Data Science

Lab Record

Course Outcome 1

Submitted By : Jomin K Mathew
Department : MCA
Roll no : MCA321

COURSE OUTCOME 1.1

AIM:

Review of python programming – Programs review the fundamentals of python

PROGRAM:

Datatypes

```
#numbers
```

```
3+3 #addition
```

```
6
```

```
4-3 #subtraction
```

```
1
```

```
10*5  #multiplication
```

```
50
```

```
10/5   #divison
```

```
2.0
```

```
5**2  #power
```

```
25
```

```
8%2    #modulo function
```

```
0
```

Strings

```
'hello' #single quotes
```

```
'hello'
```

```
"hello world"  #double quotes
```

```
'hello world'
```

print

```
#variable assignmnet
x=22
y=20
z=x+y
print (z)
```

```
42
```

```
a= 'tanu'
b='manu'
print('my name is :{}, and my friend is :{}'.format(a,b))
```

```
my name is :tanu, and my friend is :manu
```

List

```
my_list=[1,2,3,4]
my_list.append(6)
my_list
```

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

```
my_list[3]
```

```
4
```

```
my_list[0:2]
```

```
[1, 2]
```

```
my_list[2:]
```

```
[3, 4, 6]
```

```
my_list[:2]
```

```
[1, 2]
```

```
my_list[1]= 34
my_list
```

```
[1, '34', 3, 4, 6]
```

Dictionary

```
d = {'key1':'item1','key2':'item2'}
d
```

```
{'key1': 'item1', 'key2': 'item2'}
```

```
d['key2']
```

```
'item2'
```

Comparison Operators

```
2>5
```

```
False
```

```
5>2
```

```
True
```

```
3 == 5
```

```
False
```

Tuples

```
t=(1,2,3)
t
```

```
(1, 2, 3)
```

```
t[1]
```

```
2
```

Sets

```
s={1,2,3,2,4,5,6,1,2,7}
s
```

```
{1, 2, 3, 4, 5, 6, 7}
```

Logic Operators

```
(1>2) or (2<3)
```

```
True
```

```
(3>4) and (4>5)
```

```
False
```

if else statements

```
if 2> 3:
  print("correct")
else:
   print('wrong')
```

```
wrong
```

```
if 1 == 2:
    print('first')
elif 2 == 2:
    print('second')
else:
    print('Last')
```

```
second
```

Loops

```
a=[1,2,3,4,5,6] #for loop
for i in a:
  print(i)
```

```
1
2
3
4
5
6
```

```
i=1          #while loop
while i<7:
  print('i is:{}'.format(i))
  i=i+1
```

```
i is:1
i is:2
i is:3
i is:4
i is:5
i is:6
```

Range

```
list(range(10))
```

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

```
for i in range(10):
  print(i)
```

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

Lambda

```
def a(var):
  return var**2
```

```
a(5)
```

```
25
```

functions

```
def my_func(param1='default'):

    print(param1)
```

```
my_func
```

```
<function __main__.my_func>
```

```
my_func()
```

```
default
```

```
def cube(x):
  print(x**3)
```

```
a=cube(8)
```

```
512
```

COURSE OUTCOME 1-2

AIM:

Matrix operations (using vectorization) and transformation using python and SVD using Python.

PROGRAM:

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

```
array([1, 2, 3, 4])
```

```
import numpy as np
list2=[[1,2,3],[2,1,3],[1,1,2]]
np.array(list2)
```

```
array([[1, 2, 3],
       [2, 1, 3],
       [1, 1, 2]])
```

```
import numpy as np
np.arange(0,11)
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

```
import numpy as np
np.zeros(3)
```

```
array([0., 0., 0.])
```

```
import numpy as np
np.zeros((5,5))
```

```
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

```
import numpy as np
np.ones((3,3))
```

```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

```
import numpy as np
np.eye(2)
```

```
array([[1., 0.],
       [0., 1.]])
```

```
import numpy as np
np.linspace(0,11,3)
```

```
array([ 0. ,  5.5, 11. ])
```

```
import numpy as np
np.linspace(0,10,100)
```

```
array([ 0.        ,  0.1010101 ,  0.2020202 ,  0.3030303 ,  0.4040404 ,
        0.50505051,  0.60606061,  0.70707071,  0.80808081,  0.90909091,
        1.01010101,  1.11111111,  1.21212121,  1.31313131,  1.41414141,
        1.51515152,  1.61616162,  1.71717172,  1.81818182,  1.91919192,
        2.02020202,  2.12121212,  2.22222222,  2.32323232,  2.42424242,
        2.52525253,  2.62626263,  2.72727273,  2.82828283,  2.92929293,
        3.03030303,  3.13131313,  3.23232323,  3.33333333,  3.43434343,
        3.53535354,  3.63636364,  3.73737374,  3.83838384,  3.93939394,
        4.04040404,  4.14141414,  4.24242424,  4.34343434,  4.44444444,
        4.54545455,  4.64646465,  4.74747475,  4.84848485,  4.94949495,
        5.05050505,  5.15151515,  5.25252525,  5.35353535,  5.45454545,
        5.55555556,  5.65656566,  5.75757576,  5.85858586,  5.95959596,
        6.06060606,  6.16161616,  6.26262626,  6.36363636,  6.46464646,
        6.56565657,  6.66666667,  6.76767677,  6.86868687,  6.96969697,
        7.07070707,  7.17171717,  7.27272727,  7.37373737,  7.47474747,
        7.57575758,  7.67676768,  7.77777778,  7.87878788,  7.97979798,
        8.08080808,  8.18181818,  8.28282828,  8.38383838,  8.48484848,
        8.58585859,  8.68686869,  8.78787879,  8.88888889,  8.98989899,
        9.09090909,  9.19191919,  9.29292929,  9.39393939,  9.49494949,
        9.5959596 ,  9.6969697 ,  9.7979798 ,  9.8989899 , 10.        ])
```

```
import numpy as np
np.random.rand(20)
```

```
array([0.86693426, 0.22154582, 0.92190843, 0.97621991, 0.2607321 ,
       0.79181996, 0.28109623, 0.75633686, 0.2931279 , 0.45279405,
       0.72936983, 0.08791871, 0.08975821, 0.26870815, 0.20210111,
       0.65894863, 0.57489058, 0.87338686, 0.75801352, 0.82038037])
```

```
import numpy as np
np.random.rand(5,5)
```

```
array([[0.4578422 , 0.45407171, 0.14037027, 0.20996102, 0.65799849],
       [0.89489157, 0.27252678, 0.14044572, 0.95375516, 0.34507616],
       [0.85023199, 0.80579089, 0.97009992, 0.62703003, 0.20291678],
       [0.58278451, 0.68772279, 0.5908991 , 0.5129685 , 0.71650913],
       [0.95386864, 0.59909619, 0.38565939, 0.22834402, 0.21237599]])
```

```
import numpy as np
np.random.randn(7,5)
```

```
array([[ 0.01686955,  0.39993997, -1.46144873, -0.92948814,  0.9691076 ],
       [-0.1188221 , -0.01427539, -0.58388294, -2.07453237,  0.09995917],
       [ 0.70871901, -0.29827147, -1.25963057,  0.13932838,  0.11790414],
       [ 1.19973859, -1.9807397 , -0.77545743,  0.46533678, -1.14334568],
       [ 1.11421291, -2.06236284,  1.01947782, -0.74464748, -0.64964145],
       [-1.45402831,  0.37458063, -1.11946822,  0.43936759, -1.45837857],
       [ 0.87746297, -0.21431929,  0.41909028, -0.70797156, -0.00442427]])
```

```
import numpy as np
np.random.randint(1,100)
```

```
67
```

```
np.arange(1,25)
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24])
```

```
np.random.randint(1,100,11)
```

```
array([48, 21, 63, 97, 31, 96, 88, 17, 64, 52, 73])
```

```
import numpy as np
a=np.random.rand(20)
a.reshape(5,4)
```

```
array([[0.51986833, 0.37150064, 0.2337934 , 0.53493424],
       [0.14982039, 0.85672271, 0.01099853, 0.97762309],
       [0.51471568, 0.29009164, 0.15862369, 0.0304231 ],
       [0.15803882, 0.48553675, 0.7518431 , 0.74513697],
       [0.35614582, 0.43030297, 0.96768973, 0.3236649 ]])
```

```
a.max()
```

```
0.9776230941885315
```

```
a.min()
```

```
0.010998526602094327
```

```
a.shape
```

```
(20,)
```

```
a.argmax()
```

```
7
```

```
a.argmin()
```

```
6
```

```
a.reshape(20,1).shape
```

```
(20, 1)
```

```
a.reshape(20,1)
```

```
array([[0.51986833],
       [0.37150064],
       [0.2337934 ],
       [0.53493424],
       [0.14982039],
       [0.85672271],
       [0.01099853],
       [0.97762309],
       [0.51471568],
       [0.29009164],
       [0.15862369],
       [0.0304231 ],
       [0.15803882],
       [0.48553675],
       [0.7518431 ],
       [0.74513697],
       [0.35614582],
       [0.43030297],
       [0.96768973],
       [0.3236649 ]])
```

```
a.dtype
```

```
dtype('float64')
```

```
a[3]
```

```
0.5349342392889179
```

```
arr=np.arange(25)
```

```
arr
```

```
array([ 2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
       19, 20, 21, 22, 23, 24])
```

```

np.array(b)
```

```
array([4, 6, 1, 2, 3])
```

```
arr
```

```
array([ 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])
```

```
arr.reshape(5,5)
```

```
array([[ 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]])
```

```
arr[0:5]=100
```

```
arr
```

```
array([100, 100, 100, 100, 100,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24])
```

```
arr.reshape(5,5)
```

```
array([[100, 100, 100, 100, 100],
       [  5,   6,   7,   8,   9],
       [ 10,  11,  12,  13,  14],
       [ 15,  16,  17,  18,  19],
       [ 20,  21,  22,  23,  24]])
```

```
arr
```

```
array([100, 100, 100, 100, 100,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24])
```

```
arr=np.random.randint(1,100,25).reshape(5,5)
arr[0:2,0:2]=6
arr
```

```
array([[ 6,  6, 32, 65, 85],
       [ 6,  6, 37, 18, 87],
       [51, 23, 63, 34,  1],
       [99, 53, 89, 56, 67],
       [52, 76, 50, 32, 61]])
```

```
arr[1,:]
```

```
array([ 6,  6, 37, 18, 87])
```

```
arr[1:3,4]
```

```
array([87,  1])
```

```
arr >7
```

```
array([[False, False,  True,  True,  True],
       [False, False,  True,  True,  True],
       [ True,  True,  True,  True, False],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True]])
```

```
arr
```

```
array([[ 6,  6, 32, 65, 85],
       [ 6,  6, 37, 18, 87],
       [51, 23, 63, 34,  1],
       [99, 53, 89, 56, 67],
       [52, 76, 50, 32, 61]])
```

```
arr[arr >7]
```

```
array([32, 65, 85, 37, 18, 87, 51, 23, 63, 34, 99, 53, 89, 56, 67, 52, 76,
       50, 32, 61])
```

Double-click (or enter) to edit

```
import numpy as np
np.arange(10)
```

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

numpy arithmetic operations

```
n= np.arange(10)
n
```

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

```
a=n+n
a
```

```
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])
```

```
b=n-n
b
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
m=n*n
m
```

```
array([ 0,  1,  4,  9, 16, 25, 36, 49, 64, 81])
```

```
d=n/n
d
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: RuntimeWarning: invalid
  """Entry point for launching an IPython kernel.
array([nan,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.])
```

```
import numpy as np
np.zeros(10)
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

```
np.ones(10)
```

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

```
n=np.ones(10)
n
```

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

```
f=n*5
f
```

```
array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])
```

```
a=np.arange(10,51)
a
```

```
array([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])
```

Double-click (or enter) to edit

```
c=np.arange(9)
c.reshape(3,3)
```

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

Double-click (or enter) to edit

```
np.eye(3)
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

generate a random number between 0 and 1

```
np.random.rand(1)
```

```
array([0.26472847])
```

use numpy to generate an array of 25 random numbers sampled from a standard normal distribution

