
4	1015	Rajat	167000.0

```
pd.DataFrame([['Apple','Fruit',70],['Orange','Fruit',
              40],['Potato','Vegetable',20]])
```

	0	1	2
0	Apple	Fruit	70
1	Orange	Fruit	40
2	Potato	Vegetable	20

	EMP ID	EMPNAME	SALARY
0	1011	Ajay	78000.0
1	1012	Mukesh	NaN
2	1013	Anil	81000.0
3	1014	Suman	NaN
4	1015	Rajat	167000.0

ACCESING COLUMNS IN DATAFRAME

```
data2.Product
0    Apple
1    Orange
2    Potato
3    Orange
4    Carrot
5    Apple
6    Onions
Name: Product, dtype: object
```

```
data2.Price
0    70.0
1    40.0
2    20.0
3    40.0
4    NaN
5    70.0
6    40.0
```

```
data2['Category']
0    Fruit
1    Fruit
2    Vegetable
3    Fruit
4    Vegetable
5    Fruit
6    Vegetable
```

data2[['Product', 'Category']]

	Product	Category
0	Apple	Fruit
1	Orange	Fruit
2	Potato	Vegetable
3	Orange	Fruit
4	carrot	Vegetable
5	Apple	Fruit
6	Onions	Vegetable.

data1['EMP ID']

```
0    1011
1    1012
2    1013
3    1014
4    1015
```

data1[['EMPNAME', 'SALARY']]

	EMPNAME	SALARY
0	Ajay	78000.0
1	Mukesh	NaN
2	Anil	81000.0
3	Suman	NaN
4	Rajat	167000.0

So in data frames and series for logical we can not use and or, not. we can only use bitwise, operators & (and), | (or), ~ (not)

```
data2[~(data2.Category == 'Vegetable')]
```

	Product	Category	Price
0	Apple	Fruit	70.0
1	Orange	Fruit	40.0
3	Orange	Fruit	40.0
5	Apple	Fruit	70.0

```
data2[~((data2.Category == 'Vegetable') | (data2.Price > 50))]
```

	Product	Category	Price
1	Orange	Fruit	40.0
3	Orange	Fruit	40.0

.loc and .iloc (Draught)

```
data1.iloc[:,:2]
```

	EMPID	EMPNAME
0	1011	Ajay
1	1012	Mukesh
2	1013	Anil
3	1014	Suman
4	1015	Rajat

```
data1.iloc[1]
EMP ID        1012
EMP NAME      Mukesh
SALARY        NaN
Name: 1, dtype: object
```

```
data2.iloc[:3,-2:]
```

	Category	Price
0	Fruit	70.0
1	Fruit	40.0
2	Vegetable	20.0

```
data1.iloc[:,[0,-1]]
```

	EMPID	SALARY
0	1011	Ajay
1	1012	Mukesh
2	1013	Anil
3	1014	Suman
4	1015	Rajat

data1.loc[:3,['EMP NAME','SALARY']]

	EMP NAME	SALARY
0	Ajay	78000.0
1	Mukesh	NaN
2	Anil	81000.0
3	Suman	NaN

data1.loc[data1.SALARY.isnull(),['EMP ID','EMP...

	EMP ID	EMP NAME
1	1012	Mukesh
3	1014	Suman.

RANKING & SORTING DATA FRAME

data1.sort_values(by=['SALARY'])

	EMPID	EMPNAME	SALARY
0	1011	Ajay	78000.0
2	1013	Anil	81000.0
4	1015	Rajat	167000.0
1	1012	Mukesh	NaN
3	1014	Suman	NaN

data.sort_values(by=['SALARY'],ascending=fa...

	EMPID	EMPNAME	SALARY
4	1015	Rajat	167000.0
2	1013	Anil	81000.0
0	1011	Ajay	78000.0
1	1012	Mukesh	NaN
3	1014	Suman	NaN

data1.sort_values(by=['SALARY'],ascending=False,na_position='f...

	EMP ID	EMP NAME	SALARY
1	1012	Mukesh	NaN
3	1014	Suman	NaN
4	1015	Rajat	167000.0
2	1013	Anil	81000.0
0	1011	Ajay	78000.0

	EMPID	EMPNAME	SALARY
0	1012	Mukesh	NaN
1	1014	Suman	NaN
2	1015	Rajat	167000.0
3	1013	Anil	81000.0
4	1011	Ajay	78000.0

```
data2['Rank'] = data2.Price.rank(method='dense', ascending=
    False, na_options='bottom')
```

	Product	Category	Price	Rank.
0	Apple	Fruit	70.0	1.0
1	Orange	Fruit	40.0	2.0
2	Potato	Vegetable	20.0	3.0
3	Orange	Fruit	40.0	2.0
4	Carrot	Vegetable	NaN	4.0
5	Apple	Fruit	70.0	1.0
6	Onions	Vegetable	40.0	2.0

DUPLICATES

```
data2.duplicated()
0    False
1    False
2    False
3     True
4    False
5     True
6    False
dtype: bool
```

```
data2[data2.duplicated()]
```

	Product	Category	Price	Rank.
3	Orange	Fruit	40.0	2.0
5	Apple	Fruit	70.0	1.0

```
data2[data2.duplicated(keep='last')]
```

	Product	Category	Price	Rank.
0	Apple	Fruit	70.0	1.0
1	Orange	Fruit	40.0	2.0
3	Orange	Fruit	40.0	

```
data2[data2.duplicated(keep='first')]
```

	Product	Category	Price	Rank.
3	Orange			

||CONCAT AND APPEND OF SERIES AND ARRAY||

```
import pandas as pd
import numpy as np

ser1 = pd.Series(np.random.randint(10,30,10))
ser2 = pd.Series(np.random.randint(10,30,10))
ser3 = pd.Series(np.random.randint(10,30,10))
```

Concatenate Series

```
a= pd.concat([ser1, ser2], ignore_index=True, axis=1)
```

```
0  15
1  15
2  24
3  20
4  29
5  29
6  20
7  25
8  24
9  dtype: int32
```

```
df1 = pd.read_excel('data.xlsx', sheet_name=1)
df2 = pd.read_excel('data.xlsx', sheet_name=1)
df3 = pd.read_excel('data.xlsx', sheet_name=2)
df3)
```

	Emp ID	Emp Name	Dependents	Lavs	Proper
0	1011	Ajay	3	1	1
1	1012	Mukesh	2	1	0
2	1022	Priyanka	1	0	0
3	1023	Geeta	3	2	1
4	1015	Rita	2	1	1

```
# Concatenate dataframes df1 and df2
pd.concat([df1, df2])
```

