Matplotlib By Virat Tiwari

October 18, 2023

1 Why we need Data Visualization - When somebody gave us million or trillions of data for finding the sentiments or may be the relationship between the entire data so just by looking into that data it is not possible . In simple terms , it was almost impossible to find the Patterns from millions record of data by just looking into that numbers. We can not ANALYSIS the TREND , So Here we use GRAPGH OR VISUALIZATION CONCEPT for understanding the SUMMARY of Data or getting the TREND of millions Records of Data Easily.DATA VISULAIZATION help us to understand the huge amount of data and we can easily conclude the result or getting the INSIGHTS with the help of GRAPGH that we made in DATA VISUALIZATION .

[1]:
```python
# This is how we import " matplotlib " with " pylot "

import matplotlib.pyplot as plt
```

[2]:
```python
import numpy as np
```

Note - Grapgh is nothing but a representation of data on x-axis , y-axis or z-axis

Note - We can Plot or Visualize the data with the help of Matplotlib

CASE - 1

[3]:
```python
# linspace ( ) function is used foe generating the data

# In linspace ( ) function , we pass the range like 0 to 10 and nuber of data like 200 hundred data

x=np.linspace(0,10,200)
```

[4]:
```python
x
```

```
[4]: array([ 0.        ,  0.05025126,  0.10050251,  0.15075377,  0.20100503,
             0.25125628,  0.30150754,  0.35175879,  0.40201005,  0.45226131,
             0.50251256,  0.55276382,  0.60301508,  0.65326633,  0.70351759,
             0.75376884,  0.8040201 ,  0.85427136,  0.90452261,  0.95477387,
             1.00502513,  1.05527638,  1.10552764,  1.15577889,  1.20603015,
             1.25628141,  1.30653266,  1.35678392,  1.40703518,  1.45728643,
             1.50753769,  1.55778894,  1.6080402 ,  1.65829146,  1.70854271,
             1.75879397,  1.80904523,  1.85929648,  1.90954774,  1.95979899,
             2.01005025,  2.06030151,  2.11055276,  2.16080402,  2.21105528,
             2.26130653,  2.31155779,  2.36180905,  2.4120603 ,  2.46231156,
             2.51256281,  2.56281407,  2.61306533,  2.66331658,  2.71356784,
             2.7638191 ,  2.81407035,  2.86432161,  2.91457286,  2.96482412,
             3.01507538,  3.06532663,  3.11557789,  3.16582915,  3.2160804 ,
             3.26633166,  3.31658291,  3.36683417,  3.41708543,  3.46733668,
             3.51758794,  3.5678392 ,  3.61809045,  3.66834171,  3.71859296,
             3.76884422,  3.81909548,  3.86934673,  3.91959799,  3.96984925,
             4.0201005 ,  4.07035176,  4.12060302,  4.17085427,  4.22110553,
             4.27135678,  4.32160804,  4.3718593 ,  4.42211055,  4.47236181,
             4.52261307,  4.57286432,  4.62311558,  4.67336683,  4.72361809,
             4.77386935,  4.8241206 ,  4.87437186,  4.92462312,  4.97487437,
             5.02512563,  5.07537688,  5.12562814,  5.1758794 ,  5.22613065,
             5.27638191,  5.32663317,  5.37688442,  5.42713568,  5.47738693,
             5.52763819,  5.57788945,  5.6281407 ,  5.67839196,  5.72864322,
             5.77889447,  5.82914573,  5.87939698,  5.92964824,  5.9798995 ,
             6.03015075,  6.08040201,  6.13065327,  6.18090452,  6.23115578,
             6.28140704,  6.33165829,  6.38190955,  6.4321608 ,  6.48241206,
             6.53266332,  6.58291457,  6.63316583,  6.68341709,  6.73366834,
             6.7839196 ,  6.83417085,  6.88442211,  6.93467337,  6.98492462,
             7.03517588,  7.08542714,  7.13567839,  7.18592965,  7.2361809 ,
             7.28643216,  7.33668342,  7.38693467,  7.43718593,  7.48743719,
             7.53768844,  7.5879397 ,  7.63819095,  7.68844221,  7.73869347,
             7.78894472,  7.83919598,  7.88944724,  7.93969849,  7.98994975,
             8.04020101,  8.09045226,  8.14070352,  8.19095477,  8.24120603,
             8.29145729,  8.34170854,  8.3919598 ,  8.44221106,  8.49246231,
             8.54271357,  8.59296482,  8.64321608,  8.69346734,  8.74371859,
             8.79396985,  8.84422111,  8.89447236,  8.94472362,  8.99497487,
             9.04522613,  9.09547739,  9.14572864,  9.1959799 ,  9.24623116,
             9.29648241,  9.34673367,  9.39698492,  9.44723618,  9.49748744,
             9.54773869,  9.59798995,  9.64824121,  9.69849246,  9.74874372,
             9.79899497,  9.84924623,  9.89949749,  9.94974874, 10.        ])
```

```
[5]: # Here we also generate data bu using " np.sin ( ) function " , in which we␣
      ↪pass " x " that generate 200 hunded data from the range 1 to 10 same as␣
      ↪previous x but in diffrent numbers
     y=np.sin(x)
```

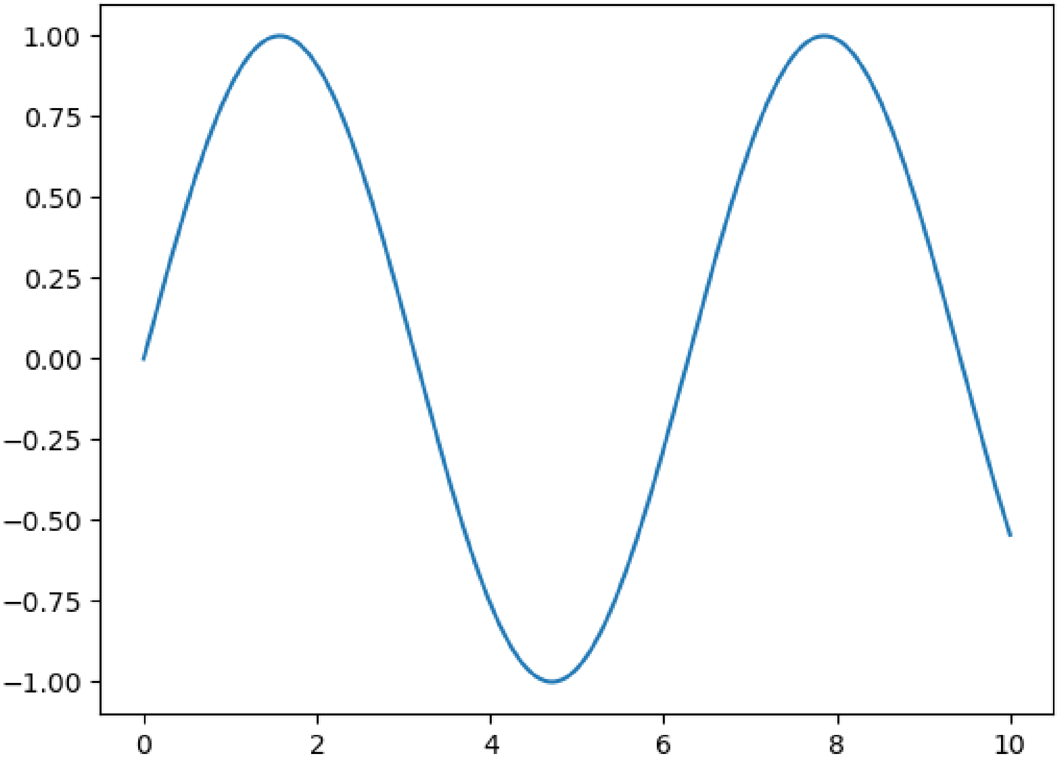
```
[6]: y
```

```
[6]: array([ 0.        ,  0.05023011,  0.10033341,  0.15018339,  0.19965422,
             0.24862099,  0.29696008,  0.34454944,  0.39126893,  0.43700061,
             0.481629  ,  0.52504145,  0.56712835,  0.60778345,  0.6469041 ,
             0.68439153,  0.72015112,  0.75409257,  0.78613019,  0.8161831 ,
             0.84417544,  0.87003651,  0.89370105,  0.91510929,  0.9342072 ,
             0.95094655,  0.96528509,  0.97718662,  0.98662108,  0.99356467,
             0.99799984,  0.99991541,  0.99930653,  0.99617474,  0.99052796,
             0.98238043,  0.97175273,  0.95867168,  0.94317032,  0.92528777,
             0.90506919,  0.88256563,  0.85783388,  0.8309364 ,  0.80194109,
             0.77092115,  0.7379549 ,  0.70312557,  0.66652108,  0.62823386,
             0.58836056,  0.54700186,  0.50426216,  0.46024937,  0.41507461,
             0.36885193,  0.32169803,  0.27373195,  0.22507478,  0.17584939,
             0.12618003,  0.07619211,  0.02601183, -0.02423412, -0.07441889,
            -0.12441577, -0.17409855, -0.22334179, -0.27202116, -0.32001378,
            -0.36719847, -0.41345611, -0.45866992, -0.50272574, -0.54551235,
            -0.58692173, -0.62684933, -0.66519435, -0.70185999, -0.73675367,
            -0.7697873 , -0.80087747, -0.82994571, -0.85691862, -0.88172811,
            -0.90431153, -0.92461187, -0.94257789, -0.95816422, -0.97133152,
            -0.98204653, -0.99028221, -0.99601778, -0.99923873, -0.99993695,
            -0.99811068, -0.99376451, -0.98690943, -0.97756275, -0.96574805,
            -0.95149517, -0.93484009, -0.91582485, -0.89449748, -0.8709118 ,
            -0.84512737, -0.81720929, -0.78722803, -0.75525929, -0.72138377,
            -0.68568702, -0.64825913, -0.60919462, -0.56859209, -0.52655407,
            -0.48318668, -0.4385994 , -0.39290482, -0.34621828, -0.29865766,
            -0.25034303, -0.20139637, -0.15194126, -0.10210255, -0.05200606,
            -0.00177827,  0.048454  ,  0.09856395,  0.14842506,  0.19791144,
             0.24689816,  0.29526155,  0.34287951,  0.38963181,  0.43540043,
             0.48006981,  0.52352718,  0.56566282,  0.60637036,  0.64554701,
             0.68309389,  0.71891618,  0.75292346,  0.78502987,  0.81515434,
             0.84322083,  0.86915847,  0.89290179,  0.91439084,  0.93357136,
             0.95039493,  0.96481908,  0.9768074 ,  0.98632961,  0.99336168,
             0.99788585,  0.99989069,  0.99937116,  0.99632856,  0.99077057,
             0.98271122,  0.97217086,  0.95917611,  0.94375976,  0.92596075,
             0.905824  ,  0.88340035,  0.85874643,  0.83192446,  0.80300216,
             0.77205257,  0.7391538 ,  0.70438892,  0.66784571,  0.62961641,
             0.58979754,  0.54848964,  0.50579699,  0.46182738,  0.41669181,
             0.37050423,  0.32338126,  0.27544187,  0.22680707,  0.17759967,
             0.12794389,  0.07796509,  0.02778946, -0.02245633, -0.07264543,
            -0.12265112, -0.17234716, -0.22160808, -0.27030952, -0.31832851,
            -0.36554384, -0.4118363 , -0.45708901, -0.50118772, -0.54402111])
```