```
a=np.random.randn(25)
a
```

```
array([ 0.48457468, -0.69343187, -0.05088668,  0.05397066, -0.08924475,
       -1.58337588,  0.7412226 ,  0.76586551,  0.16132974,  0.81434727,
       -0.76950699,  0.5417698 ,  1.6265055 , -1.27285384, -0.64963137,
       -1.73651043,  0.59499325,  0.55428232,  0.56223966, -0.29924504,
       -0.86134432,  0.4188935 , -0.93878007, -0.1952214 ,  0.47361981])
```

```
a.reshape(5,5)
```

```
array([[ 0.48457468, -0.69343187, -0.05088668,  0.05397066, -0.08924475],
       [-1.58337588,  0.7412226 ,  0.76586551,  0.16132974,  0.81434727],
       [-0.76950699,  0.5417698 ,  1.6265055 , -1.27285384, -0.64963137],
       [-1.73651043,  0.59499325,  0.55428232,  0.56223966, -0.29924504],
       [-0.86134432,  0.4188935 , -0.93878007, -0.1952214 ,  0.47361981]])
```

create the following matrix the number should in range between o and 1

a:using linspace

```
z=np.linspace(0,1,100)
z
```

```
array([0.        , 0.01010101, 0.02020202, 0.03030303, 0.04040404,
       0.05050505, 0.06060606, 0.07070707, 0.08080808, 0.09090909,
       0.1010101 , 0.11111111, 0.12121212, 0.13131313, 0.14141414,
       0.15151515, 0.16161616, 0.17171717, 0.18181818, 0.19191919,
       0.2020202 , 0.21212121, 0.22222222, 0.23232323, 0.24242424,
       0.25252525, 0.26262626, 0.27272727, 0.28282828, 0.29292929,
       0.3030303 , 0.31313131, 0.32323232, 0.33333333, 0.34343434,
       0.35353535, 0.36363636, 0.37373737, 0.38383838, 0.39393939,
       0.4040404 , 0.41414141, 0.42424242, 0.43434343, 0.44444444,
       0.45454545, 0.46464646, 0.47474747, 0.48484848, 0.49494949,
       0.50505051, 0.51515152, 0.52525253, 0.53535354, 0.54545455,
       0.55555556, 0.56565657, 0.57575758, 0.58585859, 0.5959596 ,
       0.60606061, 0.61616162, 0.62626263, 0.63636364, 0.64646465,
       0.65656566, 0.66666667, 0.67676768, 0.68686869, 0.6969697 ,
       0.70707071, 0.71717172, 0.72727273, 0.73737374, 0.74747475,
       0.75757576, 0.76767677, 0.77777778, 0.78787879, 0.7979798 ,
       0.80808081, 0.81818182, 0.82828283, 0.83838384, 0.84848485,
       0.85858586, 0.86868687, 0.87878788, 0.88888889, 0.8989899 ,
       0.90909091, 0.91919192, 0.92929293, 0.93939394, 0.94949495,
       0.95959596, 0.96969697, 0.97979798, 0.98989899, 1.        ])
```

```
z.reshape(10,10)
```

```
array([[0.        , 0.01010101, 0.02020202, 0.03030303, 0.04040404,
        0.05050505, 0.06060606, 0.07070707, 0.08080808, 0.09090909],
       [0.1010101 , 0.11111111, 0.12121212, 0.13131313, 0.14141414,
        0.15151515, 0.16161616, 0.17171717, 0.18181818, 0.19191919],
       [0.2020202 , 0.21212121, 0.22222222, 0.23232323, 0.24242424,
        0.25252525, 0.26262626, 0.27272727, 0.28282828, 0.29292929],
       [0.3030303 , 0.31313131, 0.32323232, 0.33333333, 0.34343434,
        0.35353535, 0.36363636, 0.37373737, 0.38383838, 0.39393939],
       [0.4040404 , 0.41414141, 0.42424242, 0.43434343, 0.44444444,
        0.45454545, 0.46464646, 0.47474747, 0.48484848, 0.49494949],
       [0.50505051, 0.51515152, 0.52525253, 0.53535354, 0.54545455,
        0.55555556, 0.56565657, 0.57575758, 0.58585859, 0.5959596 ],
       [0.60606061, 0.61616162, 0.62626263, 0.63636364, 0.64646465,
        0.65656566, 0.66666667, 0.67676768, 0.68686869, 0.6969697 ],
       [0.70707071, 0.71717172, 0.72727273, 0.73737374, 0.74747475,
        0.75757576, 0.76767677, 0.77777778, 0.78787879, 0.7979798 ],
       [0.80808081, 0.81818182, 0.82828283, 0.83838384, 0.84848485,
        0.85858586, 0.86868687, 0.87878788, 0.88888889, 0.8989899 ],
       [0.90909091, 0.91919192, 0.92929293, 0.93939394, 0.94949495,
        0.95959596, 0.96969697, 0.97979798, 0.98989899, 1.        ]])
```

b:reshape and divison

```
s=np.arange(1,101)
s
```

```
array([  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])
```

Double-click (or enter) to edit

```
a=s/100
a
```

```
array([0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 , 0.11,
       0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 , 0.21, 0.22,
       0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 , 0.31, 0.32, 0.33,
       0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 , 0.41, 0.42, 0.43, 0.44,
       0.45, 0.46, 0.47, 0.48, 0.49, 0.5 , 0.51, 0.52, 0.53, 0.54, 0.55,
       0.56, 0.57, 0.58, 0.59, 0.6 , 0.61, 0.62, 0.63, 0.64, 0.65, 0.66,
       0.67, 0.68, 0.69, 0.7 , 0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77,
       0.78, 0.79, 0.8 , 0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88,
       0.89, 0.9 , 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99,
       1.  ])
```

```
a.reshape(10,10)
```

```
array([[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 ],
       [0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 ],
       [0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 ],
       [0.31, 0.32, 0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 ],
       [0.41, 0.42, 0.43, 0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5 ],
       [0.51, 0.52, 0.53, 0.54, 0.55, 0.56, 0.57, 0.58, 0.59, 0.6 ],
       [0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67, 0.68, 0.69, 0.7 ],
       [0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77, 0.78, 0.79, 0.8 ],
       [0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88, 0.89, 0.9 ],
       [0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1.  ]])
```

create and array of 20 lineraly spaced points between 0 and 1

```
w=np.linspace(0,1,20)
w
```

```
array([0.        , 0.05263158, 0.10526316, 0.15789474, 0.21052632,
       0.26315789, 0.31578947, 0.36842105, 0.42105263, 0.47368421,
       0.52631579, 0.57894737, 0.63157895, 0.68421053, 0.73684211,
       0.78947368, 0.84210526, 0.89473684, 0.94736842, 1.        ])
```

write code based on question below be careful not to run the cell below ,otherwise you wont be able to

```
mat=np.arange(1,26).reshape(5,5)
mat
```

```
array([[ 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]])
```

```
mat[2:,1:]
```

```
array([[12, 13, 14, 15],
       [17, 18, 19, 20],
       [22, 23, 24, 25]])
```

select 20 from matrix

```
mat[3,4]
```

```
20
```

```
j=mat[0:3,1].reshape(3,1)
j
```

```
array([[ 2],
       [ 7],
       [12]])
```

OR

```
mat[:3,1:2]
```

```
array([[ 2],
       [ 7],
       [12]])
```

```
mat[4,:]
```

```
array([21, 22, 23, 24, 25])
```

```
mat[3:,0:]
```

```
array([[16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

get the sum of all the values to the matrix

```
mat.sum()
```

```
325
```

get the standard deviation of the values in mat

```
mat.std()
```

```
7.211102550927978
```

get the sum of all the coloumns in mat

```
mat[:,0].sum()
```

```
55
```

SVD

```
import numpy as np
from numpy.linalg import svd
```

```
A = np.array([[1,2,3], [4,5,6], [5,7,9]])
U, S, VT = svd(A)
```

```
U
```

```
array([[-0.2354116 ,  0.78182354, -0.57735027],
       [-0.55937325, -0.5947842 , -0.57735027],
       [-0.79478485,  0.18703934,  0.57735027]])
```

```
S
```

```
array([1.56633231e+01, 8.12593979e-01, 1.13716384e-15])
```

```
VT
```

```
array([[-0.41158755, -0.56381288, -0.71603821],
       [-0.8148184 , -0.12429146,  0.56623547],
       [-0.40824829,  0.81649658, -0.40824829]])
```

```
a=(U @ np.diag(S) @ VT)
```

```
a
```

```
array([[1., 2., 3.],
       [4., 5., 6.],
       [5., 7., 9.]])
```

COURSE OUTCOME 1-3

AIM:

Programs using matplotlib / plotly / bokeh / seaborn for data visualisation

PROGRAM:

```
import matplotlib.pyplot as plt

import numpy as np
x=np.linspace(0,5,11)
y=x**2
x
```

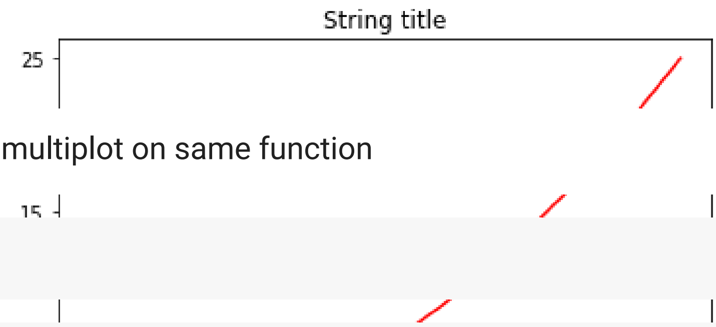
```
array([0. , 0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5, 5. ])
```

```
y
```

```
array([ 0.  ,  0.25,  1.  ,  2.25,  4.  ,  6.25,  9.  , 12.25, 16.  ,
       20.25, 25.  ])
```

basic matplotlib commands

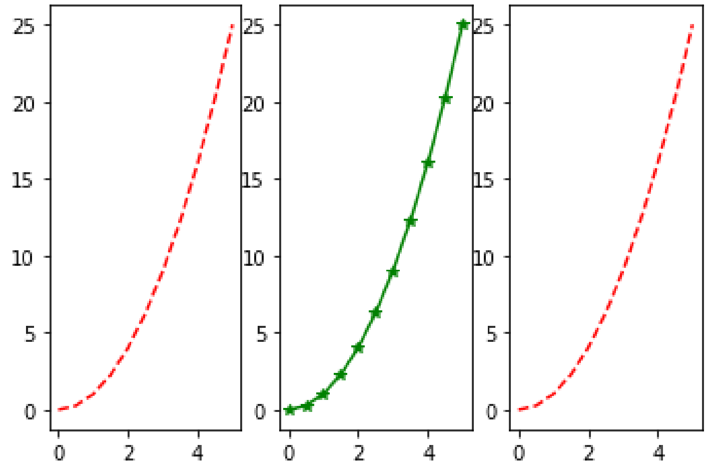
```
plt.plot(x,y,'r')
plt.xlabel('X Axis')
plt.ylabel('y Axis')
plt.title('String title')
plt.show()
```



create multiplot on same function

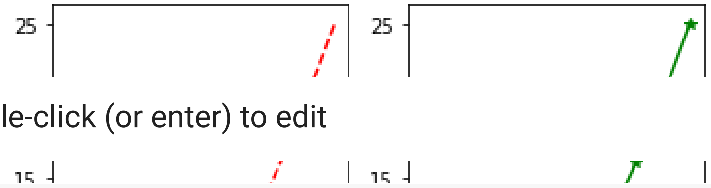
```
#plt.subplot(nrows,ncol,plot_number)
plt.subplot(1,3,1)
plt.plot(x,y,'r--')
plt.subplot(1,3,2)
plt.plot(x,y,'g*-')
plt.subplot(1,3,3)
plt.plot(x,y,'r--')
```

```
[<matplotlib.lines.Line2D at 0x7f3860707690>]
```



```
plt.subplot(1,2,1)
plt.plot(x,y,'r--')
plt.subplot(1,2,2)
plt.plot(x,y,'g*-')
```