	Emp ID	Emp Name	Deptt	Age	Salary
0	1011	Ajay	Sales	34	75000
1	1012	Mukesh	HR	28	35000
2	1013	Suman	HR	27	35000
3	1014	Neeraj	Sales	30	60000
4	1015	Rita	Market	32	67000
0	1021	Roshan	Sales	32	6600
1	1022	Priyanka	Sales	35	7100
2	1023	Geeta	HR	25	300
3	1024	Ashok	Operations	33	37
4	1025	Riya	HR	27	35

pd.concat([df1, df2], axis=1)

	EmpID	EmpName	Deptt	Age	Salary	EmpID	EmpName	Deptt
0	1011	Ajay	Sales	34	75000	1021	Roshan	Sales
1	1012	Mukesh	HR	28	35000	1022	priyanka	Sales
2	1013	Suman	HR	27	35000	1023	Greeta	HR
3	1014	Neeraj	Sales	30	60000	1024	Ashok	Operations
4	1015	Rita	Market	32	67000	1025	Riya	HR

Concatenate df1 df2 df3

pd.concat([df1, df2, df3])

	EmpID	EmpName	Deptt	Age	Salary	Dependents
0	1011	Ajay	Sales	34.0	75000.0	
1	1012	Mukesh	HR	28.0	35000.0	
2	1013	Suman	HR	27.0		
3	1014	Neeraj	Sales	30.0		
4	1015	Rita	Market	32.0		
0	1021	Roshan	Sales	32.0		
1	1022	Priyanka	Sales	35.0		
2	1023	Greeta	HR	25.0		
3	1024	Ashok	Operations	33.0		
4	1025	Riya	HR	27.0		
0	1011	Ajay	NaN	NaN		
1	1012	Mukesh	NaN	NaN		
2	1022	Priyanka	NaN	NaN		
3	1023	Greeta	NaN	NaN		

Concatenate([df1, df2], join='inner')

EmpID	EmpName	Deptt	Age	Salary

दोनों में Common data मिलेगा

pd.concatenate([df1, df3], join='inner')

Concatenate([df1, df2], join='outer')

pd.concatenate([df1, df3], join='outer')

दोनों का All data

Change the indexing of the Dataframes

df3=df3.reindex([3,4,5,6,7])

so indexing हो जाएगी df3 की।

fill_value=0 जहाँ पे भी NaN होगा वहाँ zero आ जाएगा।

pd.concat([df1,df3], axis=1, join='inner').

0 की जगह और भी value दे सकते हैं।

Joining the dataframe

df1

	EmpID	EmpName	Dept	Age	Salary
0	1011	Ajay	Sales	34	75000
1	1012	Mukesh	HR	28	35000
2	1013	Suman	HR	27	~~60000~~ 35000
3	1014	Neeraj	Sales	30	60000
4	1015	Rita	Market	32	67000

df2

	EmpID	EmpName	Dept	Age	Salary
0	1021	Roshan	Sales	32	66000
1	1022	Priyanka	Sales	35	72000
2	1023	Geeta	HR	25	30000
3	1024	Ashok	Operations	33	37000
4	1025	Riya	HR	27	35000

df1.join(df3, on=['EmpID'], lsuffix='_L')

दोनों data join हो जाएगी

lsuffix='L'
Rsuffix='R' → left side table जहाँ उसमें EmpID वाला है 2 ...

Set Index

df1.set_index(['EmpID'])

Emp ID	Emp Name	Depth	Age	Salary
1011	Ajay	Sales	34	75000
1012	Mukesh	HR	28	35000
1013	Suman	HR	27	35000
1014	Neeraj	Sales	30	60000
1015	Rita	Market	32	67000

df1.set_index(['Emp ID', 'Emp Name']).join(mdf3.set_index(['EmpID', 'Emp Name']))

Depth Age Salary Dependents (and 1 Proper

0
1
2

Emp ID	Emp Name
1011	Ajay
1012	Mukes
1013	
1014	
1015	

df1.set_index(['Emp ID']).join(df3.set_index(['Emp ID'], lsuffix='_L', rsuffix='_R')

Emp Name_L Depth Age Salary Emp Name_R Dependents

Emp ID

Right join

df1.set_index(['Emp ID']).join(df3.set_index(['Emp ID']), lsuffix='L', rsuffix='R', how='right')

left join

df1.set_index(['Emp ID']).join(df3.set_index(['Emp ID']), lsuffix='_L', rsuffix='_R', how='outer' left

outerjoin

df1.set_index(['Emp ID']).join(df3.set_index(['Emp ID']), lsuffix='_L', rsuffix='_R', how='outer')

Merge the DataFrame

df1.merge(df3, on=['Emp ID'], indicator=True)

EmpID EmpName_x Deptt Age Solary EmpName_y Dependents(a--

0
1
2

indicator=True

example ka hai ki sidemein ek column aur aajaega jisse ye pata chalega ki 2 columns konse table mein hai

_merge
both
both
left_only

Columns Rename

df1.rename(columns={'Emp Name':'EmpNames'}, inplace=True)

Upper case all elements in columns

df['EMP Name'] = df1['Emp Name'].str.upper()

(Reshape melt())

```
df1.melt(id_vars=['Emp ID'], value_vars=['Dept', 'Sa
         var_name='Info', value_name='Data')
```

	Emp ID	Info	Data
0	1011	Dept	Sales
1	1012	Dept	HR
2	1013	Dept	HR
3	1014	Dept	Sales
4	1015	Dept	Market
5	1011	Salary	75000
6	1012		35000
7	1013		35000
8	1014		60000
9	1015		67000.