Important Note - We successfully generate x-axis and y-axis data but we can not analysis the movement of x-cordinate and y-cordinate of data by just looking at the data numbers so here we use or draw a grapgh which help us to visualize the data and we simply understand the movement of x and y co-ordination of axis by just looking at the grapgh

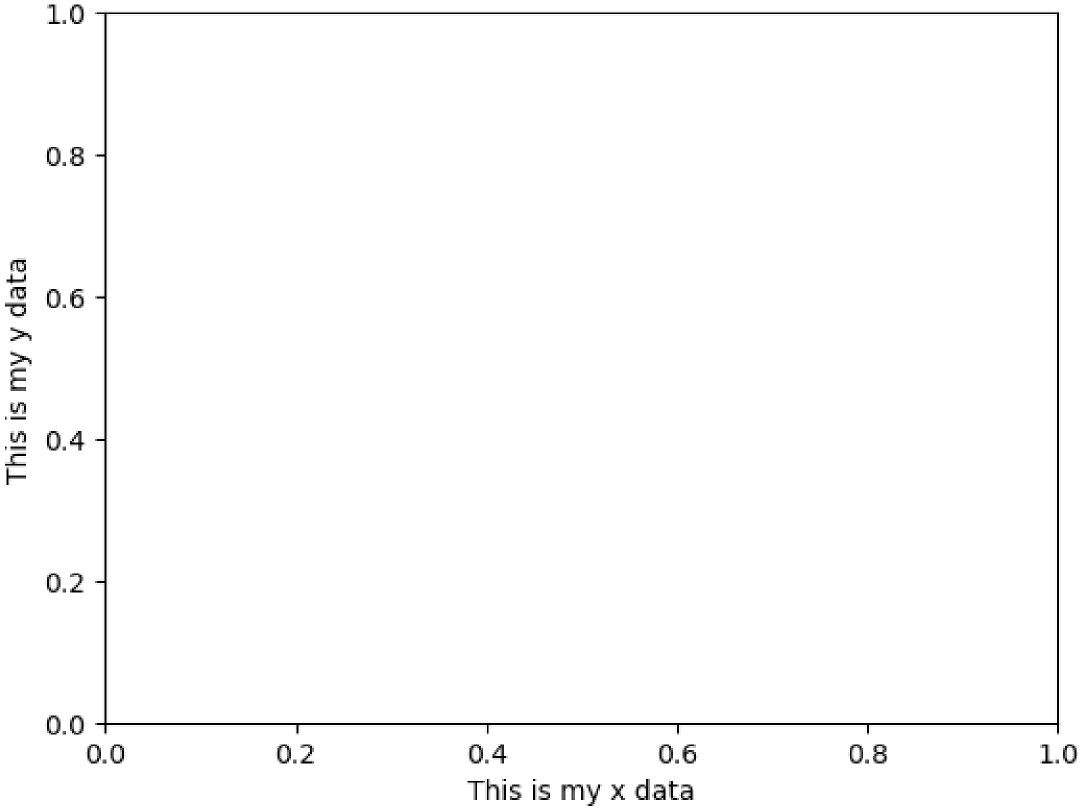
[7]:
```
# plt is our alias or matplotlib like we denote matplotlib as plt so whenevr we␣
 ↪use matplotlib we call plt

# plt.plot ( ) function is used for plot the data into the grapgh and we pass␣
 ↪the axis or variables like x and y inside the plot

plt.plot(x,y)
```

[7]:
```
[<matplotlib.lines.Line2D at 0x7f4bd4c3af50>]
```



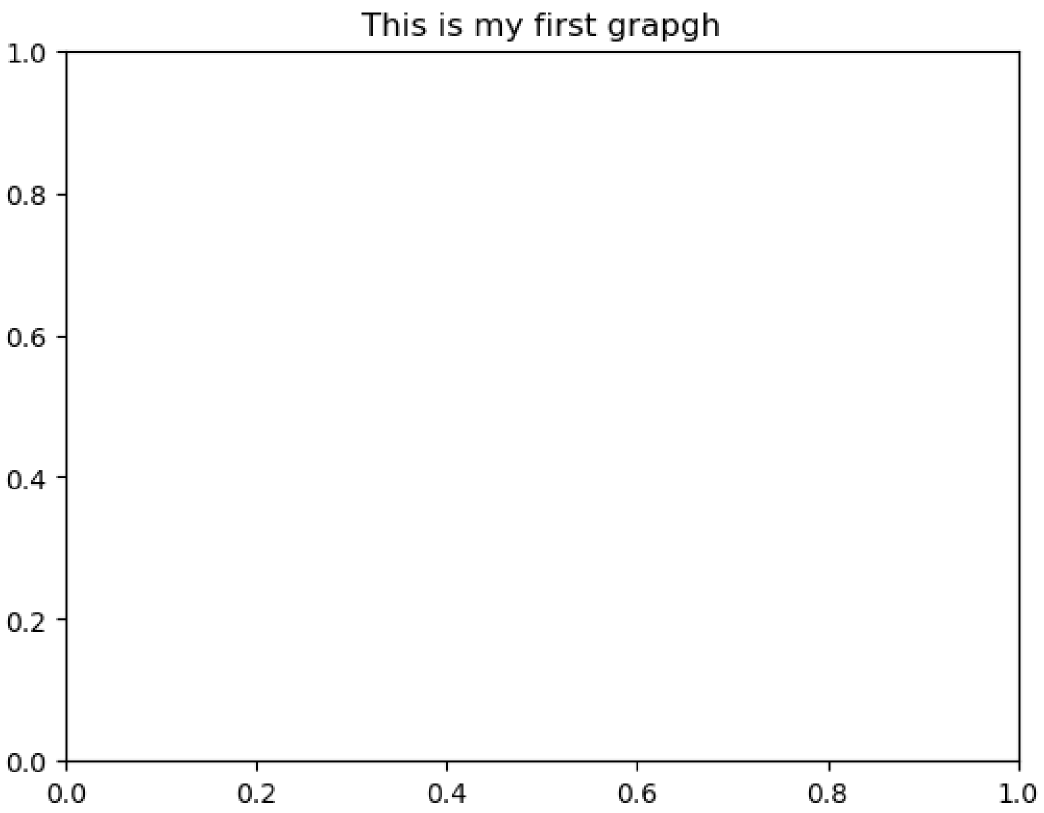
[8]:
```
# plt.xlabel or plt.ylabel ( ) function is used for labelling the data data .␣
 ↪We pass the label or any text that present the x or y axis

plt.xlabel("This is my x data")
plt.ylabel("This is my y data")
```