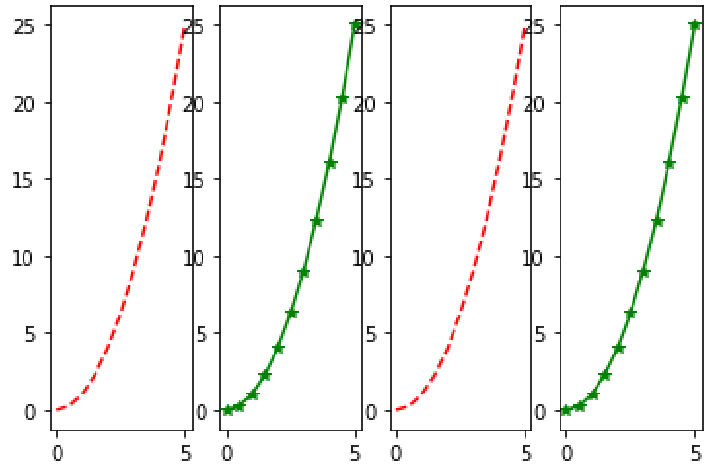
```
[<matplotlib.lines.Line2D at 0x7f38605e24d0>]
```



Double-click (or enter) to edit

```
plt.subplot(1,4,1)
plt.plot(x,y,'r--')
plt.subplot(1,4,2)
plt.plot(x,y,'g*-')
plt.subplot(1,4,3)
plt.plot(x,y,'r--')
plt.subplot(1,4,4)
plt.plot(x,y,'g*-')
```

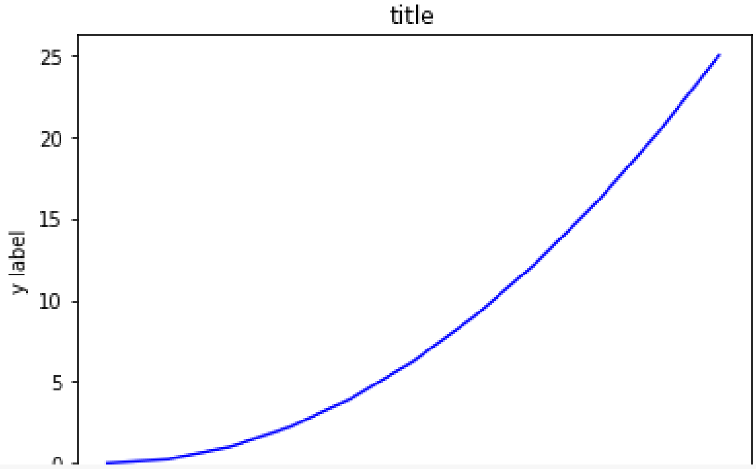
```
[<matplotlib.lines.Line2D at 0x7f386049abd0>]
```



matplot using object oriented

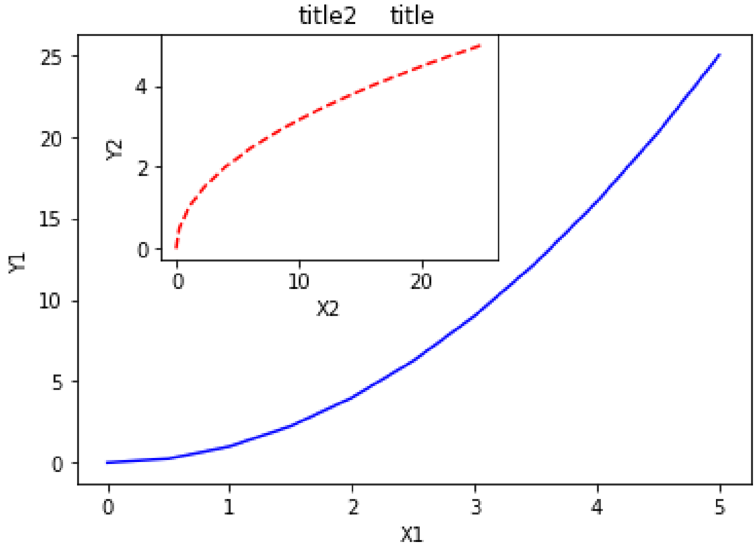
```
#create figure(empty canvas)
fig=plt.figure()

#add set of axes to figure
axes = fig.add_axes([0.1,0.1,0.8,0.8])    #(range 0 to 1)
axes.plot(x,y,'b')
axes.set_xlabel('x label')
axes.set_ylabel('y label')
axes.set_title('title')
```

```
Text(0.5, 1.0, 'title')
```



```
fig=plt.figure()

axes1 = fig.add_axes([0.1,0.1,0.8,0.8])    #main
axes2 =fig.add_axes([.2,.5,.4,.4])

axes1.plot(x,y,'b')
axes1.set_xlabel('X1')
axes1.set_ylabel('Y1')
axes1.set_title('title')

axes2.plot(y,x,'r--')
axes2.set_xlabel('X2')
axes2.set_ylabel('Y2')
axes2.set_title('title2')
```

```
Text(0.5, 1.0, 'title2')
```

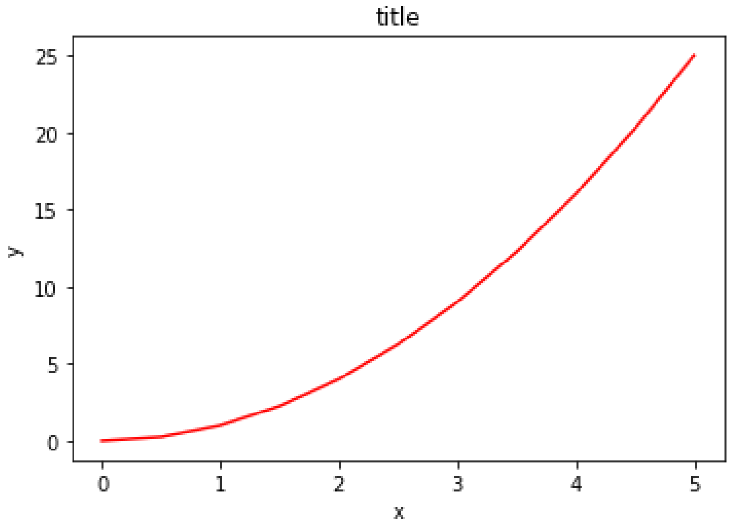


```
axes
```

```
<matplotlib.axes._axes.Axes at 0x7f38604af6d0>
```

plt.subplots() object

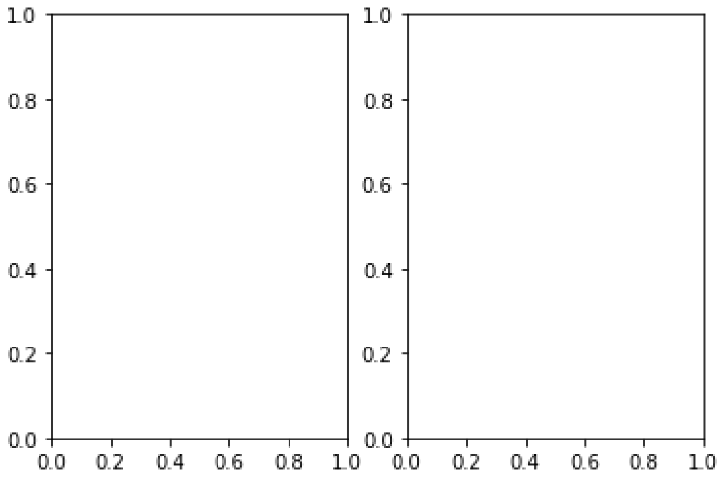
```
fig, axes= plt.subplots()
axes.plot(x,y,'r')
axes.set_xlabel('x')
axes.set_ylabel('y')
axes.set_title('title');
```



```
axes
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f386113a690>
```

```
#call subplots
fig,axes= plt.subplots(1,2)
```

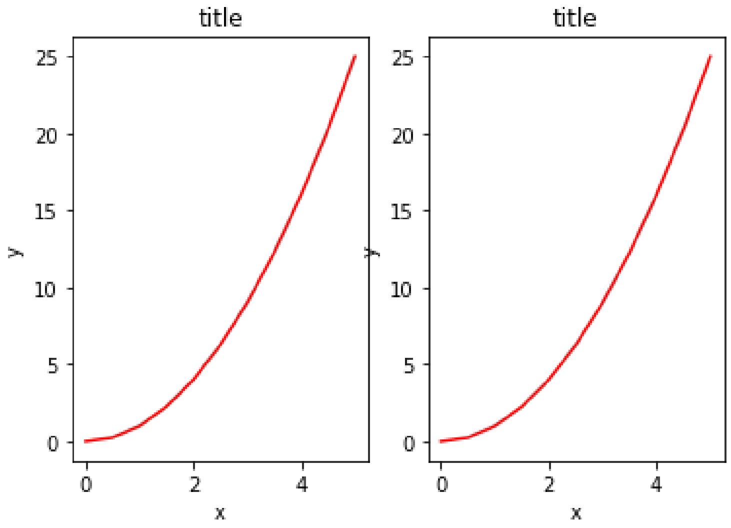


```
axes
```

```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f3860482bd0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f3860404990>],
      dtype=object)
```

```
for ax in axes:
  ax.plot(x,y,'r')
  ax.set_xlabel('x')
  ax.set_ylabel('y')
  ax.set_title('title')
```

```
fig
```



```
fig.tight_layout();
```

```
fig
```

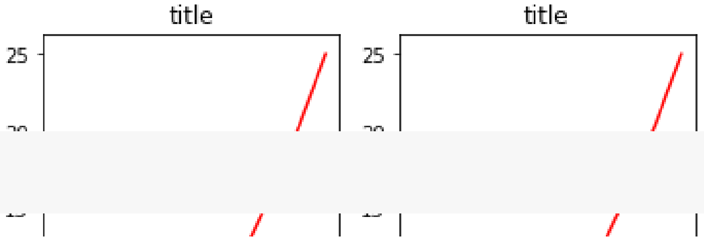
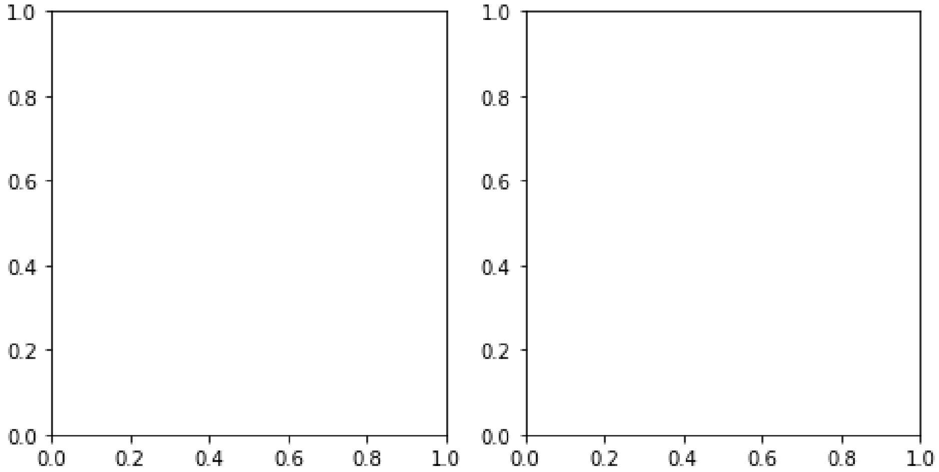


figure ratio,aspect ratio

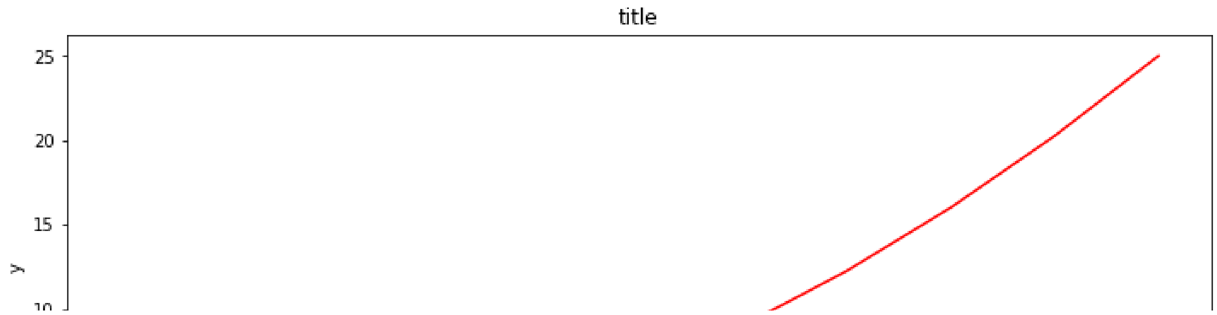
```
fig=plt.figure(figsize=(8,4), dpi=100)
```

```
<Figure size 800x400 with 0 Axes>
```

```
fig,axes =plt.subplots(1,2,figsize=(8,4))
```



```
fig, axes= plt.subplots(figsize=(12,5))
axes.plot(x,y,'r')
axes.set_xlabel('x')
axes.set_ylabel('y')
axes.set_title('title');
```