Pivot Table

Pivot

```
pd.pivot_table(data=df1, index=['Dept'], values=['Salary'],
               aggfunc='sum')
```

Dept	Salary
HR	70000
Market	67000
Sales	135000.

aggfunc='mean'

~~margins~~ margins=True

```
pd.pivot_table(data=df1, index=['Dept'], columns=['Emp Name'],
               values=['Salary'], aggfunc='sum', margins=True,
               fill_value=0)
```

Emp Name / Dept	AJAY	MUKESH	NEERAJ	RITA	SOMAN	Salary All
HR	0	35000	0	0	35000	70000
Market	0	0	0	67000	0	67000
Sales	75000	0	60000	0	0	135000
All	75000	35000	60000	67000	35000	272000

Crosstable

Daraghi

```
data = pd.DataFrame({'Gender':['Male','Male','Female','Male',
        'Male','Female'],
    'Qual':['G','PG','Doc','G','PG','G','G']})

pd.crosstab(index=data['Gender'], columns=data['Qual'],
        values=data['Qual'], aggfunc='count')
```

Gender	Doc	G	PG
Female	1.0	2.0	NaN
Male	NaN	2.0	2.0

```
import seaborn as sns
data = sns.load_dataset('diamonds')

# Show the average price in cut on the bases
    of color
pd.crosstab(index=data['cut'], columns=data['color'],
        values=data['price'], aggfunc='mean')
```

Duplicates check

```
data.duplicated()
```

Map and replace

df1

	Emp ID	Emp Name	Dept	Age	Salary
0	1011	Ajay	Sales	34	35000
1	1012	Mukesh	HR	28	35000
2	1013	Suman	HR	27	95000
3	1014	Neeraj	Sales	30	60000
4	1015	Rita	Market	32	67000

Map

```
df1['Dept'] = df1['Dept'].map({'Sales': 'Finance', 'HR': 'Human
              Resource', 'Market': 'Marketing'})
```

df1

	Emp ID	Emp Name	Dept	Age	Salary
0	1011	Ajay			
1	1012	Mukesh			
2	1013	Suman			
3	1014	Neeraj			
4	1015	Rita			

Replace

```
df1['Dept'] = df1['Dept'].replace({'Finance': 'Sale
              inplace = True)
```

df1

	Emp ID	Emp Name	Dept	Age	Salary
0	1011	Ajay	None		
1	1012	Mukesh	None		
2	1013	Suman	None		
3	1014	Neeraj	None		
4	1015	Rita	None		

GROUP BY

```
var = pd.DataFrame({"Name":["a","b","c","d",
    "a","b","a","c","c","d"],
"S_1":[12,13,14,12,13,14,15,23,25,16,10],
"S_2":[23,24,25,26,27,28,29,30,25,34,35]})
```

var

	Name	S_1	S_2
0	a	12	23
1	b	13	24
2	c	14	25
3	d	12	26
4	a	13	27
5	b	14	28
6	a	15	29
7	b	23	30
8	a	25	25
9	c	16	34
10	c	10	35

```
var_new = var.groupby("Name")
var_new
for x,y in var_new:
    print(x)
    print(y)
```