[8]:
```
Text(0, 0.5, 'This is my y data')
```



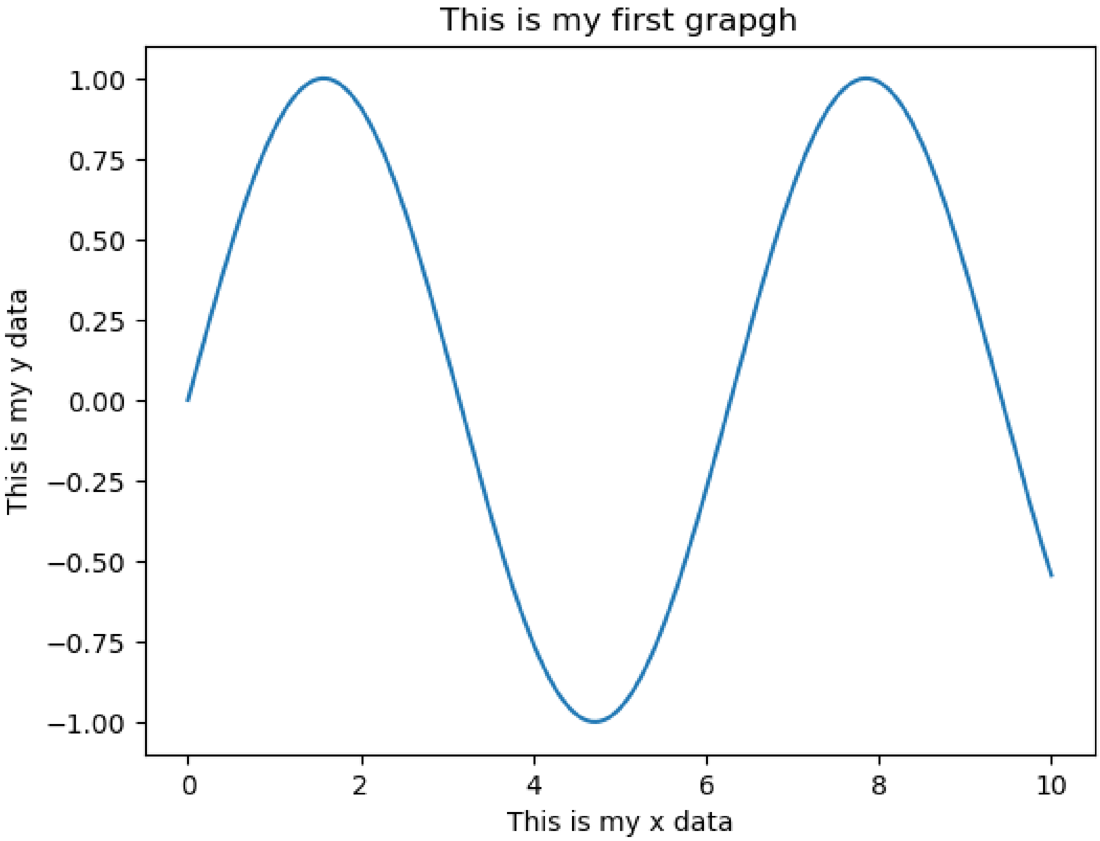
```
[9]: # plt.title ( ) function is used for given the name of grapgh by passing the␣
      ↪text inside the title ( ) function

     plt.title("This is my first grapgh")
```

```
[9]: Text(0.5, 1.0, 'This is my first grapgh')
```



```
[10]: # This is final presentation of grapgh with all naming text  that justify the␣
      ↪entire grapgh like title , name of x and y axis etc

      plt.plot(x,y)
      plt.title("This is my first grapgh")
      plt.xlabel("This is my x data")
      plt.ylabel("This is my y data")
```

```
[10]: Text(0, 0.5, 'This is my y data')
```



CASE - 2

[11]:

```
# np.random.rand ( ) function is used for genrating the random data by passing the value inside the function like here we pass 50 data

# x is a variable that store the random 50 data

x=np.random.rand(50)
```

[12]:

```
# We call x to show the data that we generate

x
```

[12]:

```
array([0.3264215 , 0.72191338, 0.7333547 , 0.81296528, 0.4911694 ,
       0.03063241, 0.42982928, 0.00637064, 0.68367324, 0.93270052,
       0.26664415, 0.06082933, 0.12513395, 0.41651939, 0.74616772,
       0.9637124 , 0.89000411, 0.83290546, 0.20541492, 0.24282567,
       0.02972526, 0.98269047, 0.94256722, 0.68632248, 0.19913464,
       0.70416475, 0.49065208, 0.12038608, 0.37673275, 0.4517113 ,
       0.24964646, 0.319042  , 0.30186613, 0.83047236, 0.01001234,
       0.92400325, 0.13612251, 0.23418177, 0.36338538, 0.30657197,
       0.3052248 , 0.56588282, 0.64823433, 0.58840345, 0.40956623,
       0.91555259, 0.20541003, 0.92396157, 0.57789832, 0.08059255])
```

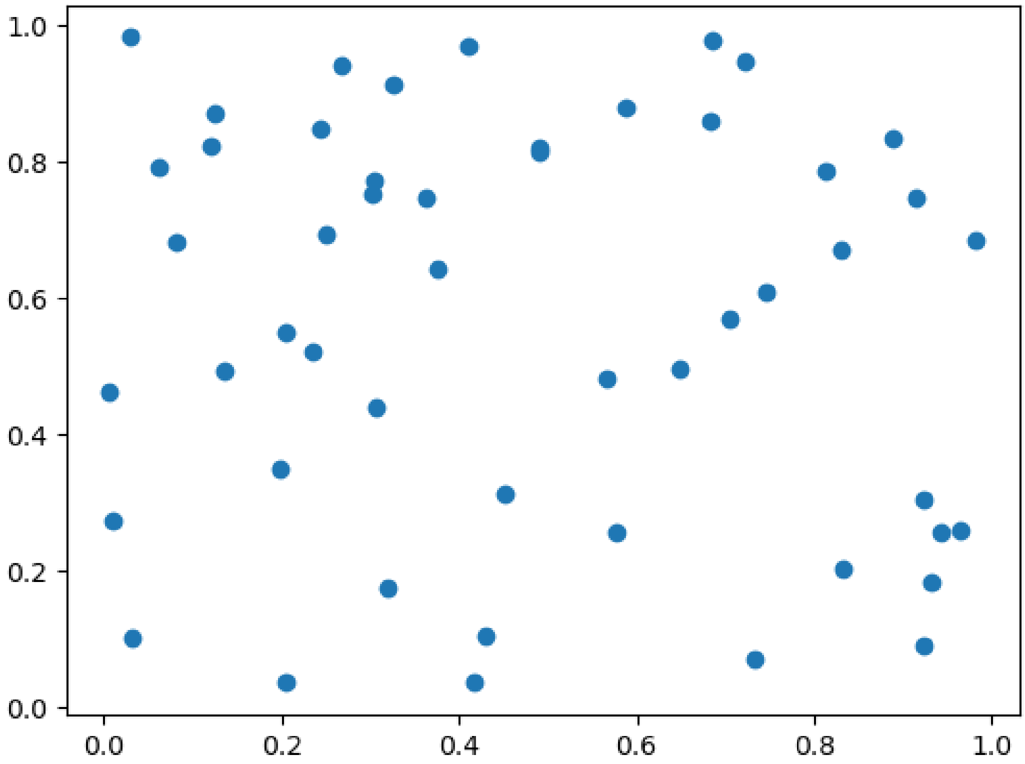
```
[13]: # np.random.rand ( ) function is used for genrating the random data by passing␣
       ↪the value inside the function like here we pass 50 data

      # y is a variable that store the random 50 data



      y=np.random.rand(50)
```

```
[14]: # Now we call y to show the generated data

      y
```

```
[14]: array([0.91341842, 0.94708124, 0.06875672, 0.78643276, 0.81436647,
             0.10016853, 0.10482149, 0.46316924, 0.8606652 , 0.18358126,
             0.94270757, 0.79259743, 0.86975442, 0.03467551, 0.6095806 ,
             0.25922808, 0.83297392, 0.20273026, 0.54995152, 0.84956303,
             0.9823761 , 0.68375716, 0.25733   , 0.97788305, 0.34847496,
             0.56770074, 0.82106037, 0.82286519, 0.64152776, 0.31240952,
             0.69248289, 0.17329901, 0.75242493, 0.67140246, 0.27214963,
             0.30399486, 0.4931393 , 0.52092903, 0.74746735, 0.44004225,
             0.77351238, 0.48088991, 0.49597598, 0.88060822, 0.97056656,
             0.7467046 , 0.03526698, 0.08932695, 0.25692132, 0.68314107])
```

```
[15]: # plt.scatter ( ) function is used for present the data in scatter form

      plt.scatter(x,y)
```