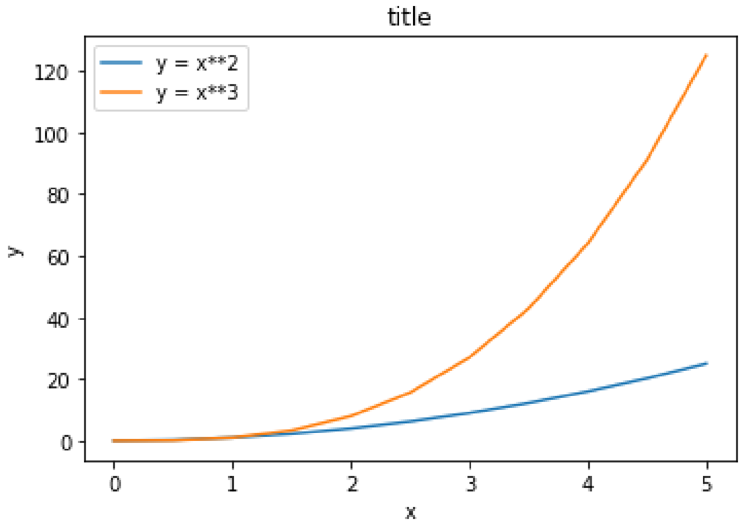
SAVING FIGURES

```
fig.savefig("filename.png")
```

```
fig.savefig("filename.png",dpi=200)
```

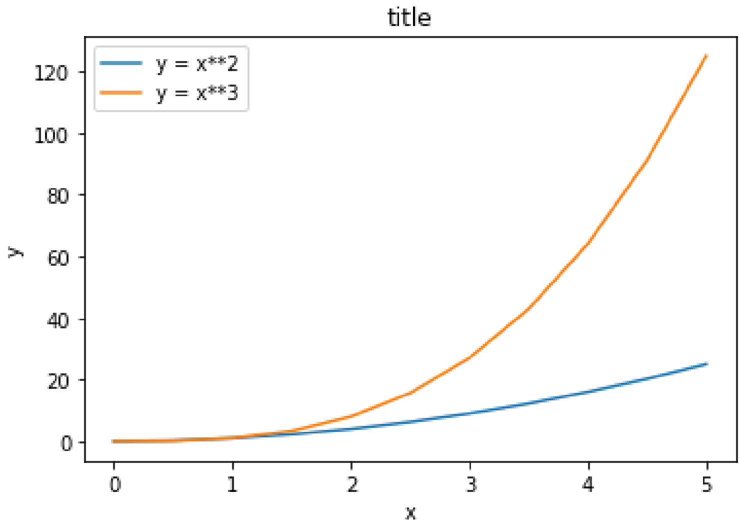
LEGENDS,LABELS AND TITLES

```
fig, ax = plt.subplots()

ax.plot(x, x**2, label="y = x**2")
ax.plot(x, x**3, label="y = x**3")
ax.legend(loc=2); # upper left corner
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_title('title');
```



```
import matplotlib.pyplot as plt
```

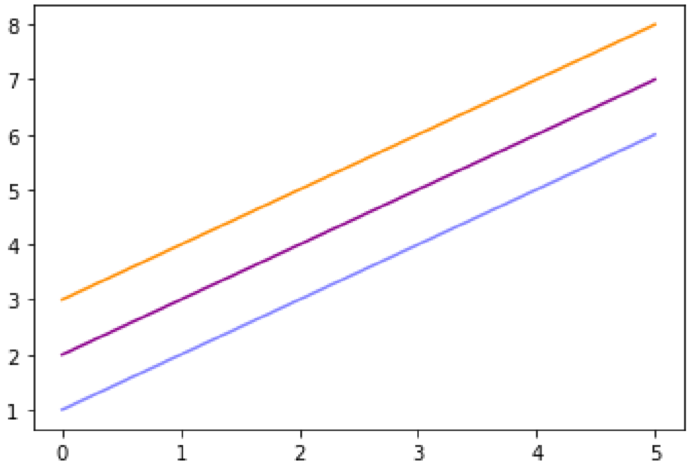
```
fig, ax = plt.subplots()

ax.plot(x, x**2, label="y = x**2")
ax.plot(x, x**3, label="y = x**3")
ax.legend(loc=2);
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_title('title');
```



SETTING COLORS,LINEWIDTH AND LINETYPES

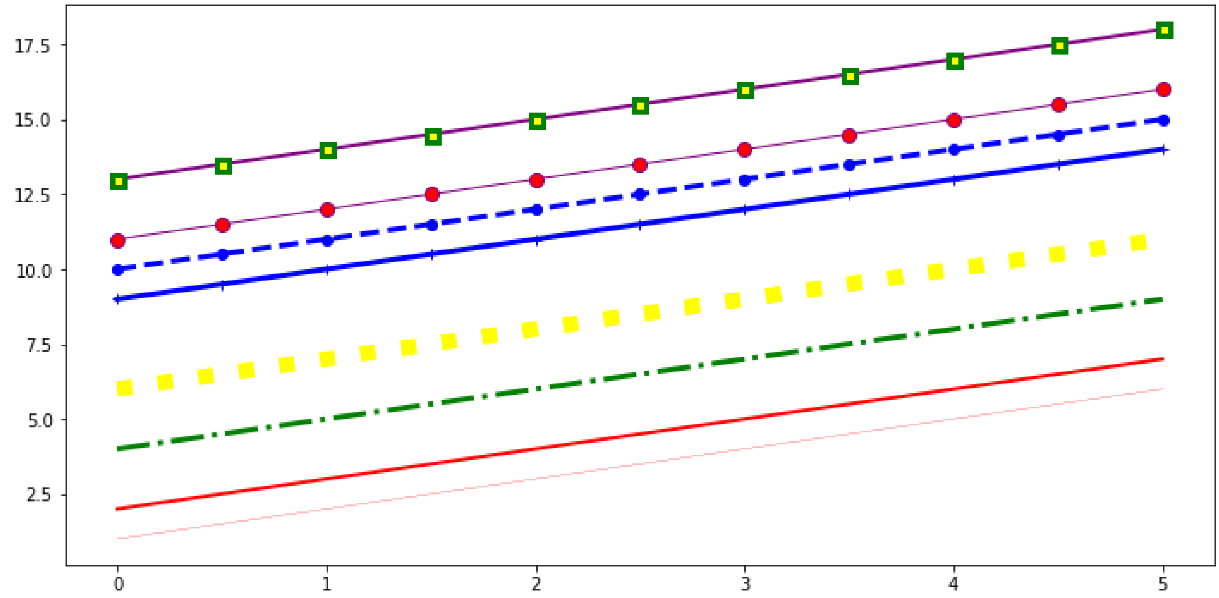
```
import matplotlib.pyplot as plt
fig, ax = plt. subplots()
ax.plot(x, x+1, color="blue", alpha=0.5)
ax.plot(x, x+2, color="#8B008B")
ax.plot(x, x+3, color="#FF8C00")
```

```
[<matplotlib.lines.Line2D at 0x7f38601036d0>]
```



line and marker style

```
fig, ax = plt.subplots(figsize=(12,6))

ax.plot(x, x+1, color="red", linewidth=0.25)
ax.plot(x, x+2, color="red", linewidth=2.00)

# possible linestype options ‘-‘, ‘–’, ‘-.’, ‘:’, ‘steps’
ax.plot(x, x+4, color="green", lw=3, ls='-.')
ax.plot(x, x+6, color="yellow", lw=9, ls=':')

# possible marker symbols: marker = '+', 'o', '*', 's', ',', '.', '1', '2', '
ax.plot(x, x+ 9, color="blue", lw=3, ls='-', marker='+')
ax.plot(x, x+10, color="blue", lw=3, ls='--', marker='o')

# marker size and color
ax.plot(x, x+11, color="purple", lw=1, ls='-', marker='o', markersize=8, mark
ax.plot(x, x+13, color="purple", lw=2, ls='-', marker='s', markersize=7, mark
```



DISTRIBUTION PLOTS

distplot,joinplot,pairpolot,kdeplot are some of the plots that allow as to visualize the distribution of data set

```
import seaborn as sns
%matplotlib inline
```

```
tips=sns.load_dataset('tips')
```

```
tips.head()
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

DISPLOT: shows the distribution of a univariate set of observation

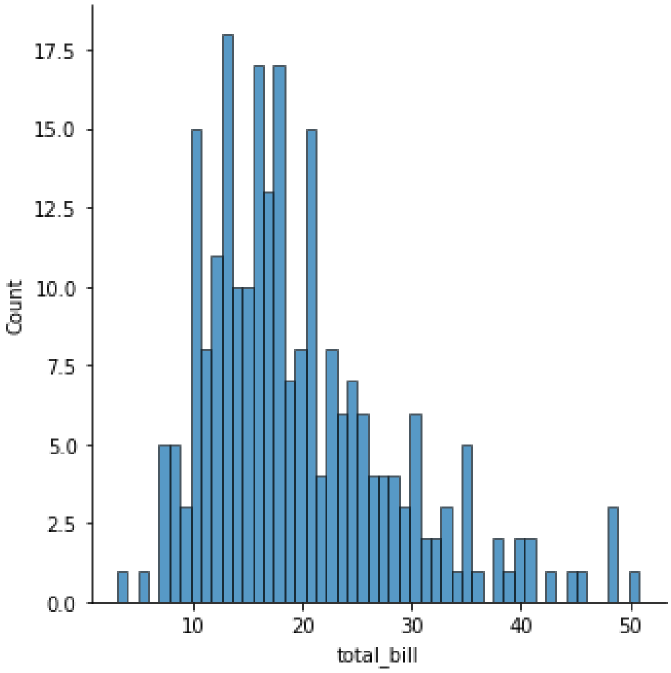
```
sns.distplot(tips['total_bill'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarnin
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7fc552bc8c90>
```

to remove kde layer and just have the histogram use the following syntax

```
sns.displot(tips['total_bill'],kde=False,bins=50)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fc5420f4b50>
```

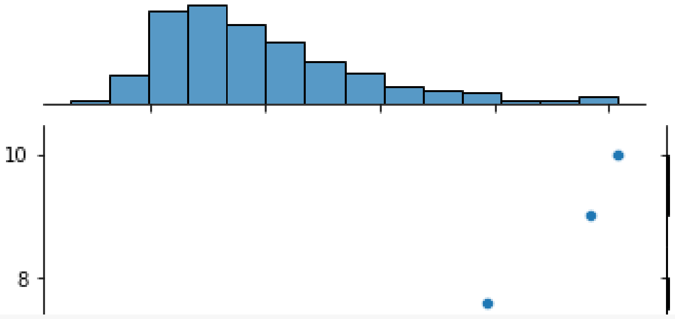


JOINTPLOT

jointplot allows to basically match up two displots for bivariate data scatter,reg,resid,kde,hex

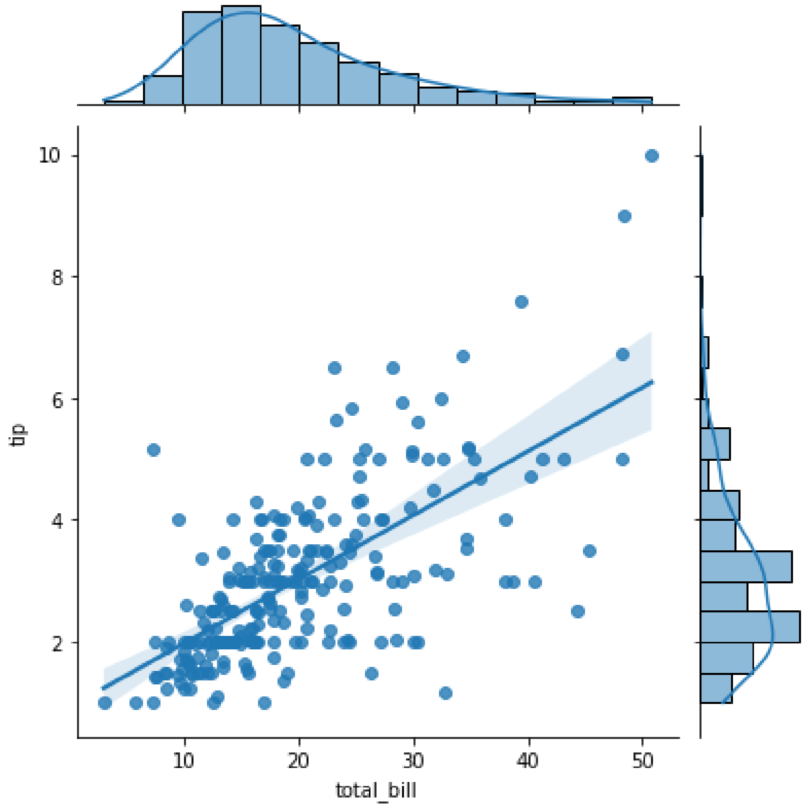
```
sns.jointplot(x='total_bill',y='tip',data=tips,kind='scatter')
```