a

	Name	S_1	S_2
0	a	12	23
4	a	13	27
6	a	15	29
8	a	25	25

b

	Name	S_1	S_2
1	b	13	24
5	b	14	28
7	b	23	30

c

	Name	S_1	S_2
2	c	14	25
9	c	16	34
10	c	10	35

d

	Name	S_1	S_2
3	d	12	26
11	d	34	56

```
var_new.get_group("d")
```

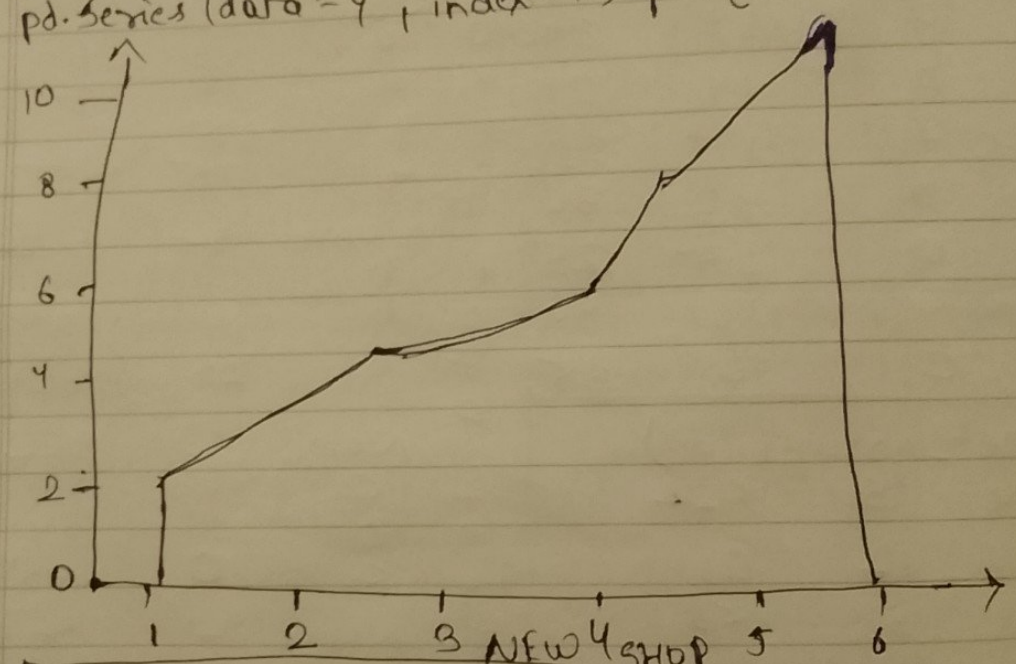
	Name	S_1	S_2
3	d	12	26
11	d	34	56

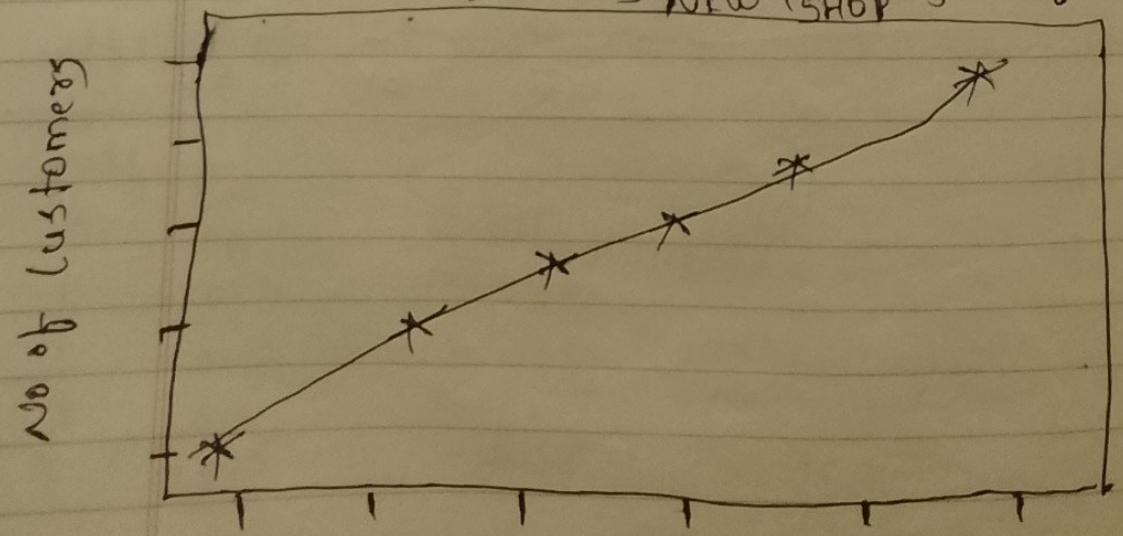
Visualization

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns.
```

LINE CHART

```
x = np.array([1,2,3,4,5,6])
y = np.array([2,4,5,6,8,11])
pd.Series(data=y, index=x).plot(kind='area')
```

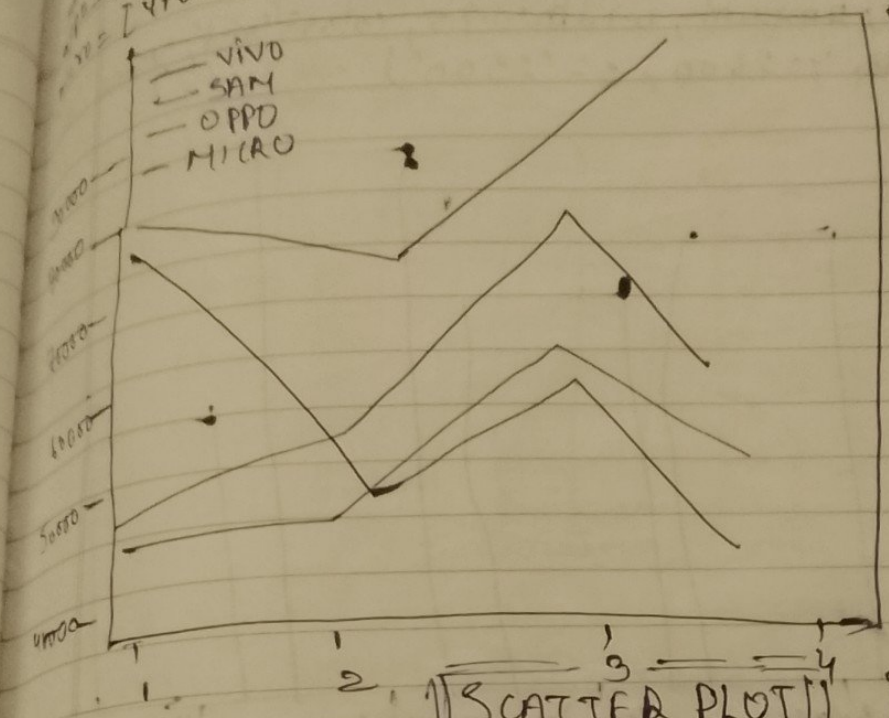