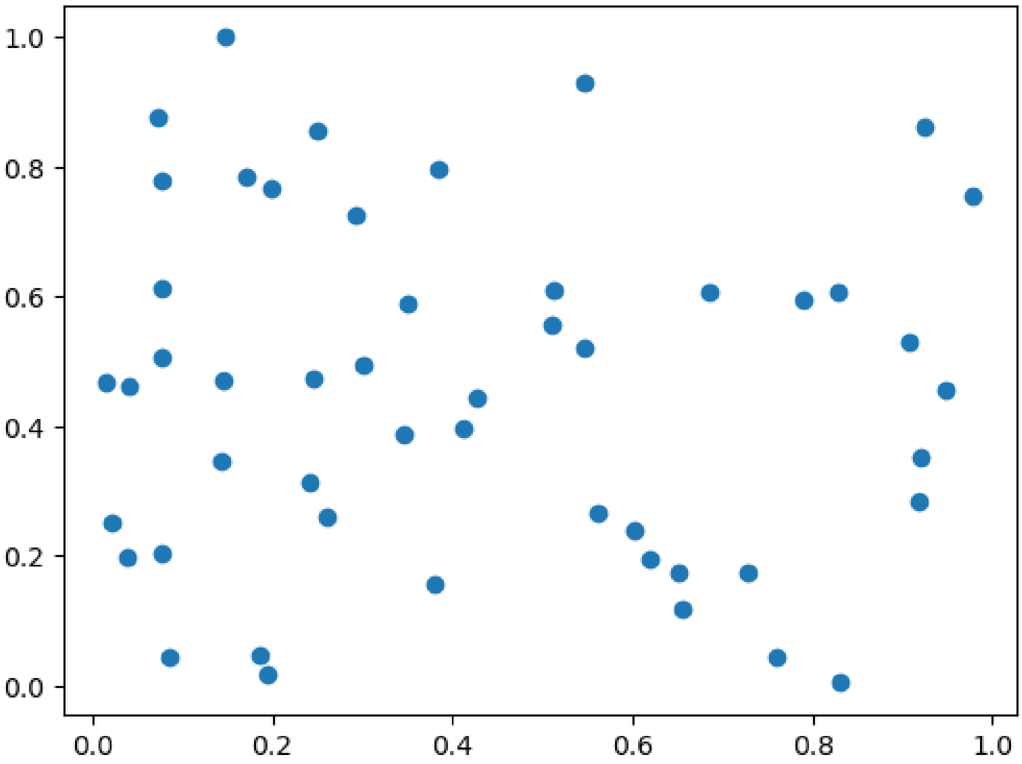
```
[15]: <matplotlib.collections.PathCollection at 0x7f4bcca7eb00>
```



OR

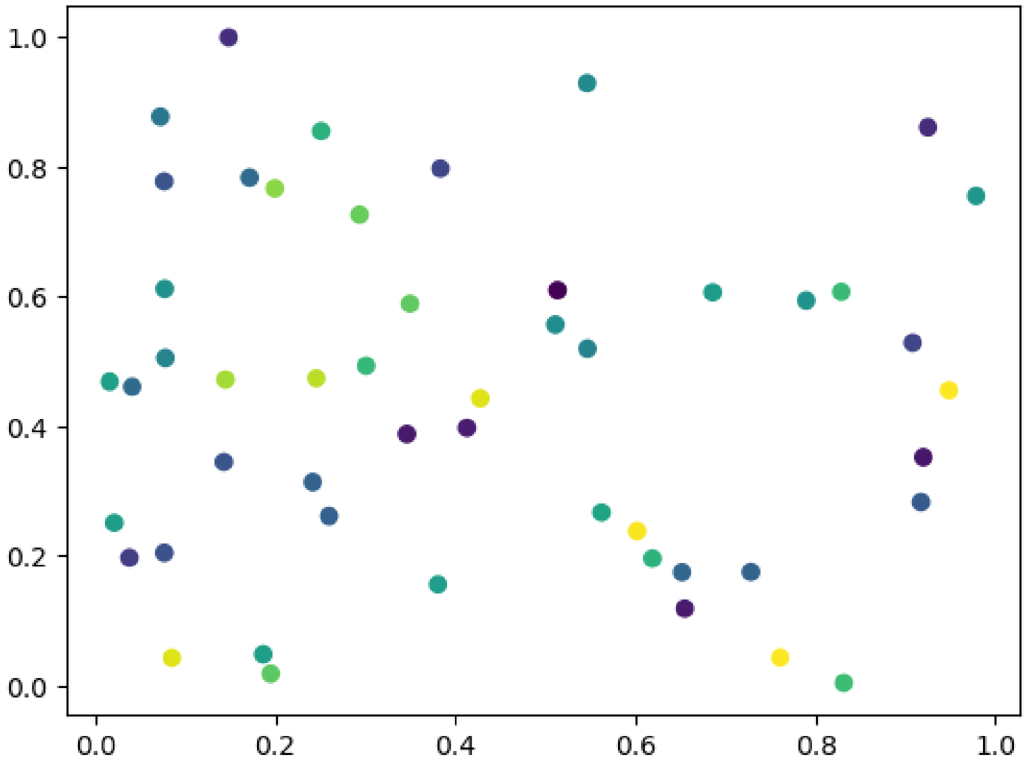
```
[16]: # By compiling all three functions we direct call the grapgh without executing␣
        ↪the function seperately

      x=np.random.rand(50)
      y=np.random.rand(50)
      plt.scatter(x,y)
```

```
[16]: <matplotlib.collections.PathCollection at 0x7f4bcc8ffa30>
```



[17]:
```
# This is we colouring the data

colours=np.random.rand(50)
```

[18]:
```
# This is we colouring the data

plt.scatter(x,y,c=colours)
colours=np.random.rand(50)
```



CASE - 3

We present Categorical vs Numerical Data so we take 5 categorical data and 5 numerical data

[19]:

```
# We take varaibles like a , b , c , d etc in x

# Here we generate the 5 variables data

x=["a","b","c","d","e"]
```

[20]:

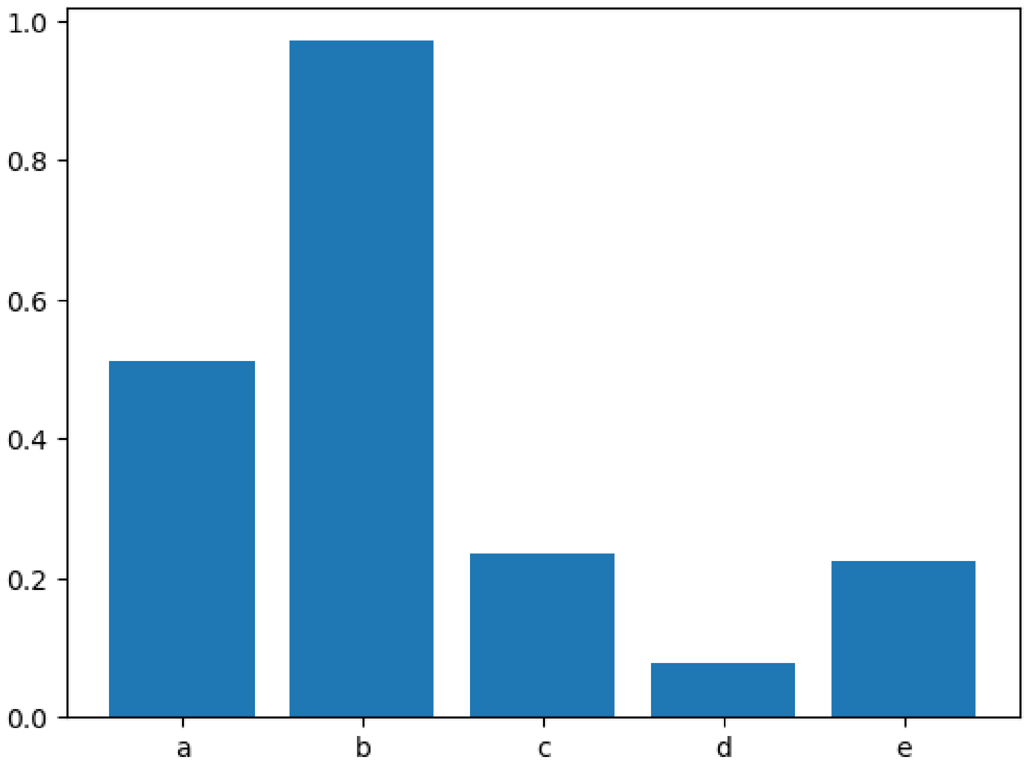
```
# Here we generate the 5 numerical data

y= np.random.rand(5)
```

[21]:

```
# plt.bar ( )  function is used for presenting the bar graph  by passing the␣
  ↪values of x and y inside the function

plt.bar(x,y)
```

[21]:

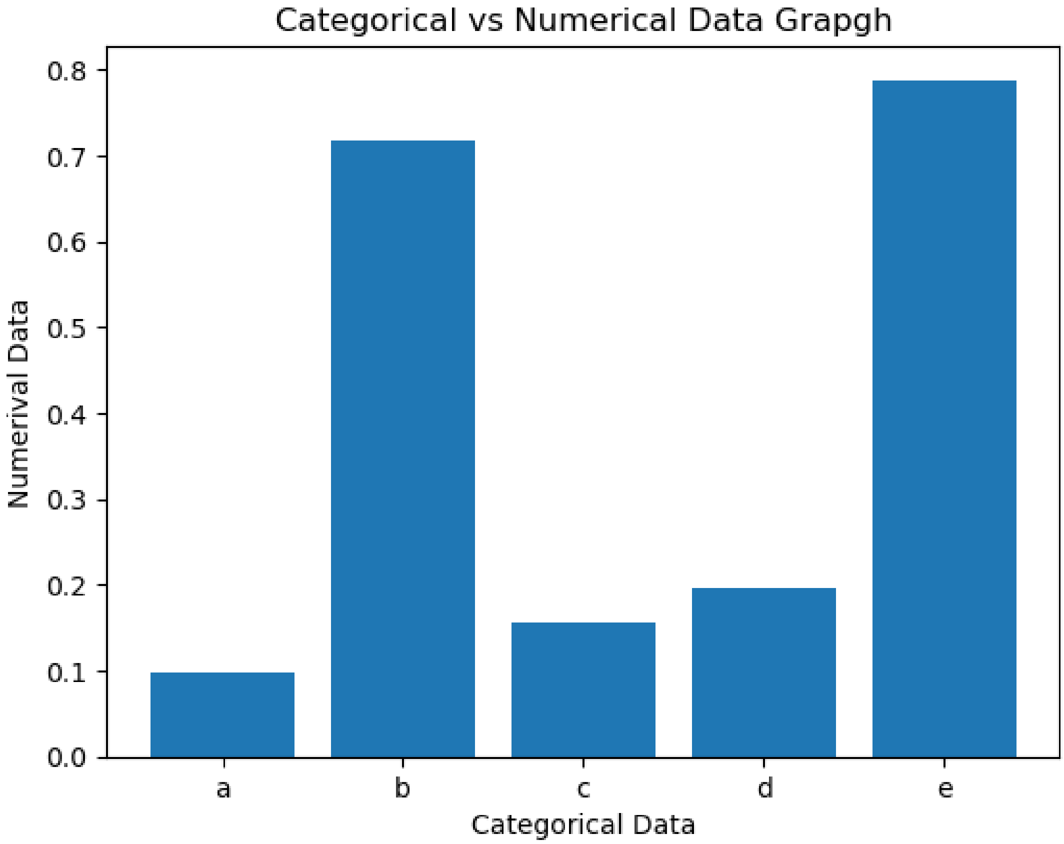
```
<BarContainer object of 5 artists>
```



OR

```
[22]: x=["a","b","c","d","e"]
      y= np.random.rand(5)
      plt.bar(x,y)
      plt.xlabel("Categorical Data")
      plt.ylabel("Numerival Data")
      plt.title("Categorical vs Numerical Data Grapgh")
```

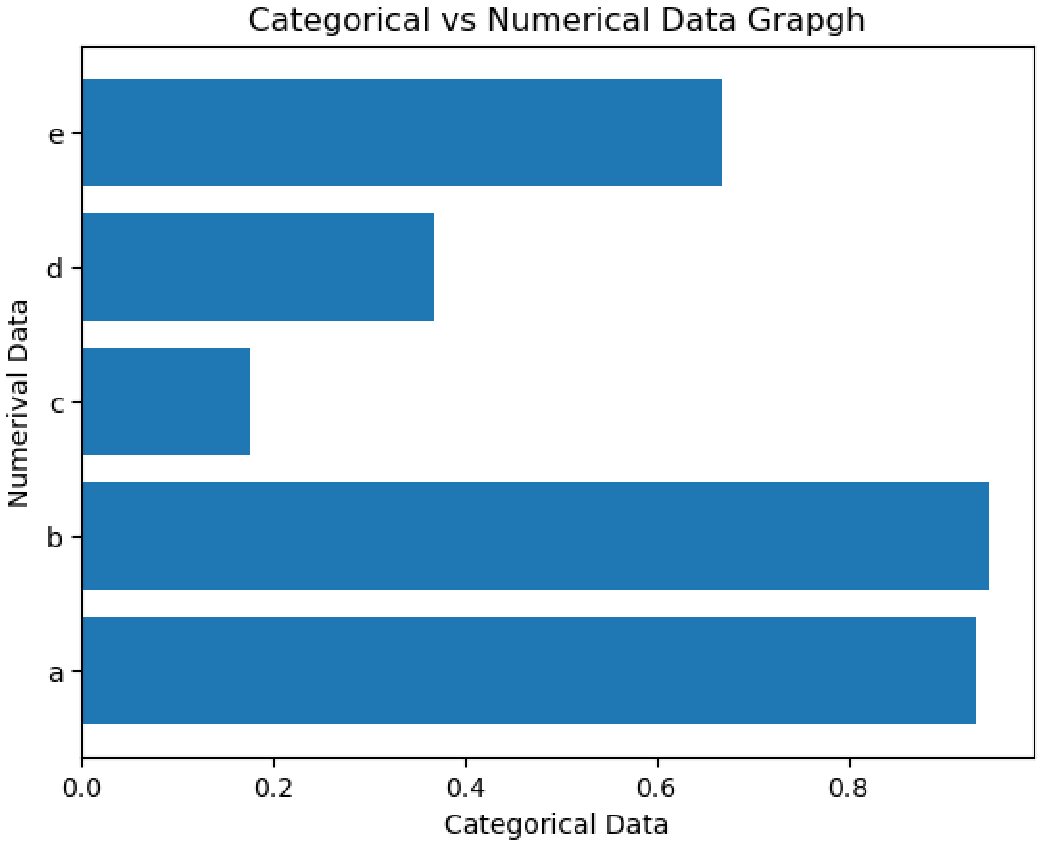
```
[22]: Text(0.5, 1.0, 'Categorical vs Numerical Data Grapgh')
```



So yeah , This is how we Draw Categorical vs Numerical Data

```
[23]: # Note - plt.barh ( ) function is used for horizontal bar plot

      x=["a","b","c","d","e"]
      y= np.random.rand(5)

      plt.barh(x,y)

      plt.xlabel("Categorical Data")
      plt.ylabel("Numerival Data")
      plt.title("Categorical vs Numerical Data Grapgh")
```

```
[23]: Text(0.5, 1.0, 'Categorical vs Numerical Data Grapgh')
```



CASE - 4

```
[24]: x=np.linspace(0,10,200)
```

```
[25]: y=np.sin(x)
```

```
[26]: x
```