```
<seaborn.axisgrid.JointGrid at 0x7fc5420c6c10>
```



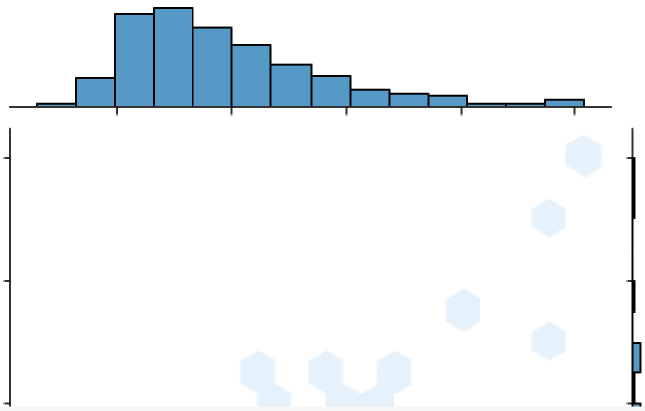
```
sns.jointplot(x='total_bill',y='tip',data=tips,kind='reg')
```

```
<seaborn.axisgrid.JointGrid at 0x7fc541f082d0>
```



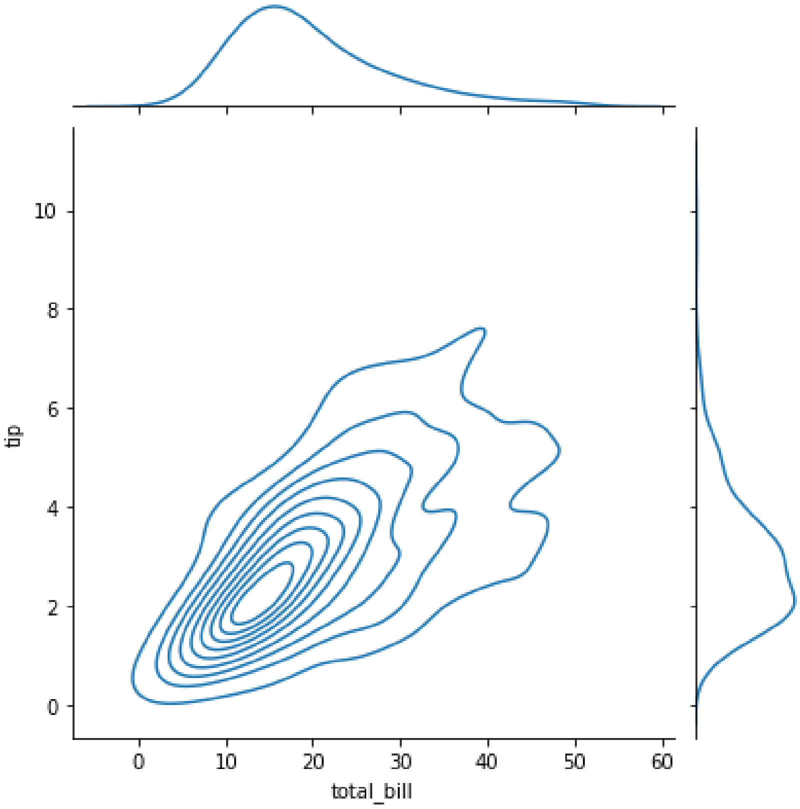
```
sns.jointplot(x='total_bill',y='tip',data=tips,kind='hex')
```

```
<seaborn.axisgrid.JointGrid at 0x7fc541cab790>
```



```
sns.jointplot(x='total_bill',y='tip',data=tips,kind='kde')
```

```
<seaborn.axisgrid.JointGrid at 0x7fc541a4c110>
```

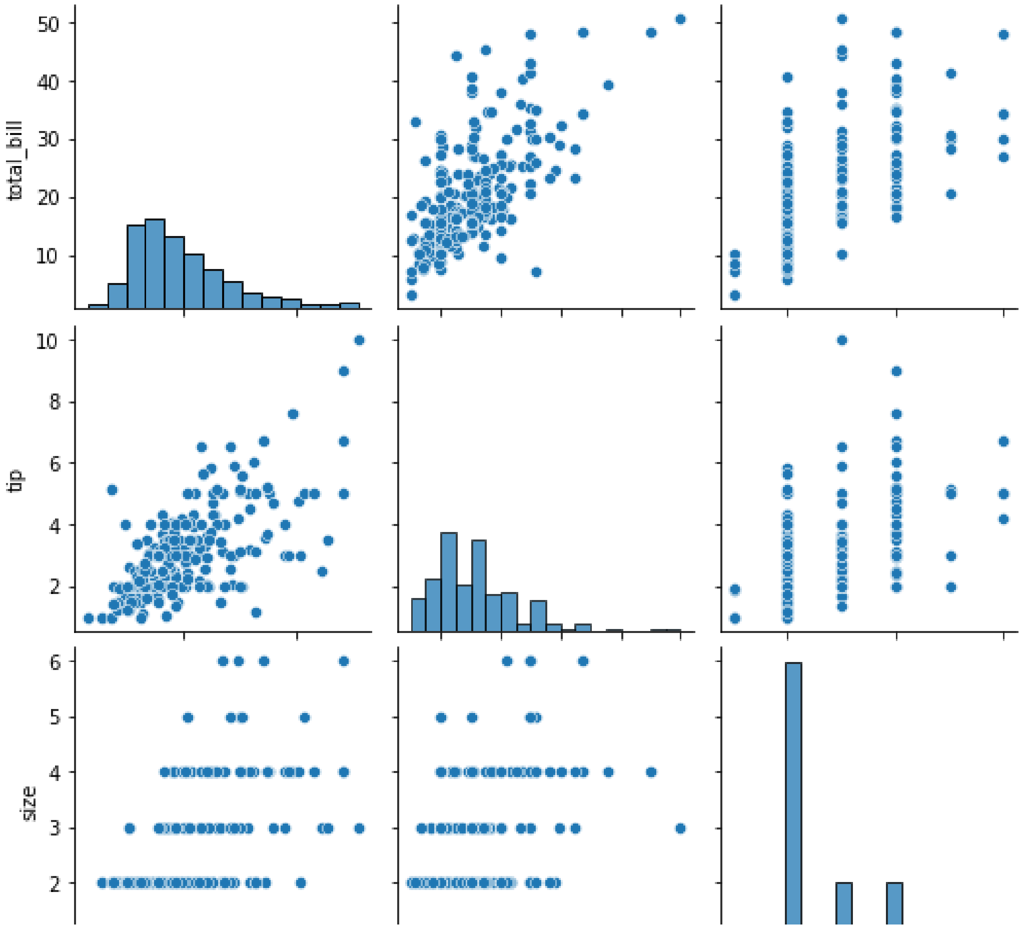


PAIRPLOT

pairwise relationships across an entire dataframe

```
sns.pairplot(tips)
```

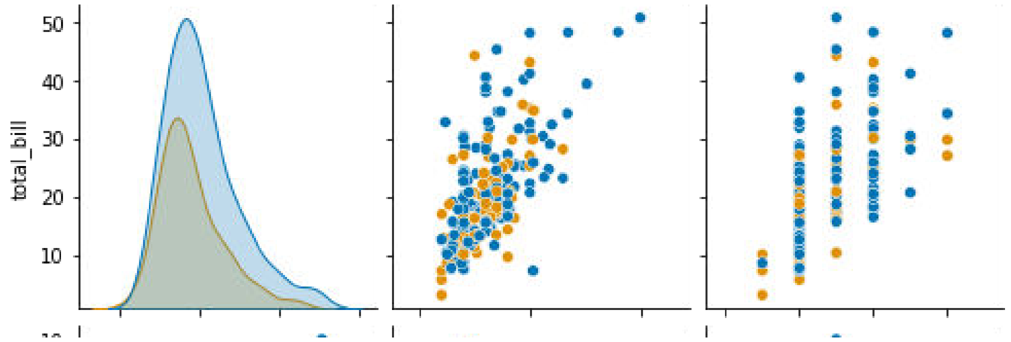
```
<seaborn.axisgrid.PairGrid at 0x7fc541955a90>
```



palette color :deep , muted , pastel , bright , dark,colorblind,coolwarm

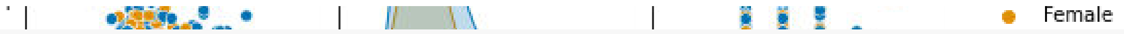
```
sns.pairplot(tips,hue='sex',palette='colorblind')
```

```
<seaborn.axisgrid.PairGrid at 0x7fc541ac1a10>
```



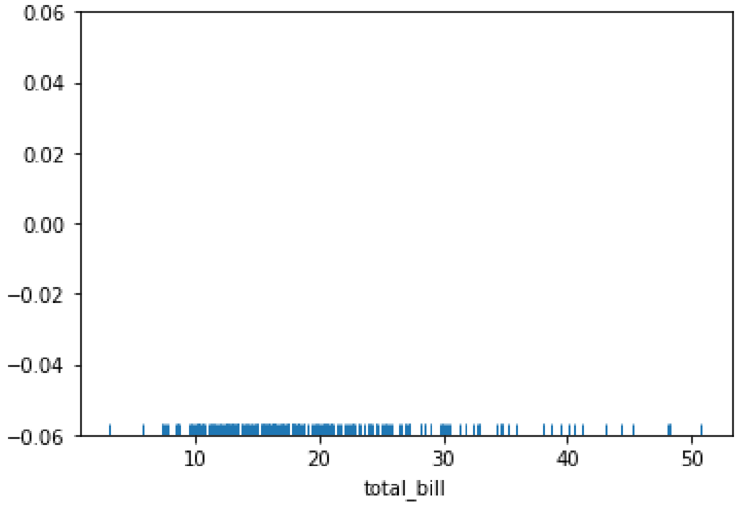
RUGPLOT

building blocks of kde plot



```
sns.rugplot(tips['total_bill'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc540df3f10>
```



CATEGORICAL PLOTS

```
import seaborn as sns
%matplotlib inline
```

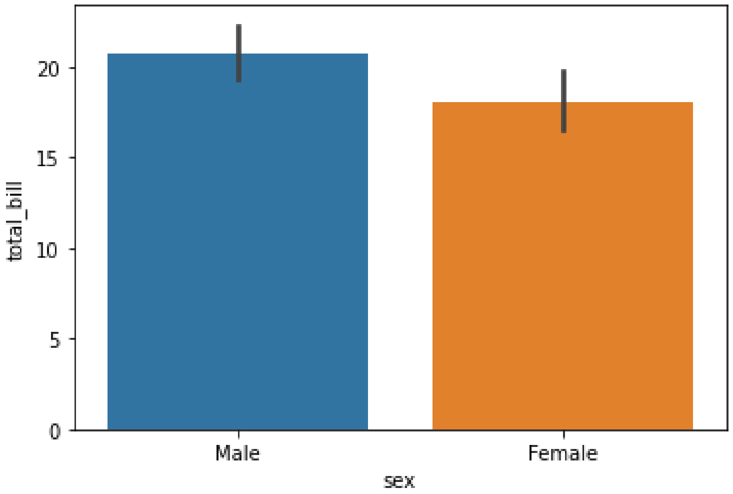
```
tips = sns.load_dataset('tips')
tips.head()
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