```
plt.plot(..., marker='*', color='purple', lw=5, ms=18, ls='-')
plt.xlabel('Week')
```

[1, 2, 3, 4]
[80000, 78000, 87000, 95000]
[45000, 45000, 60833, 54000]
[75000, 44000, 58000, 40888]
[47000, 55000, 78000, 65700]

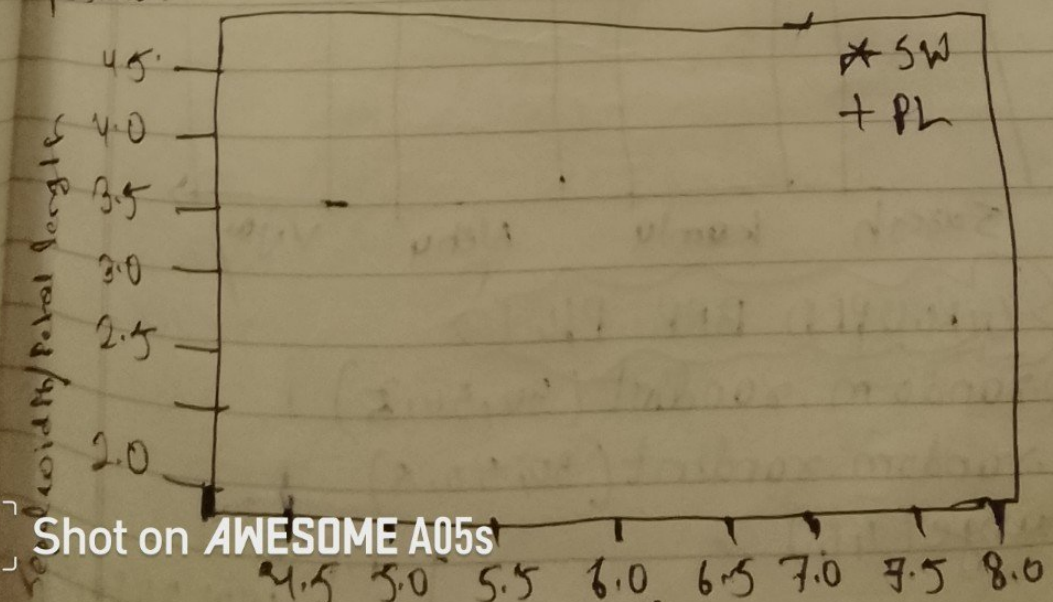
plt.figure(fig size=(5,4))
plt.plot(data, vivo, label='vivo')
plt.plot(data, samsung, label='SAM')
plt.plot(data, oppo, label='OPPO')
plt.plot(data, micro, label='MICRO')
plt.xticks(data)
plt.legend()
plt.show().



SCATTER PLOT

iris = sns.load_dataset('iris')
iris

plt.scatter(iris['sepal_length'], iris['sepal_width'], color='darkgreen', marker='*', label='SW')
plt.scatter(iris['sepal_length'], iris['sepal_width'], color='red', marker='+', label='PL')
plt.legend()
plt.ylabel('Sepal_width / Petal. length')
plt.xlabel('Sepal length')
plt.show()



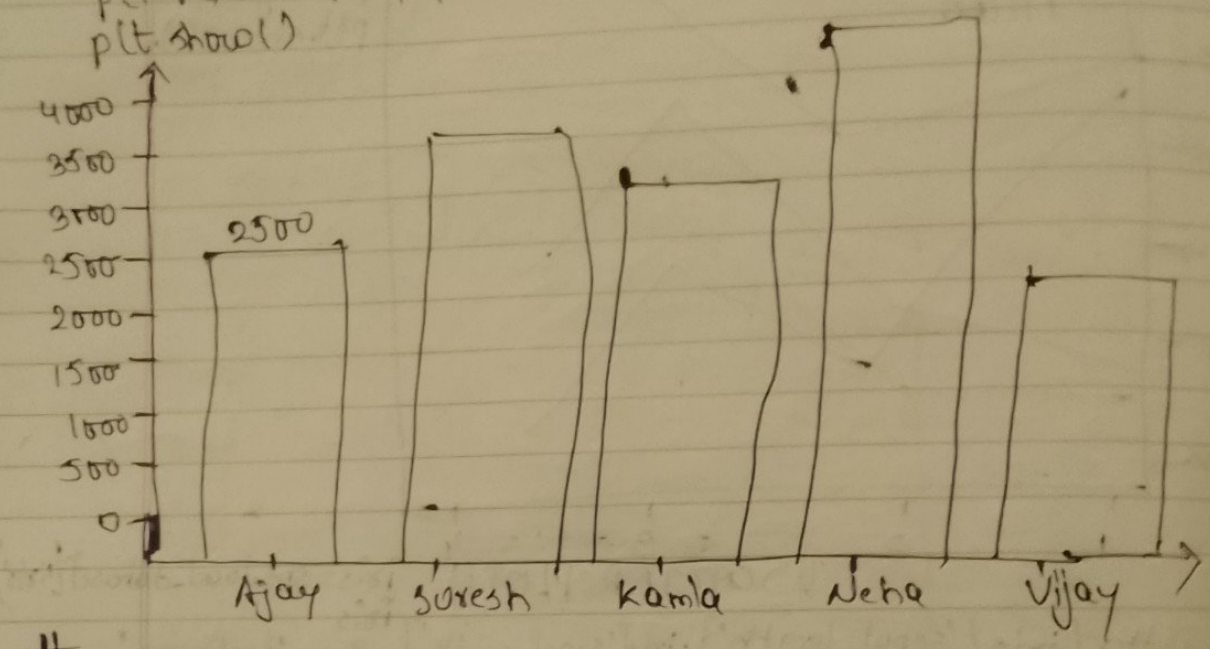
BAR PLOT

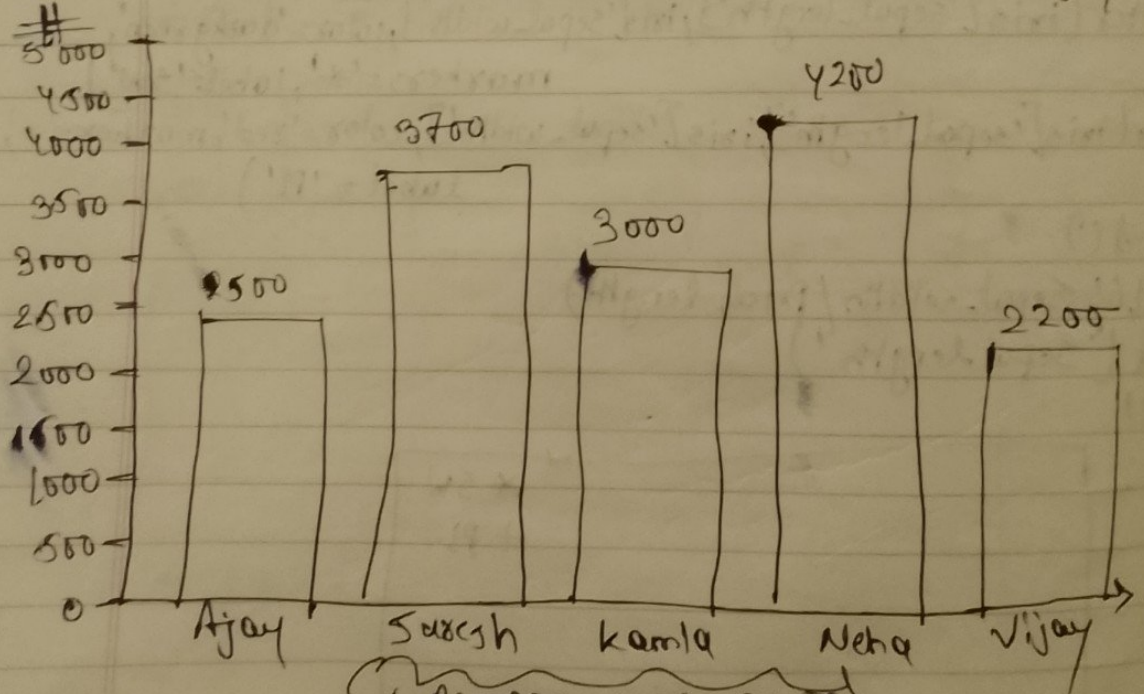
amount = [2500, 3700, 3000, 4200, 2200]

cust = ['Ajay', 'Suresh', 'kamla', 'Neha', 'Vijay']

make a vertical barplot b/w amount and cust