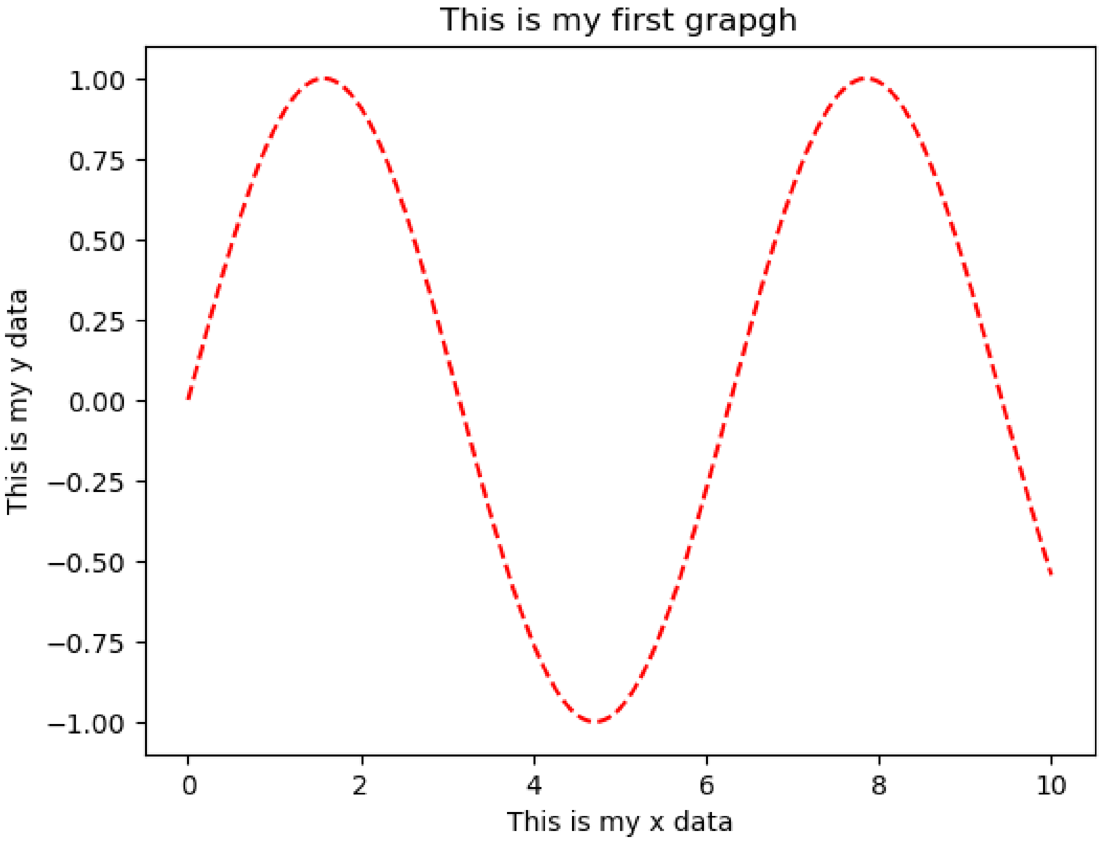
```
[26]: array([ 0.        ,  0.05025126,  0.10050251,  0.15075377,  0.20100503,
             0.25125628,  0.30150754,  0.35175879,  0.40201005,  0.45226131,
             0.50251256,  0.55276382,  0.60301508,  0.65326633,  0.70351759,
             0.75376884,  0.8040201 ,  0.85427136,  0.90452261,  0.95477387,
             1.00502513,  1.05527638,  1.10552764,  1.15577889,  1.20603015,
             1.25628141,  1.30653266,  1.35678392,  1.40703518,  1.45728643,
             1.50753769,  1.55778894,  1.6080402 ,  1.65829146,  1.70854271,
             1.75879397,  1.80904523,  1.85929648,  1.90954774,  1.95979899,
             2.01005025,  2.06030151,  2.11055276,  2.16080402,  2.21105528,
             2.26130653,  2.31155779,  2.36180905,  2.4120603 ,  2.46231156,
             2.51256281,  2.56281407,  2.61306533,  2.66331658,  2.71356784,
             2.7638191 ,  2.81407035,  2.86432161,  2.91457286,  2.96482412,
             3.01507538,  3.06532663,  3.11557789,  3.16582915,  3.2160804 ,
             3.26633166,  3.31658291,  3.36683417,  3.41708543,  3.46733668,
             3.51758794,  3.5678392 ,  3.61809045,  3.66834171,  3.71859296,
             3.76884422,  3.81909548,  3.86934673,  3.91959799,  3.96984925,
             4.0201005 ,  4.07035176,  4.12060302,  4.17085427,  4.22110553,
             4.27135678,  4.32160804,  4.3718593 ,  4.42211055,  4.47236181,
             4.52261307,  4.57286432,  4.62311558,  4.67336683,  4.72361809,
             4.77386935,  4.8241206 ,  4.87437186,  4.92462312,  4.97487437,
             5.02512563,  5.07537688,  5.12562814,  5.1758794 ,  5.22613065,
             5.27638191,  5.32663317,  5.37688442,  5.42713568,  5.47738693,
             5.52763819,  5.57788945,  5.6281407 ,  5.67839196,  5.72864322,
             5.77889447,  5.82914573,  5.87939698,  5.92964824,  5.9798995 ,
             6.03015075,  6.08040201,  6.13065327,  6.18090452,  6.23115578,
             6.28140704,  6.33165829,  6.38190955,  6.4321608 ,  6.48241206,
             6.53266332,  6.58291457,  6.63316583,  6.68341709,  6.73366834,
             6.7839196 ,  6.83417085,  6.88442211,  6.93467337,  6.98492462,
             7.03517588,  7.08542714,  7.13567839,  7.18592965,  7.2361809 ,
             7.28643216,  7.33668342,  7.38693467,  7.43718593,  7.48743719,
             7.53768844,  7.5879397 ,  7.63819095,  7.68844221,  7.73869347,
             7.78894472,  7.83919598,  7.88944724,  7.93969849,  7.98994975,
             8.04020101,  8.09045226,  8.14070352,  8.19095477,  8.24120603,
             8.29145729,  8.34170854,  8.3919598 ,  8.44221106,  8.49246231,
             8.54271357,  8.59296482,  8.64321608,  8.69346734,  8.74371859,
             8.79396985,  8.84422111,  8.89447236,  8.94472362,  8.99497487,
             9.04522613,  9.09547739,  9.14572864,  9.1959799 ,  9.24623116,
             9.29648241,  9.34673367,  9.39698492,  9.44723618,  9.49748744,
             9.54773869,  9.59798995,  9.64824121,  9.69849246,  9.74874372,
             9.79899497,  9.84924623,  9.89949749,  9.94974874, 10.        ])
```

```
[27]: y
```

```
[27]: array([ 0.        ,  0.05023011,  0.10033341,  0.15018339,  0.19965422,
        0.24862099,  0.29696008,  0.34454944,  0.39126893,  0.43700061,
        0.481629  ,  0.52504145,  0.56712835,  0.60778345,  0.6469041 ,
        0.68439153,  0.72015112,  0.75409257,  0.78613019,  0.8161831 ,
        0.84417544,  0.87003651,  0.89370105,  0.91510929,  0.9342072 ,
        0.95094655,  0.96528509,  0.97718662,  0.98662108,  0.99356467,
        0.99799984,  0.99991541,  0.99930653,  0.99617474,  0.99052796,
        0.98238043,  0.97175273,  0.95867168,  0.94317032,  0.92528777,
        0.90506919,  0.88256563,  0.85783388,  0.8309364 ,  0.80194109,
        0.77092115,  0.7379549 ,  0.70312557,  0.66652108,  0.62823386,
        0.58836056,  0.54700186,  0.50426216,  0.46024937,  0.41507461,
        0.36885193,  0.32169803,  0.27373195,  0.22507478,  0.17584939,
        0.12618003,  0.07619211,  0.02601183, -0.02423412, -0.07441889,
       -0.12441577, -0.17409855, -0.22334179, -0.27202116, -0.32001378,
       -0.36719847, -0.41345611, -0.45866992, -0.50272574, -0.54551235,
       -0.58692173, -0.62684933, -0.66519435, -0.70185999, -0.73675367,
       -0.7697873 , -0.80087747, -0.82994571, -0.85691862, -0.88172811,
       -0.90431153, -0.92461187, -0.94257789, -0.95816422, -0.97133152,
       -0.98204653, -0.99028221, -0.99601778, -0.99923873, -0.99993695,
       -0.99811068, -0.99376451, -0.98690943, -0.97756275, -0.96574805,
       -0.95149517, -0.93484009, -0.91582485, -0.89449748, -0.8709118 ,
       -0.84512737, -0.81720929, -0.78722803, -0.75525929, -0.72138377,
       -0.68568702, -0.64825913, -0.60919462, -0.56859209, -0.52655407,
       -0.48318668, -0.4385994 , -0.39290482, -0.34621828, -0.29865766,
       -0.25034303, -0.20139637, -0.15194126, -0.10210255, -0.05200606,
       -0.00177827,  0.048454  ,  0.09856395,  0.14842506,  0.19791144,
        0.24689816,  0.29526155,  0.34287951,  0.38963181,  0.43540043,
        0.48006981,  0.52352718,  0.56566282,  0.60637036,  0.64554701,
        0.68309389,  0.71891618,  0.75292346,  0.78502987,  0.81515434,
        0.84322083,  0.86915847,  0.89290179,  0.91439084,  0.93357136,
        0.95039493,  0.96481908,  0.9768074 ,  0.98632961,  0.99336168,
        0.99788585,  0.99989069,  0.99937116,  0.99632856,  0.99077057,
        0.98271122,  0.97217086,  0.95917611,  0.94375976,  0.92596075,
        0.905824  ,  0.88340035,  0.85874643,  0.83192446,  0.80300216,
        0.77205257,  0.7391538 ,  0.70438892,  0.66784571,  0.62961641,
        0.58979754,  0.54848964,  0.50579699,  0.46182738,  0.41669181,
        0.37050423,  0.32338126,  0.27544187,  0.22680707,  0.17759967,
        0.12794389,  0.07796509,  0.02778946, -0.02245633, -0.07264543,
       -0.12265112, -0.17234716, -0.22160808, -0.27030952, -0.31832851,
       -0.36554384, -0.4118363 , -0.45708901, -0.50118772, -0.54402111])
```