BARPLOT

```
sns.barplot(x='sex',y='total_bill',data=tips)
```

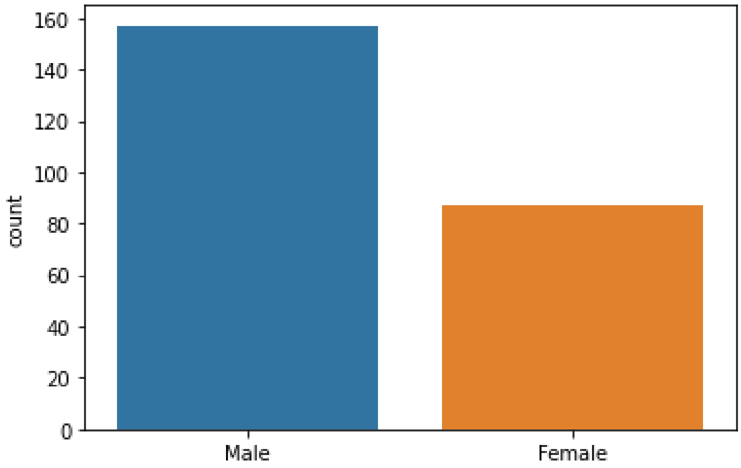
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f384bc5ff50>
```



COUNTPLOT

```
sns.countplot(x='sex',data=tips)
```

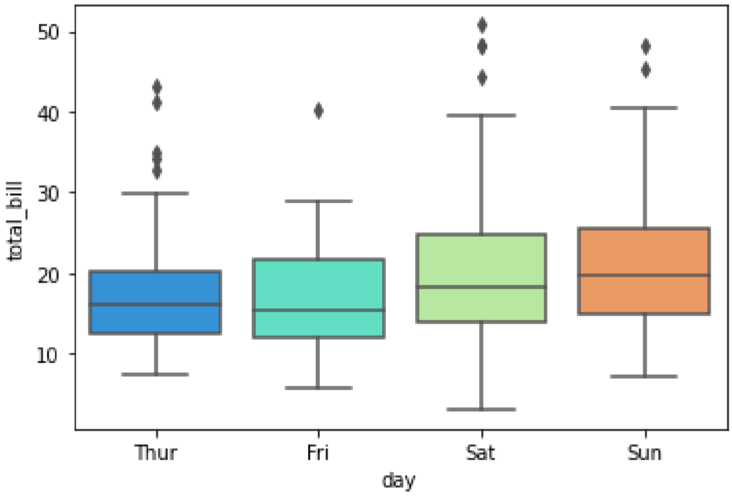
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f384b69bb50>
```



BOXPLOT: Used for comparison between variables

```
sns.boxplot(x="day", y="total_bill", data=tips,palette='rainbow')
```

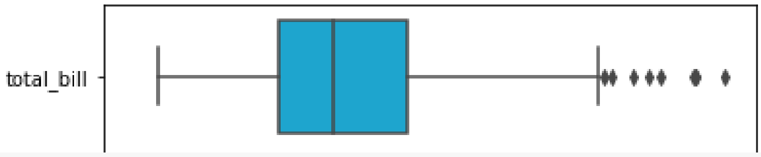
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f384bb31d90>
```



to set horizontally do

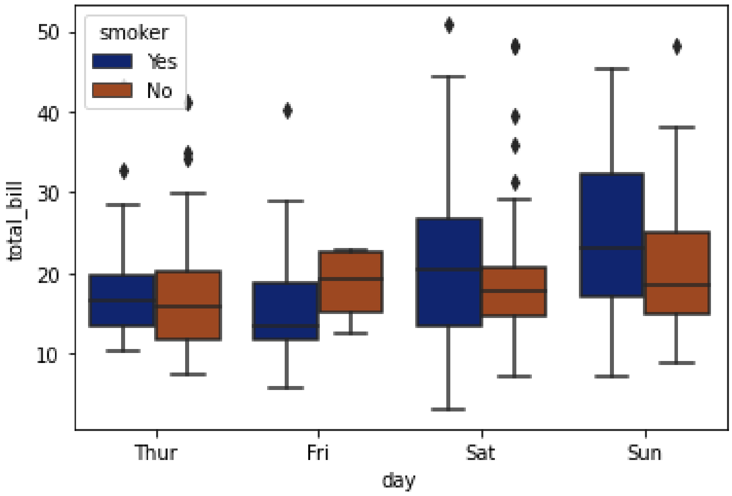
```
sns.boxplot(data=tips,palette='rainbow',orient='h')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f384b686c90>
```



```
sns.boxplot(x="day", y="total_bill", hue="smoker",data=tips, palette="dark")
```

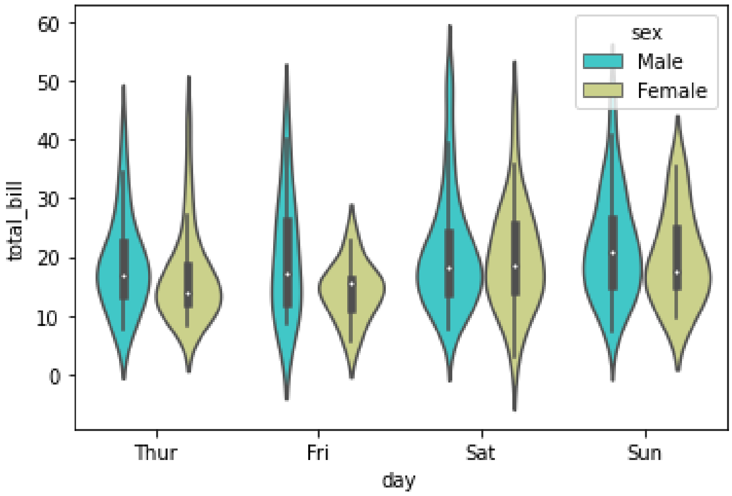
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f384b46ed10>
```



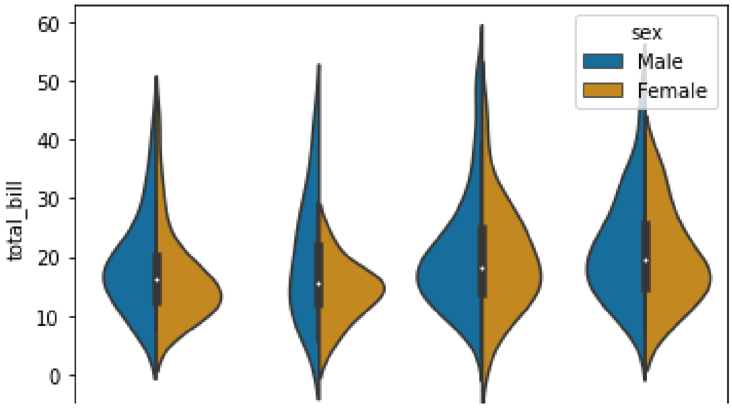
VIOLINPLOT the violin plot features a kernel density estimation of the underlying distribution.

```
sns.violinplot(x="day", y="total_bill",hue='sex', data=tips,palette='rainbow'
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3848022490>
```



```
sns.violinplot(x="day", y="total_bill", data=tips,hue='sex',split=True,palett
<matplotlib.axes._subplots.AxesSubplot at 0x7f384bc63f50>
```

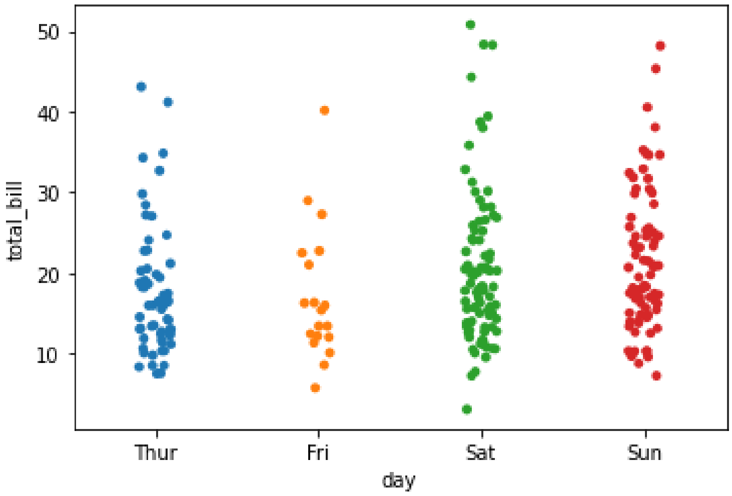


STRIPPLOT : draw a scatterplot where one variable is categorical

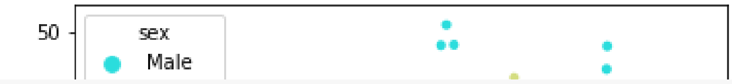
SWARMPLOT: similar to stripplot but points are adjusted(no overlap)

```
sns.stripplot(x="day", y="total_bill", data=tips)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3843521710>
```

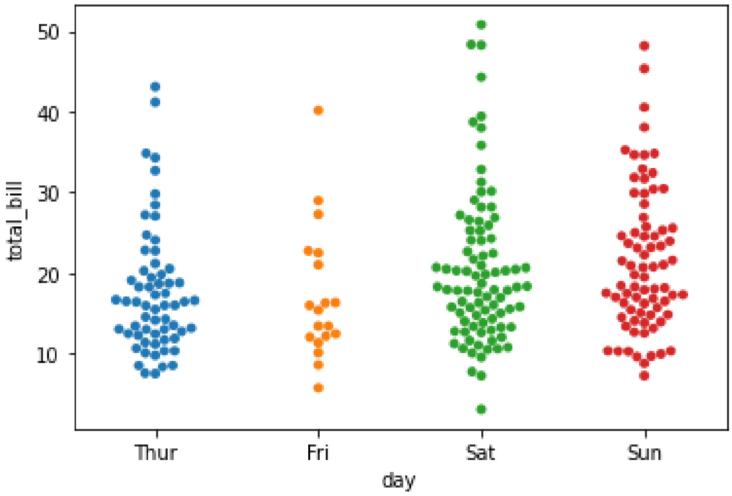


```
sns.stripplot(x="day", y="total_bill", data=tips,jitter=True,hue='sex',palett
/usr/local/lib/python3.7/dist-packages/seaborn/categorical.py:2805: UserWarning: T
  warnings.warn(msg, UserWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f38433b5410>
```



```
sns.swarmplot(x='day',y='total_bill',data=tips)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f38432f5490>
```



GRID

DATASET:iris

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
iris=sns.load_dataset('iris')
```