```
plt.bar(cust, amount, width=0.75, color=['red', 'green', ...
plt.text(x=0, y=2600, s='2500')
plt.show()
```





GROUPED BAR PLOT

python = np.random.randint(30, 50, 5)

npv = np.random.randint(30, 50, 5)

x = np.arange(1, 6)

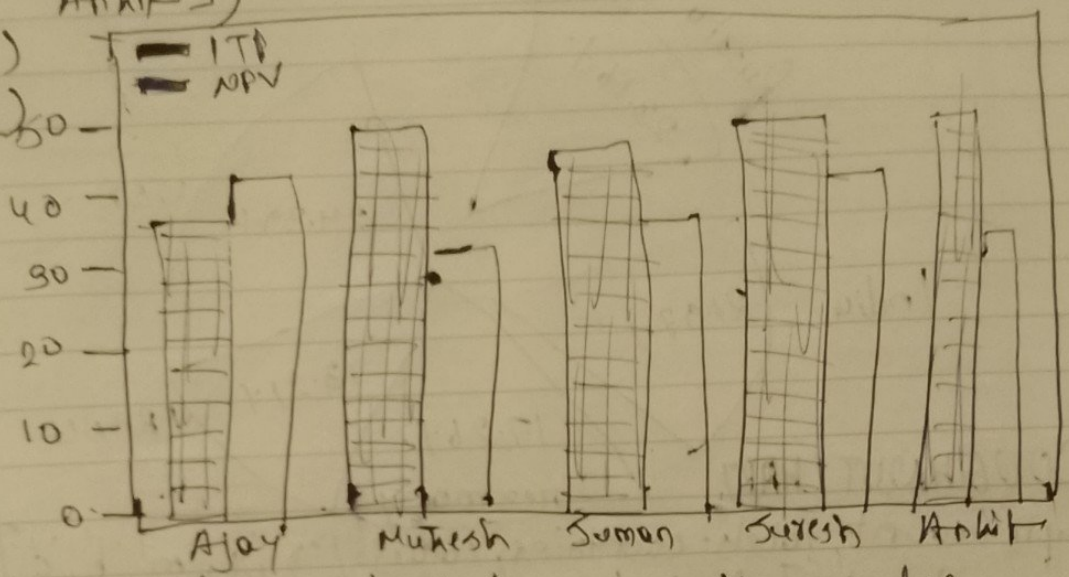
array([36, 48, 44, 47, 46])

npv

array([41, 31, 36, 31])

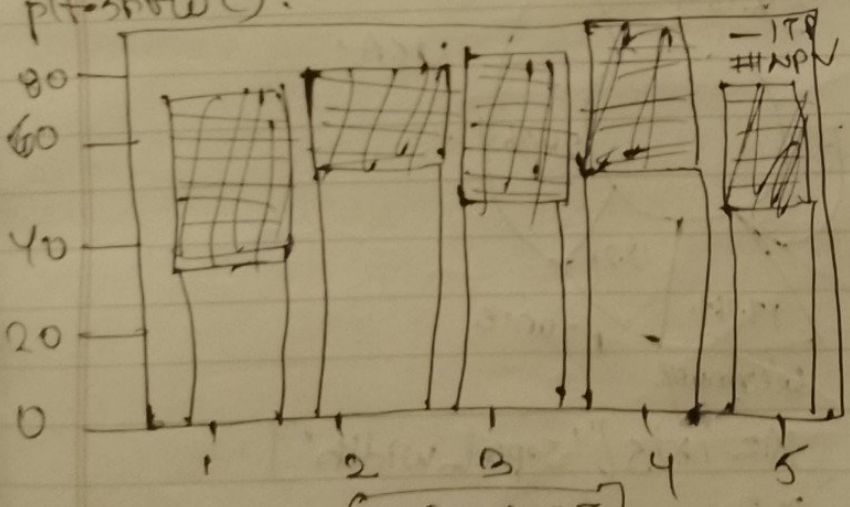
Make a plot grouped bar plot between python, npv and x

```
plt.bar(x=x, height=python, width=0.4, label='ITP')
plt.bar(x=x+0.4, height=npv, width=0.4, label='NPV')
plt.xticks(ticks=x+0.4/2, label=['Ajay', 'Mukesh', 'Suman', 'Suresh', 'Ankit'])
plt.legend()
plt.show()
```



Make a stacked bar plot between python and x

```
plt.bar(x=x, height=python, label='ITP')
plt.bar(x=x, height=npv, bottom=python, label='NPV')
plt.legend()
plt.show().
```

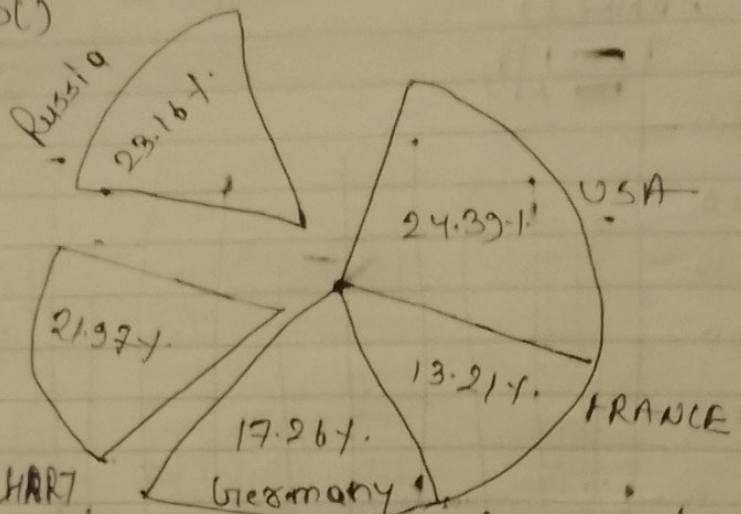


(PIE PLOT)

```
countries = ['USA', 'France', 'Germany', 'India', 'Russia']
count = [63.76, 34.54, 45.11, 57.43, 60.54]
countries
['USA', 'France', 'Germany', 'India', 'Russia']
count
[63.76, 34.54, 45.11, 57.43, 60.54]
```

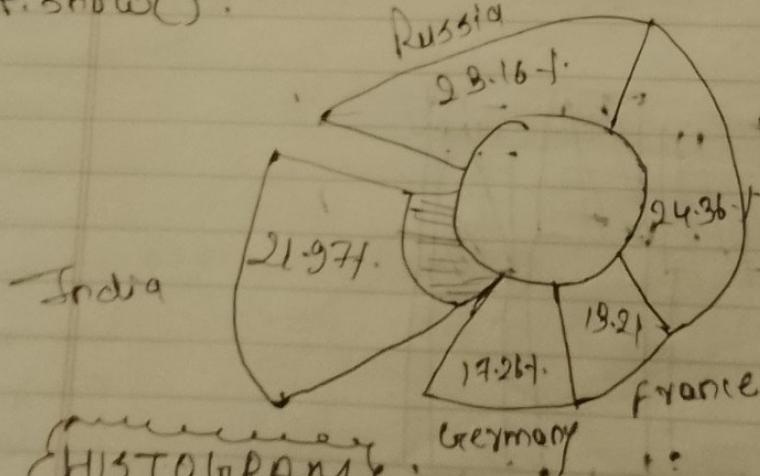
Make a Pie Plot

```
plt.pie(count, labels=countries, colors=['r','g','m','b','c'], radius=1.5,
        autopct='%.0f%%', counterclock=False, startangle=90,
        pctdistance=0.8, shadow=True, explode=[0,0,0,0.2,0])
plt.show()
```



DOUGHNUT CHART

```
plt.pie(count, labels=countries, colors=['r','g','m','b','c'], radius=1.5,
        autopct='%.2f%%', counterclock=False, startangle=90,
        pctdistance=0.8, shadow=True, explode=[0,0,0,0,0.2])
plt.pie([1], radius=0.7, colors=['w'])
plt.show()
```



HISTOGRAM