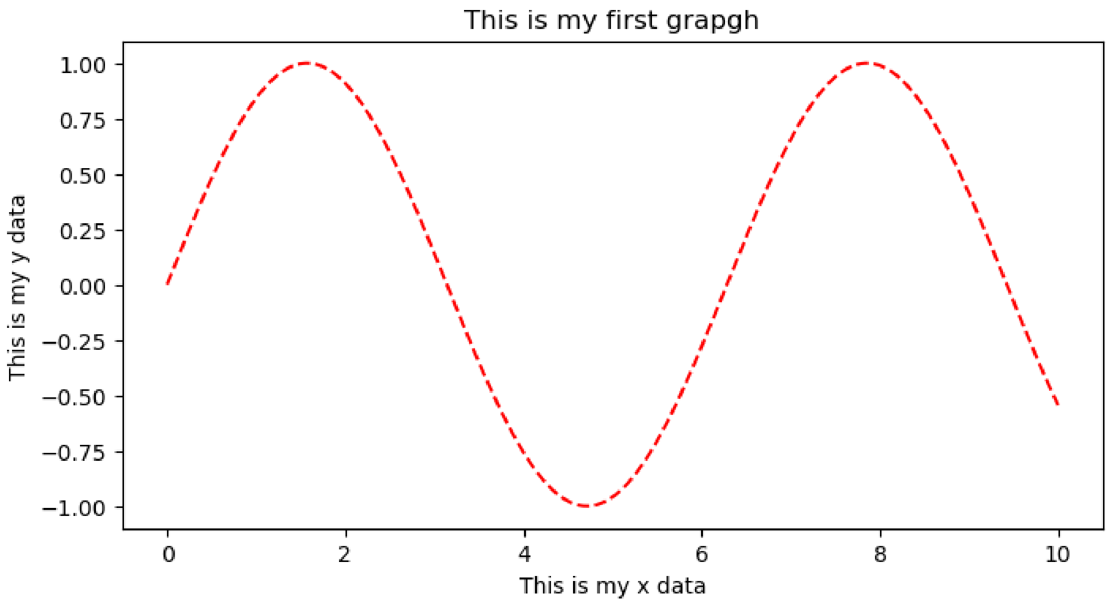
[28]:
```
# plt.plot (" - - r " ) , in this function double hyphen (--) represent the dot␣
 ↪in graph and " r " represent the colour of line grapgh that is r (red)

plt.plot(x,y,"--r")


plt.title("This is my first grapgh")
plt.xlabel("This is my x data")
plt.ylabel("This is my y data")
```

[28]:
```
Text(0, 0.5, 'This is my y data')
```



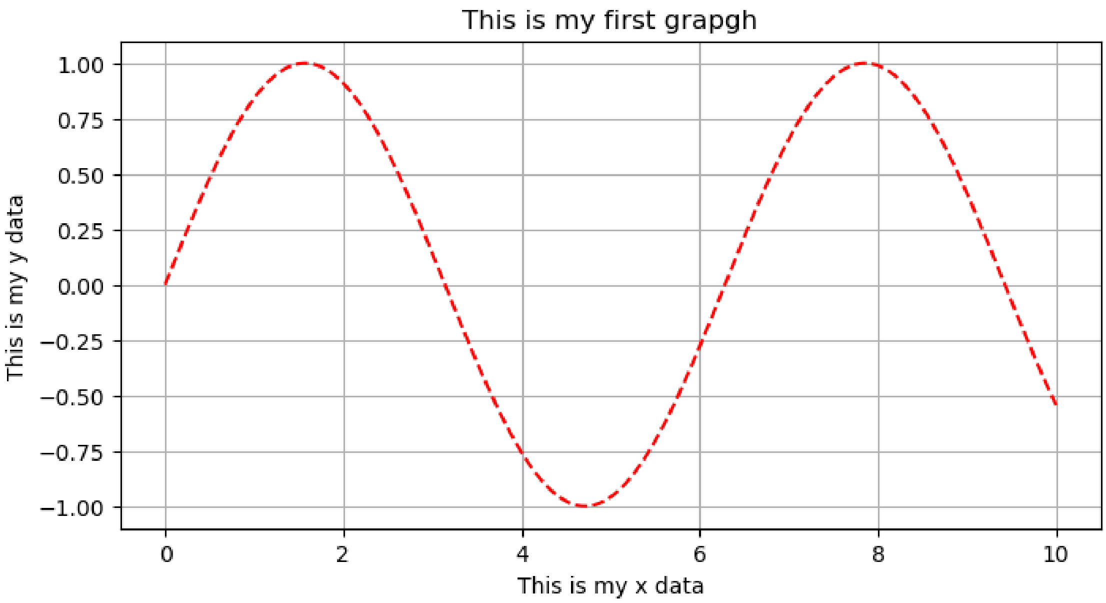
```
[29]: # We cantrol the sixe of figure as well

      plt.figure(figsize=(8,4))
      plt.plot(x,y,"--r")
      plt.title("This is my first grapgh")
      plt.xlabel("This is my x data")
      plt.ylabel("This is my y data")
```

```
[29]: Text(0, 0.5, 'This is my y data')
```



```
[30]: # plt.grid ( ) function is used for grid background kind of a box background␣
       ↪behind the grapgh

      plt.figure(figsize=(8,4))
      plt.plot(x,y,"--r")
      plt.grid()
      plt.title("This is my first grapgh")
      plt.xlabel("This is my x data")
      plt.ylabel("This is my y data")
```

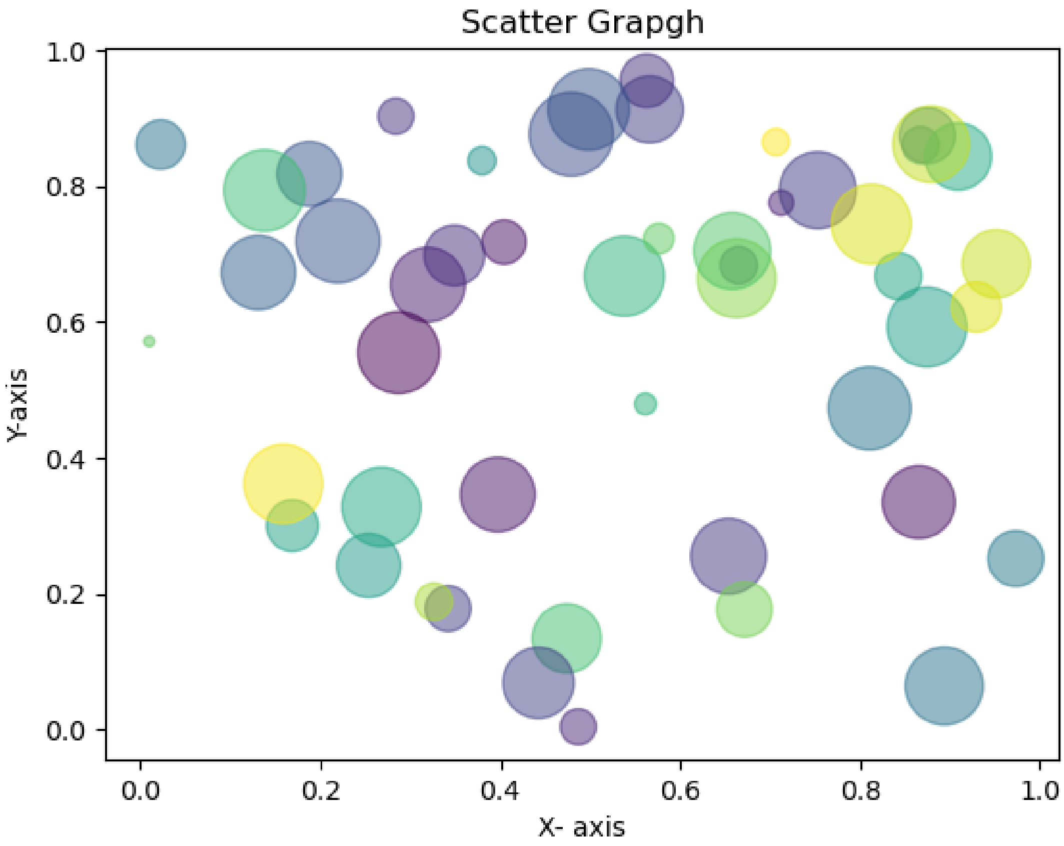
```
[30]: Text(0, 0.5, 'This is my y data')
```