```
iris.head()
```

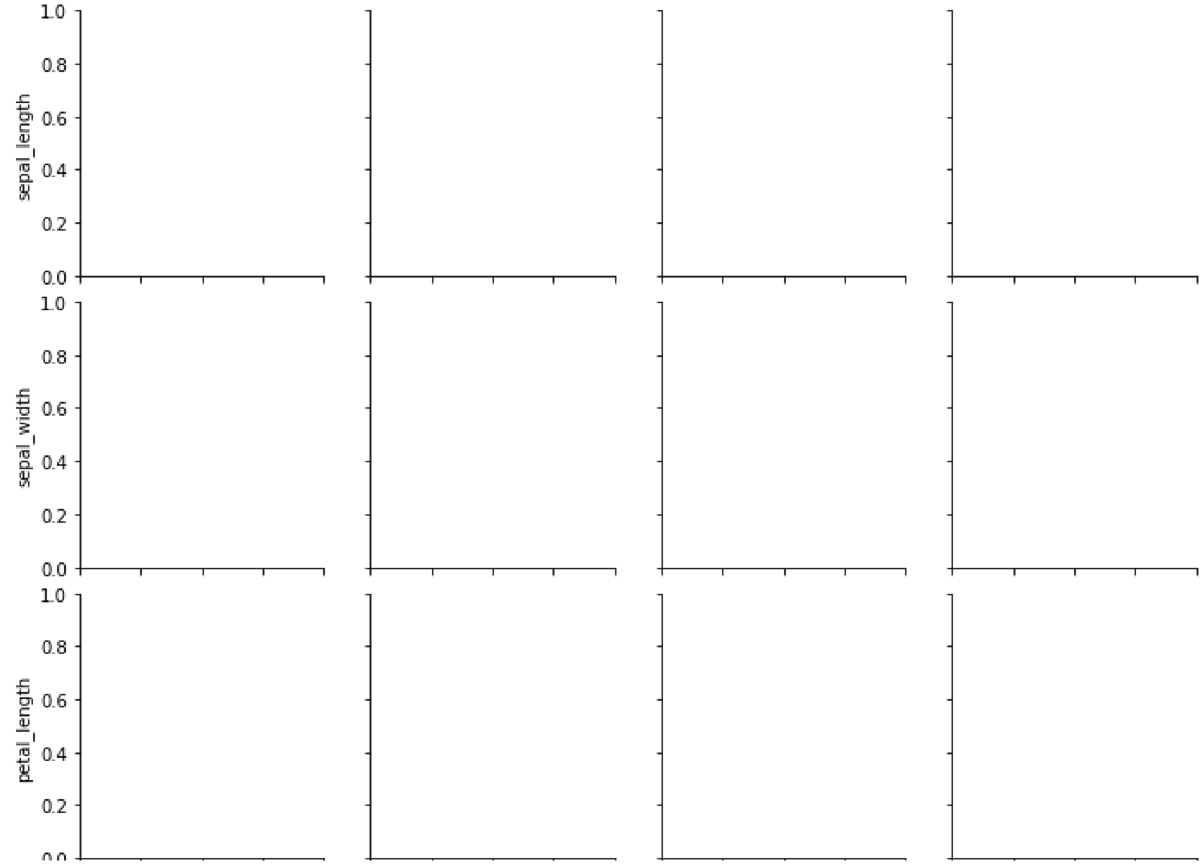
	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

pairgrid

is a subplot grid for plotting pairwise relatnship in a dataset

```
sns.PairGrid(iris)
```

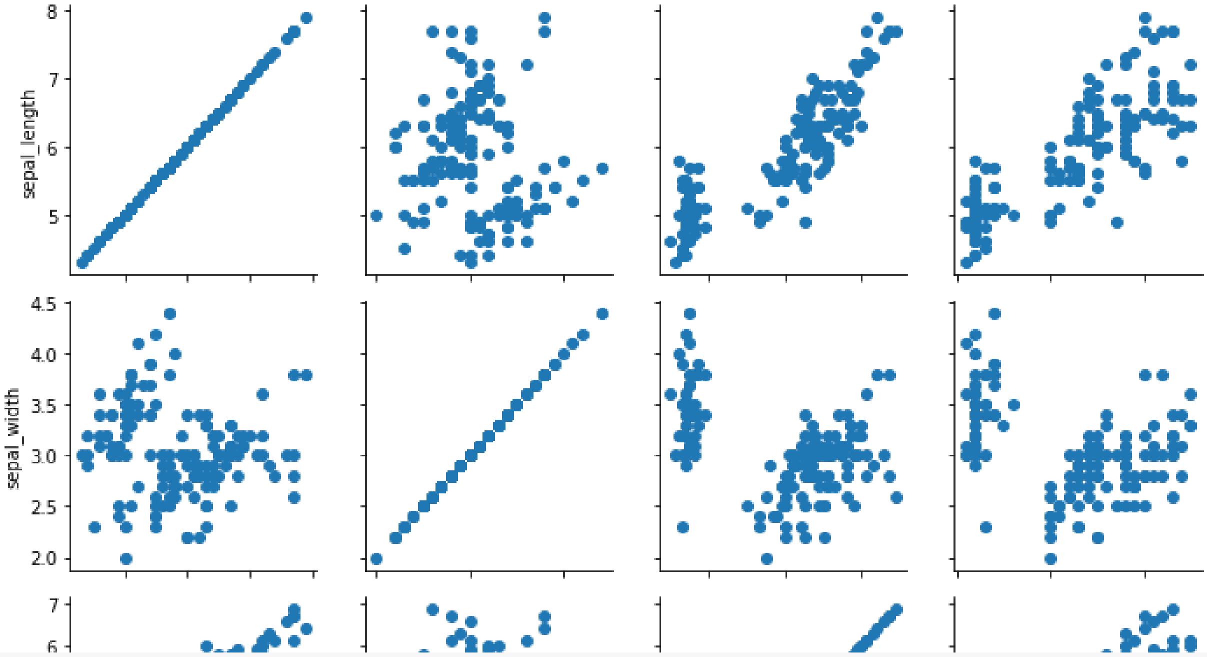
```
<seaborn.axisgrid.PairGrid at 0x7f06f6100550>
```



map to the grid

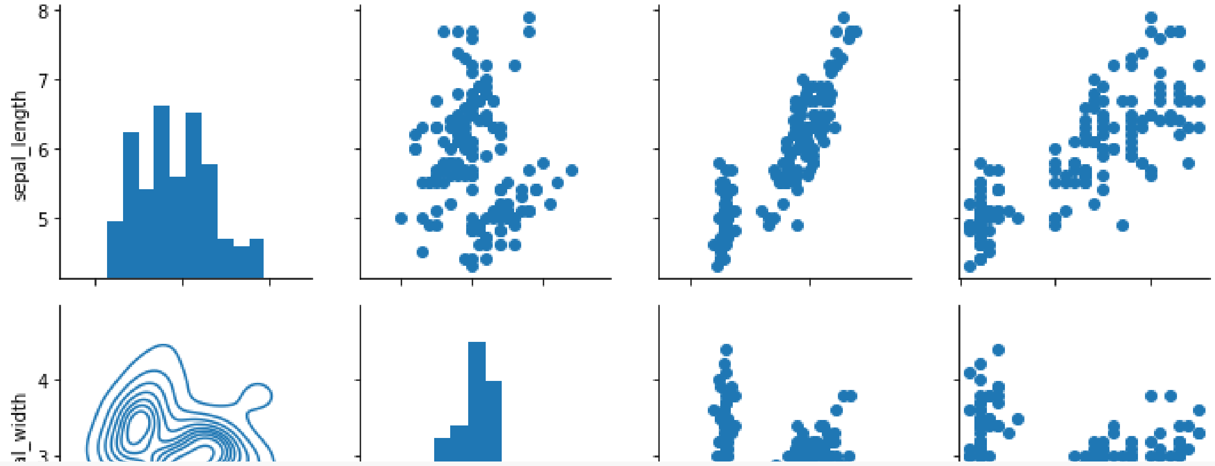
```
g= sns.PairGrid(iris)
g.map(plt.scatter)
```

```
<seaborn.axisgrid.PairGrid at 0x7f06ed3cac90>
```



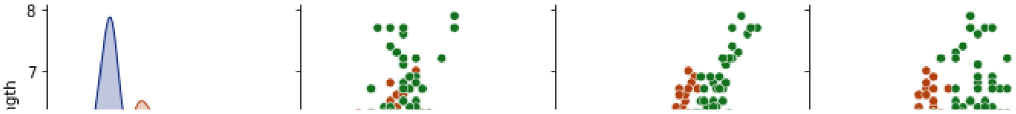
```
#map to upper,lower and diagonal
g=sns.PairGrid(iris)
g.map_diag(plt.hist)
g.map_upper(plt.scatter)
g.map_lower(sns.kdeplot)
```

```
<seaborn.axisgrid.PairGrid at 0x7f06f67a9f50>
```



```
sns.pairplot(iris,hue='species',palette='dark')
```

```
<seaborn.axisgrid.PairGrid at 0x7f06e5410cd0>
```



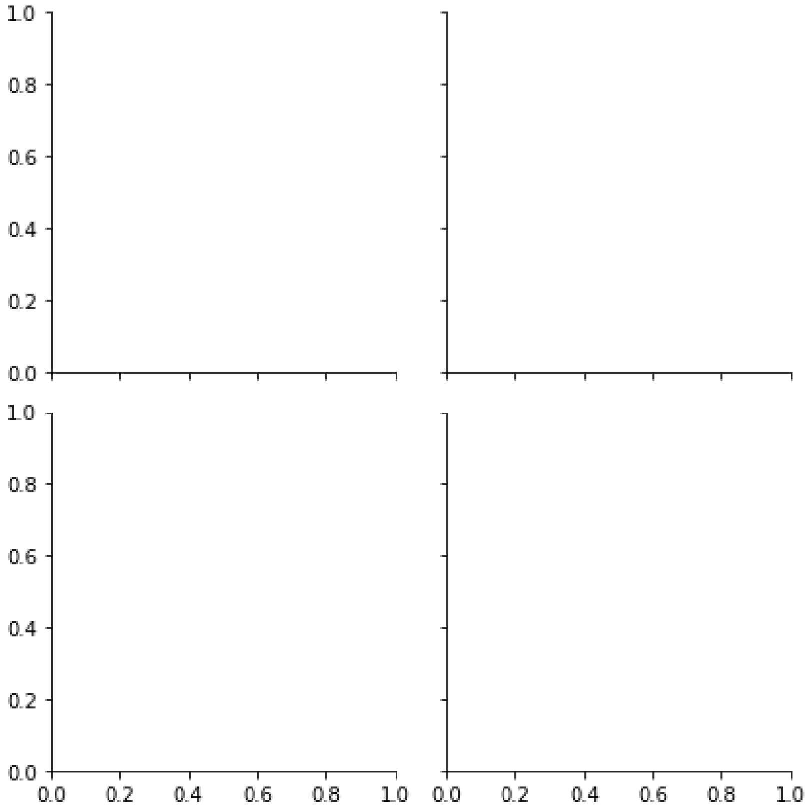
facegrid

```
tips=sns.load_dataset('tips')
```

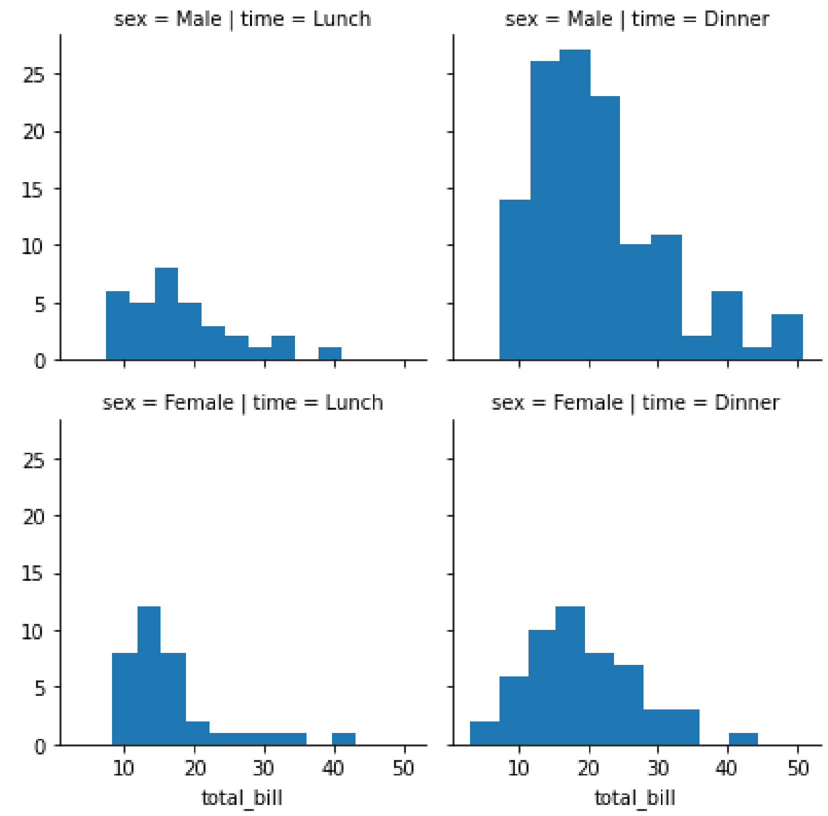
```
tips.head()
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

```
g= sns.FacetGrid(tips, col='time', row='smoker')
```



```
g= sns.FacetGrid(tips, col='time', row='sex')
g = g.map(plt.hist, 'total_bill')
```



REGRESSION PLOTS

im plot to display linear

```
sns.lmplot(x='total_bill',y='size',data=tips,hue='sex')
```