```
x = iris['sepal_width']
x5
```

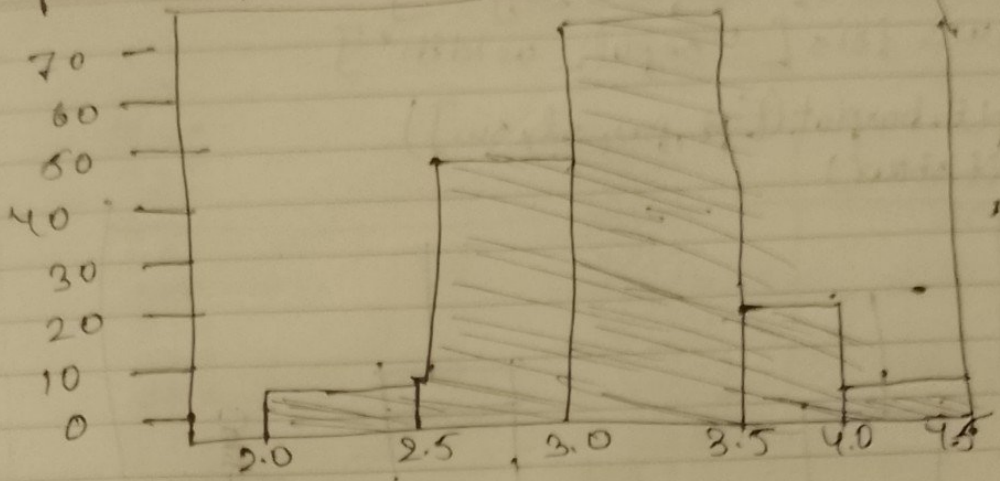
0	3.5
1	3.0
2	3.2
3	3.1
4	3.6
...	...
145	3.0
146	2.5
148	3.4
149	3.0

```
max(x)
4.4
min(x)
2.0
```

bin_ = [2, 2.5, 3, 3.5, 4, 4.5]

Make a histogram of sepal width

f = plt.hist(x, bins=bin_, edgecolor='k', color='y')



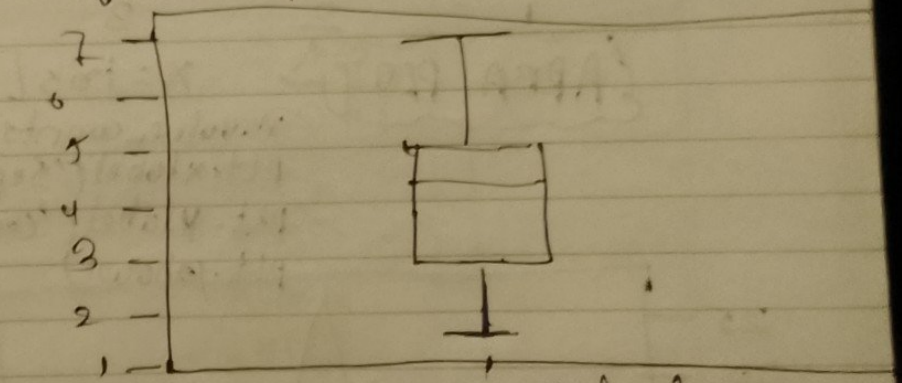
f[0] : array([11., 46., 68., 21., 4.])

{Box Plot}

x = iris['petal length']

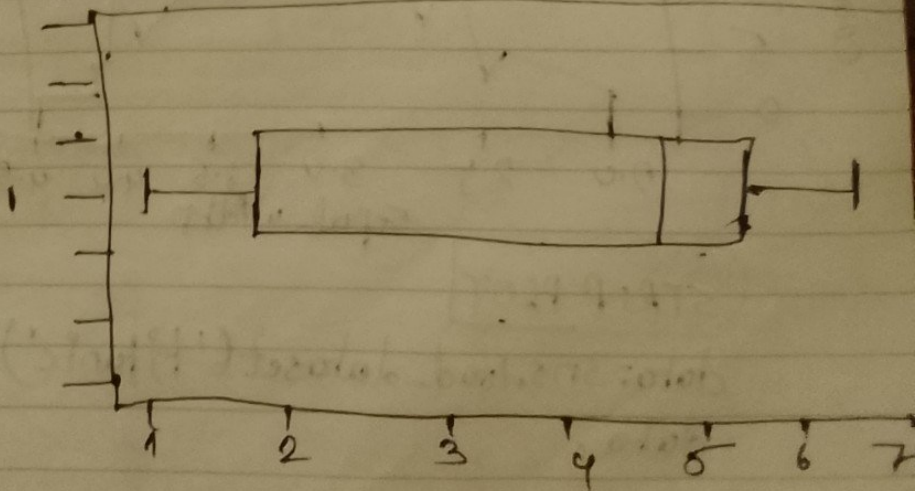
Create a box plot for iris petal length.

```
plt.boxplot(x)
plt.show()
```



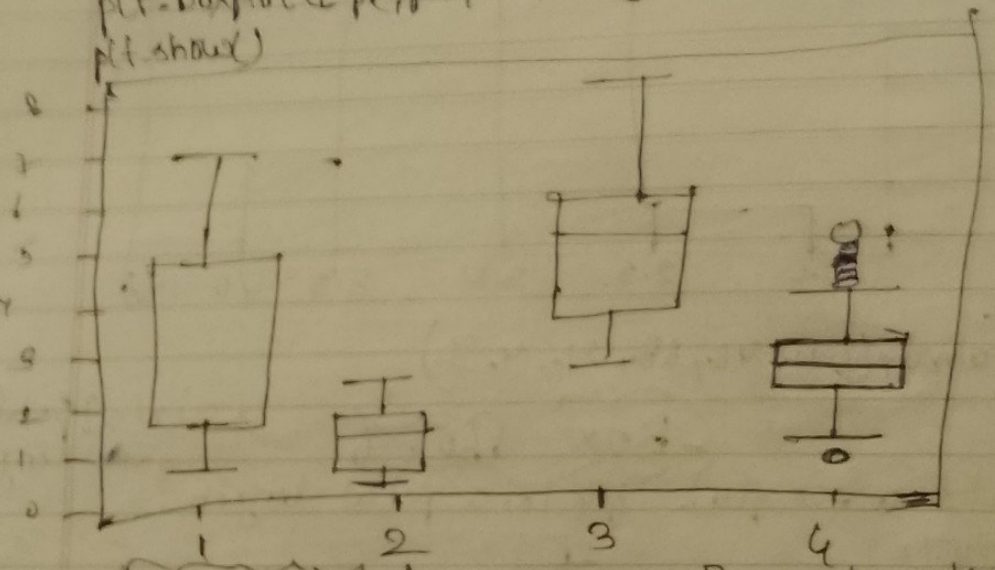
Create a horizontal box plot for iris 'petal length.