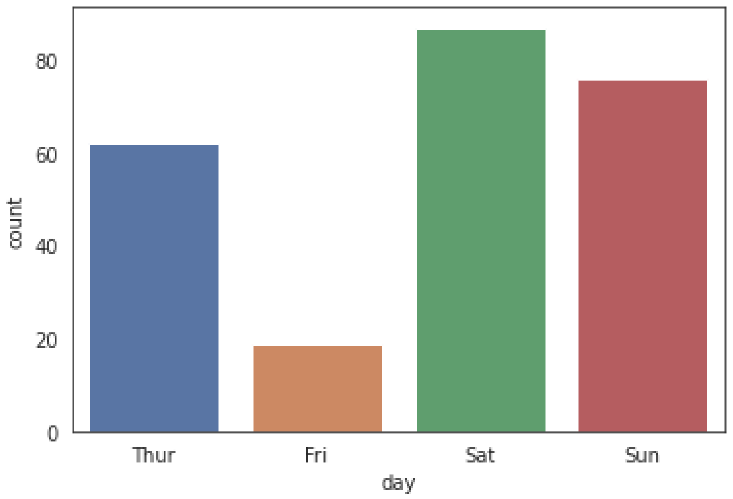
```
<seaborn.axisgrid.FacetGrid at 0x7f06e482fbd0>
```

STYLE AND COLOR

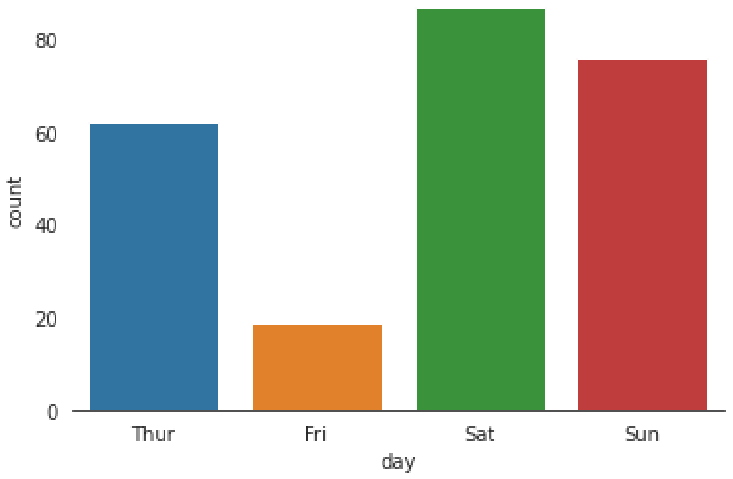
```
sns.countplot(x='sex',data=tips)
```

```
sns.set_style('white')
sns.countplot(x='day',data=tips,palette='deep')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f06e4267ad0>
```



```
sns.countplot(x='day',data=tips)
sns.despine(left='true')
```



COURSE OUTCOME 1-4

AIM:

Programs to handle data using pandas.

PROGRAM:

```
import numpy as np
import pandas as pd
```

convert a list into series

```
label=['a','b','c']
my_list=[10,20,30]
arr=np.array([10,20,30])
d={'a':10,'b':20,'c':30}
```

```
pd.Series(data=my_list)
```

```
0    10
1    20
2    30
dtype: int64
```

```
pd.Series(data=my_list,index=label)
```

```
a    10
b    20
c    30
dtype: int64
```

numpy series

```
pd.Series(arr)
```

```
0    10
1    20
2    30
dtype: int64
```

```
pd.Series(arr,label)
```

```
a    10
b    20
c    30
dtype: int64
```

dictionary

```
pd.Series(d)
```

```
a    10
b    20
c    30
dtype: int64
```

using an index

```
ser1= pd.Series([1,2,3,4],index=['usa','uk','uae','london'])
```

```
ser2= pd.Series([1,2,3,5],index=['usa','india','uae','london'])
```

```
ser1 + ser2
```

```
india     NaN
london    9.0
uae       6.0
uk        NaN
usa       2.0
dtype: float64
```

DATA FRAMES

collection of

```
import numpy as np
import pandas as pd
```

```
from numpy.random import randn
np.random.seed(101)
```

```
df= pd.DataFrame(randn(5,4),index='A B C D E'.split(),columns='W X Y Z'.split
```

```
df
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

selection and indexing

```
df['W']
```

```
A    2.706850
B    0.651118
C   -2.018168
D    0.188695
E    0.190794
Name: W, dtype: float64
```

```
df[['W','Z']]
```

	W	Z
A	2.706850	0.503826
B	0.651118	0.605965
C	-2.018168	-0.589001
D	0.188695	0.955057
E	0.190794	0.683509

```
df['new']=df['W']+df['Z']
df
```

	W	X	Y	Z	new
A	2.706850	0.628133	0.907969	0.503826	3.210676
B	0.651118	-0.319318	-0.848077	0.605965	1.257083

drop to delete col axis=0

D	0.188695	0.758872	0.933237	0.955057	1.143752

```
df.drop('new',axis=1)
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

```
df
```

	W	X	Y	Z	new
A	2.706850	0.628133	0.907969	0.503826	3.210676
B	0.651118	-0.319318	-0.848077	0.605965	1.257083
C	-2.018168	0.740122	0.528813	-0.589001	-2.607169
D	0.188695	-0.758872	-0.933237	0.955057	1.143752
E	0.190794	1.978757	2.605967	0.683509	0.874303

```
df.drop('new',axis=1,inplace=True)
```

```
df
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

```
df['A']
```

display row

```
df.loc['A']
```

```
W    2.706850
X    0.628133
Y    0.907969
Z    0.503826
Name: A, dtype: float64
```

```
df.loc[1]
```

```
df.iloc[1]
```

```
W    0.651118
X   -0.319318
Y   -0.848077
Z    0.605965
Name: B, dtype: float64
```

```
df.loc['A','X']
```

```
0.6281327087844596
```

```
df.loc[['A','B'],['X','Y']]
```

	X	Y
A	0.628133	0.907969
B	-0.319318	-0.848077

conditional selection

```
df
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965

```
df[df>0]
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	NaN	NaN	0.605965
C	NaN	0.740122	0.528813	NaN
D	0.188695	NaN	NaN	0.955057
E	0.190794	1.978757	2.605967	0.683509

```
df[df['W']>0]
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

```
df[df['W']>0]['Y']
```

```
A    0.907969
B   -0.848077
D   -0.933237
E    2.605967
Name: Y, dtype: float64
```

```
df[(df['W']>0)  & (df['Y']>0)]
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
E	0.190794	1.978757	2.605967	0.683509

```
df[(df['W']>0) | (df['Y']>0)]
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

OPERATIONS

```
import numpy as np
import pandas as pd
```

```
df = pd.DataFrame({'col1':[1,2,3,4],'col2':[444,555,666,444],'col3':['abc','c
df.head()
```

	col1	col2	col3
0	1	444	abc
1	2	555	def
2	3	666	ghi
3	4	444	xyz

info on unique values

```
df['col2'].unique()
```

```
array([444, 555, 666])
```

```
df['col2'].nunique()
```

```
3
```

```
df['col2'].value_counts()
```

```
444    2
555    1
666    1
Name: col2, dtype: int64
```

selecting data

```
newdf = df[(df['col1']>2) & (df['col2']==444)]
```

```
newdf
```

	col1	col2	col3
3	4	444	xyz

```
df['col3'].apply(len)
```

```
0    3
1    3
2    3
3    3
Name: col3, dtype: int64
```

```
df['col1'].sum()
```

```
10
```

permanantly deleting a column

```
del df['col1']
```

```
df
```

	col2	col3
0	444	abc
1	555	def
2	666	ghi
3	444	xyz

```
df.columns
```

```
Index(['col2', 'col3'], dtype='object')
```

```
df.index
```

```
RangeIndex(start=0, stop=4, step=1)
```

sorting and ordering dataframe

```
df.sort_values(by='col3')
```

	col2	col3
0	444	abc
1	555	def
2	666	ghi
3	444	xyz

```
df.sort_values(by='col2',inplace=True)
```

```
df
```

	col2	col3
0	444	abc
3	444	xyz
1	555	def
2	666	ghi

find null values

```
df.isnull()
```

	col2	col3
0	False	False
3	False	False
1	False	False
2	False	False

EXCERCISE

DATASET : Salaries.csv

```
import pandas as pd
```

```
df = pd.read_csv('/content/sample_data/Salaries (2).csv')
```

```
df
```

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefi
0	1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	167411.18	0.00	400184.25	N
1	2	GARY JIMENEZ	CAPTAIN III (POLICE DEPARTMENT)	155966.02	245131.88	137811.38	N
2	3	ALBERT PARDINI	CAPTAIN III (POLICE DEPARTMENT)	212739.13	106088.18	16452.60	N
3	4	CHRISTOPHER CHONG	WIRE ROPE CABLE MAINTENANCE MECHANIC	77916.00	56120.71	198306.90	N
4	5	PATRICK GARDNER	DEPUTY CHIEF OF DEPARTMENT,(FIRE DEPARTMENT)	134401.60	9737.00	182234.59	N
...	...	...	...	...	...	...	
148649	148650	Roy I Tillery	Custodian	0.00	0.00	0.00	
148650	148651	Not provided	Not provided	NaN	NaN	NaN	N
148651	148652	Not provided	Not provided	NaN	NaN	NaN	N

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 148654 entries, 0 to 148653
Data columns (total 13 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Id                148654 non-null  int64  
 1   EmployeeName      148654 non-null  object 
 2   JobTitle          148654 non-null  object 
 3   BasePay           148045 non-null  float64
 4   OvertimePay       148650 non-null  float64
 5   OtherPay          148650 non-null  float64
 6   Benefits          112491 non-null  float64
 7   TotalPay          148654 non-null  float64
 8   TotalPayBenefits  148654 non-null  float64
 9   Year              148654 non-null  int64  
 10  Notes             0 non-null       float64
 11  Agency            148654 non-null  object 
 12  Status            0 non-null       float64
dtypes: float64(8), int64(2), object(3)
memory usage: 14.7+ MB
```

What is the average BasePay ?

```
df['BasePay'].mean()
```

```
66325.44884050643
```

what is the highest amount of overtimepay in the dataset

```
df['OvertimePay'].max()
```

```
245131.88
```

what is the job title of joseph criscoll?

```
df[df['EmployeeName']=='JOSEPH DRISCOLL']['JobTitle']
```

```
24    CAPTAIN, FIRE SUPPRESSION
Name: JobTitle, dtype: object
```

how much does JOSEPH CRISCOLL make including benefits

```
df[df['EmployeeName']=='JOSEPH DRISCOLL']['TotalPayBenefits']
```

```
24    270324.91
Name: TotalPayBenefits, dtype: float64
```

what is the name of highest paid person (including benefits)

```
df[df['TotalPayBenefits'] == df['TotalPayBenefits'].max()] ['EmployeeName']
```

```
0    NATHANIEL FORD
Name: EmployeeName, dtype: object
```

what is the name of lowest paid person

```
df[df['TotalPayBenefits'] == df['TotalPayBenefits'].min()] ['EmployeeName']
```

```
148653    Joe Lopez
Name: EmployeeName, dtype: object
```

what is the average(mean) basepay of all employees per year?

```
df.groupby('Year').mean()['BasePay']
```

```
Year
2011    63595.956517
2012    65436.406857
2013    69630.030216
2014    66564.421924
Name: BasePay, dtype: float64
```

How many unique job titles are there?

```
df['JobTitle'].nunique()
```

```
2159
```

what are the five most common jobs?

```
df['JobTitle'].value_counts()[:5]
```

```
Transit Operator                7036
Special Nurse                   4389
Registered Nurse                3736
Public Svc Aide-Public Works    2518
Police Officer 3                2421
Name: JobTitle, dtype: int64
```

```
df['JobTitle'].head(5)  #topmost
```

```
0    GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY
1                     CAPTAIN III (POLICE DEPARTMENT)
2                     CAPTAIN III (POLICE DEPARTMENT)
3               WIRE ROPE CABLE MAINTENANCE MECHANIC
4       DEPUTY CHIEF OF DEPARTMENT,(FIRE DEPARTMENT)
Name: JobTitle, dtype: object
```

how many job titles were represented by only one person in 2013

```
sum(df[df['Year'] == 2013]['JobTitle'].value_counts()==1)
```

```
202
```

how many people have the word chief in their job title?

```
sum(df['JobTitle'].apply(lambda x:'chief' in x.lower().split()))
```