```
plt.boxplot(x, vert=False)
plt.show()
```



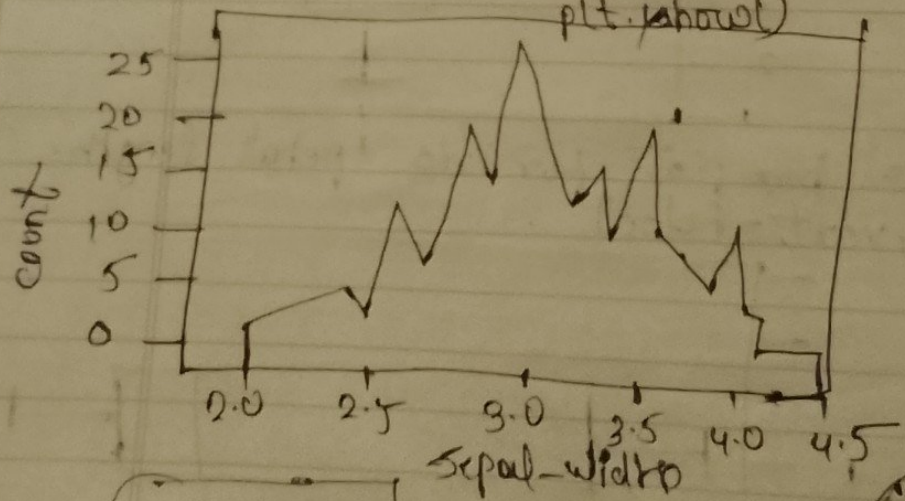
Create a box plot for iris dataset

```
pl = iris['petal_length']
pw = iris['petal_width']
sl = iris['sepal_length']
sw = iris['sepal_width']
plt.boxplot([pl, pw, sl, sw])
plt.show()
```



AREA PLOT

```
x = iris['sepal_width']
x.value_counts().sort_index().plot(kind=
plt.xlabel('Sepal_Width')
plt.xlabel('count')
plt.show()
```



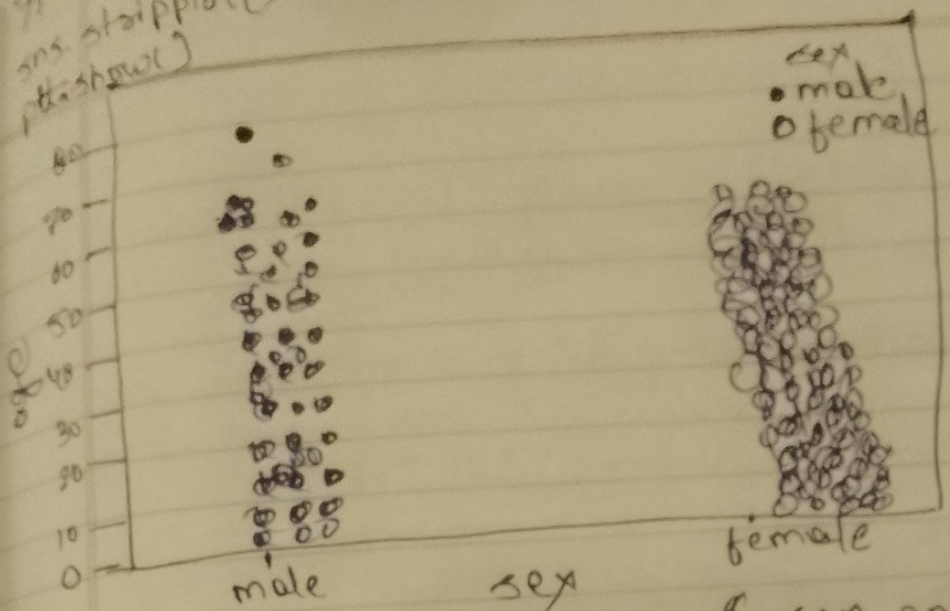
STRIP PLOT

SEABORN

```
data = sns.load_dataset('titanic')
data.
```

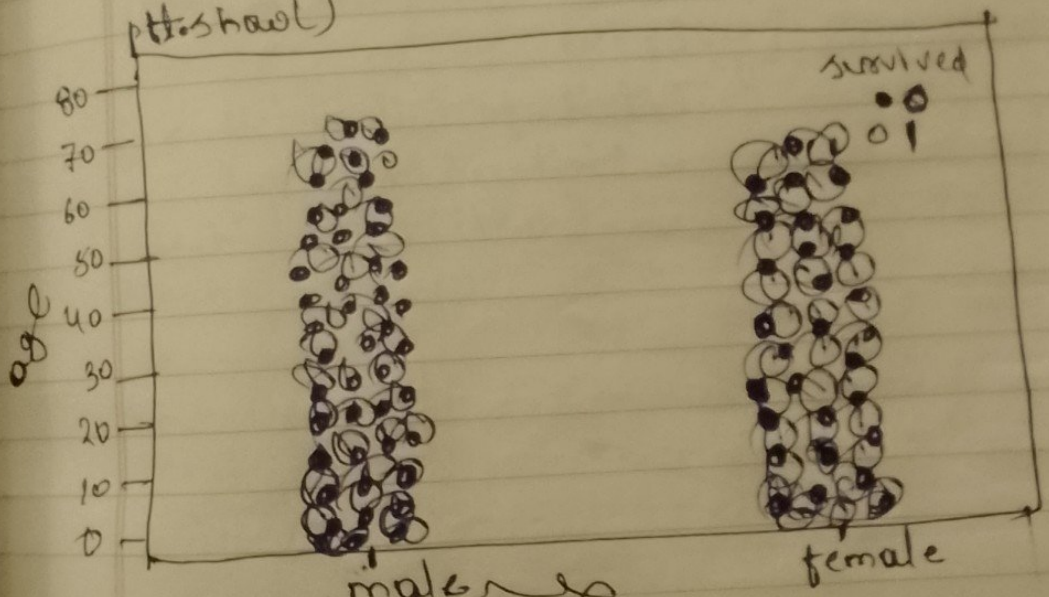
#Make a strip plot between age and sex

```
sns.stripplot(x=data['sex'], y=data['age'], color='y', hue=data['sex'])
plt.show()
```



Make a strip plot b/w age & sex and survived

```
sns.stripplot(x=data['sex'], y=data['age'], color='y', hue=data['survived'])
plt.show()
```



[VIOLIN PLOT]

```
sns.violinplot(x=data['sex'], y=data['age'], color='y',
hue=data['sex'])
plt.show
```