CASE - 5

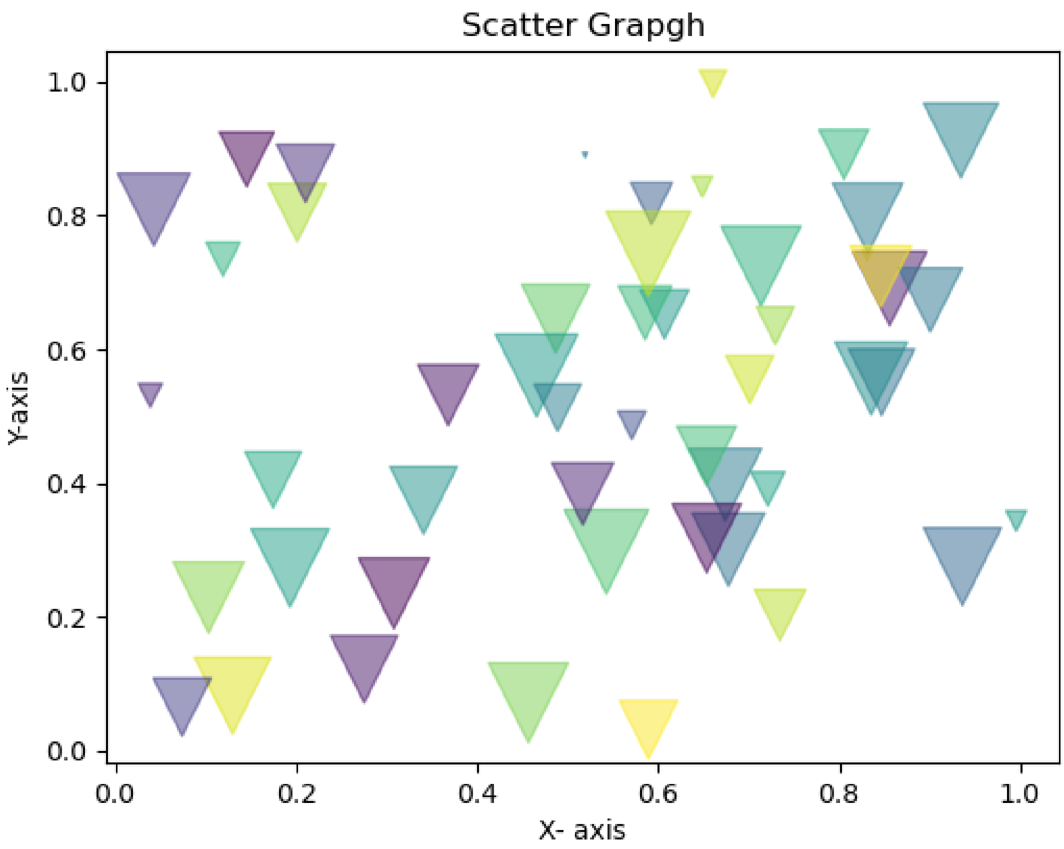
```
[31]: x=np.random.rand(50)
      y=np.random.rand(50)
      colours=np.random.rand(50)
      sizes=1000*np.random.rand(50)
      plt.scatter(x,y,c=colours,s=sizes,alpha=.5)
      plt.xlabel("X- axis")
      plt.ylabel("Y-axis")
      plt.title("Scatter Grapgh")
```

```
[31]: Text(0.5, 1.0, 'Scatter Grapgh')
```



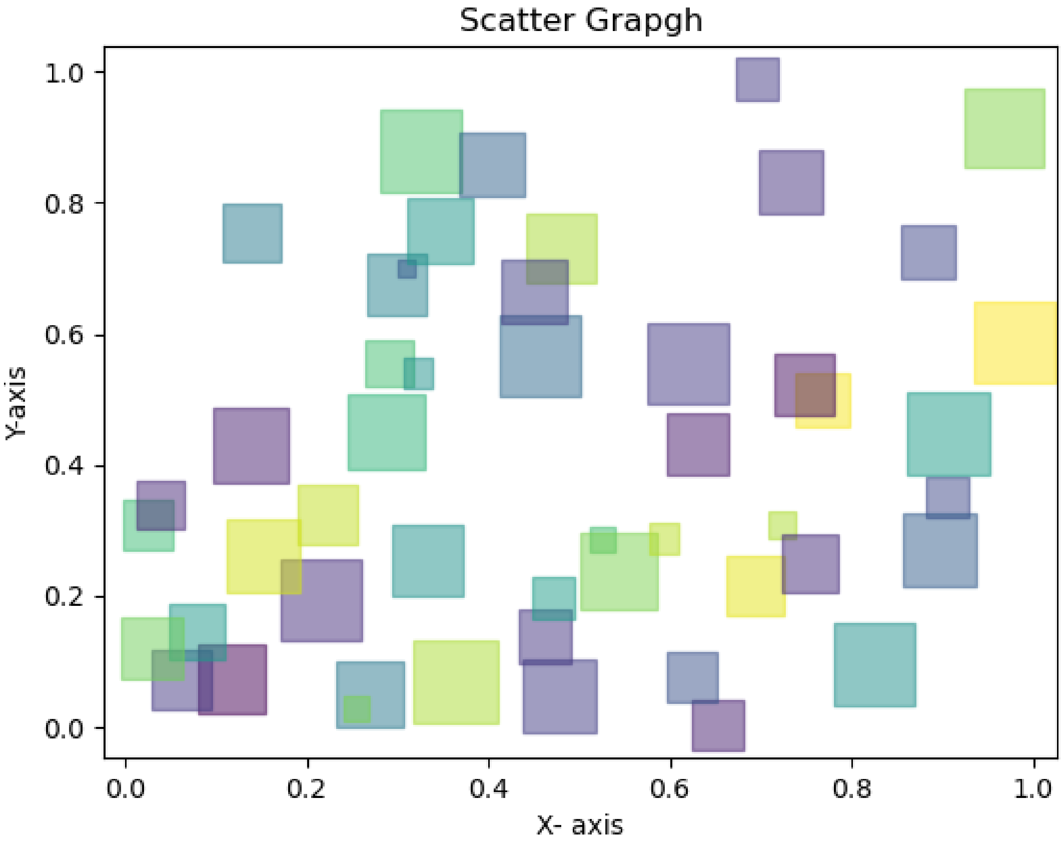
[32]:
```
# Marker function for changing the design of dots inside the grapgh

x=np.random.rand(50)
y=np.random.rand(50)
colours=np.random.rand(50)
sizes=1000*np.random.rand(50)
plt.scatter(x,y,c=colours,s=sizes,alpha=.5 , marker="v")
plt.xlabel("X- axis")
plt.ylabel("Y-axis")
plt.title("Scatter Grapgh")
```

[32]:
```
Text(0.5, 1.0, 'Scatter Grapgh')
```



[33]:
```
# Marker function for changing the design of dots inside the grapgh

x=np.random.rand(50)
y=np.random.rand(50)
colours=np.random.rand(50)
sizes=1000*np.random.rand(50)
plt.scatter(x,y,c=colours,s=sizes,alpha=.5 , marker="s")
plt.xlabel("X- axis")
plt.ylabel("Y-axis")
plt.title("Scatter Grapgh")
```

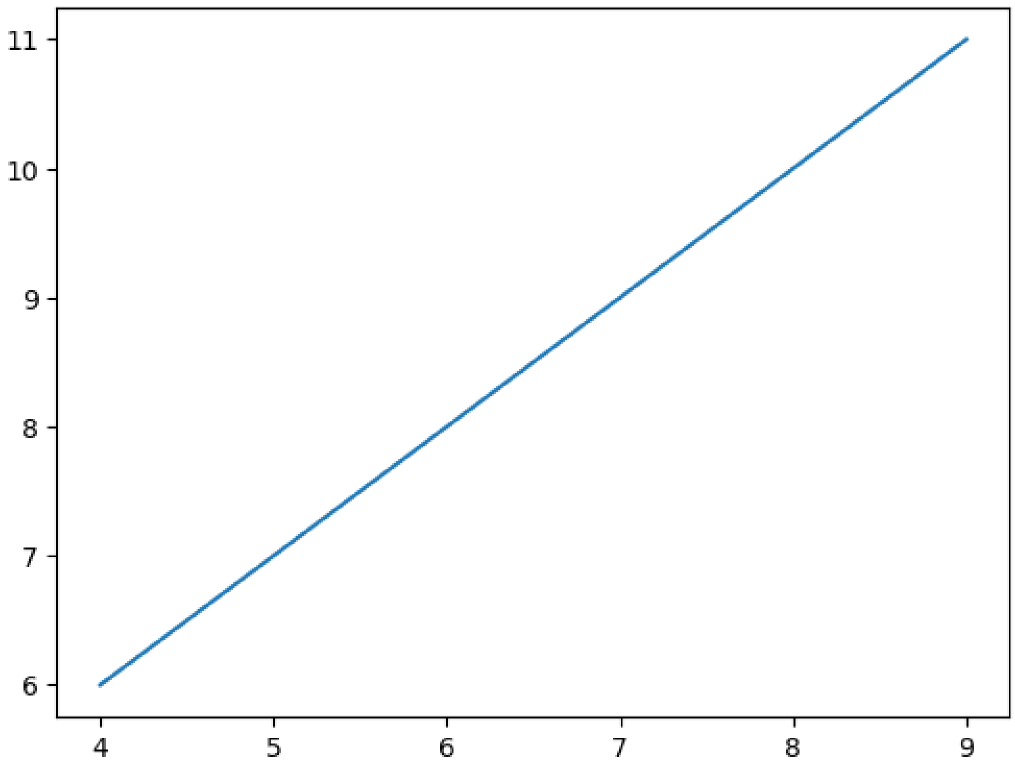
[33]:
```
Text(0.5, 1.0, 'Scatter Grapgh')
```



CASE - 6

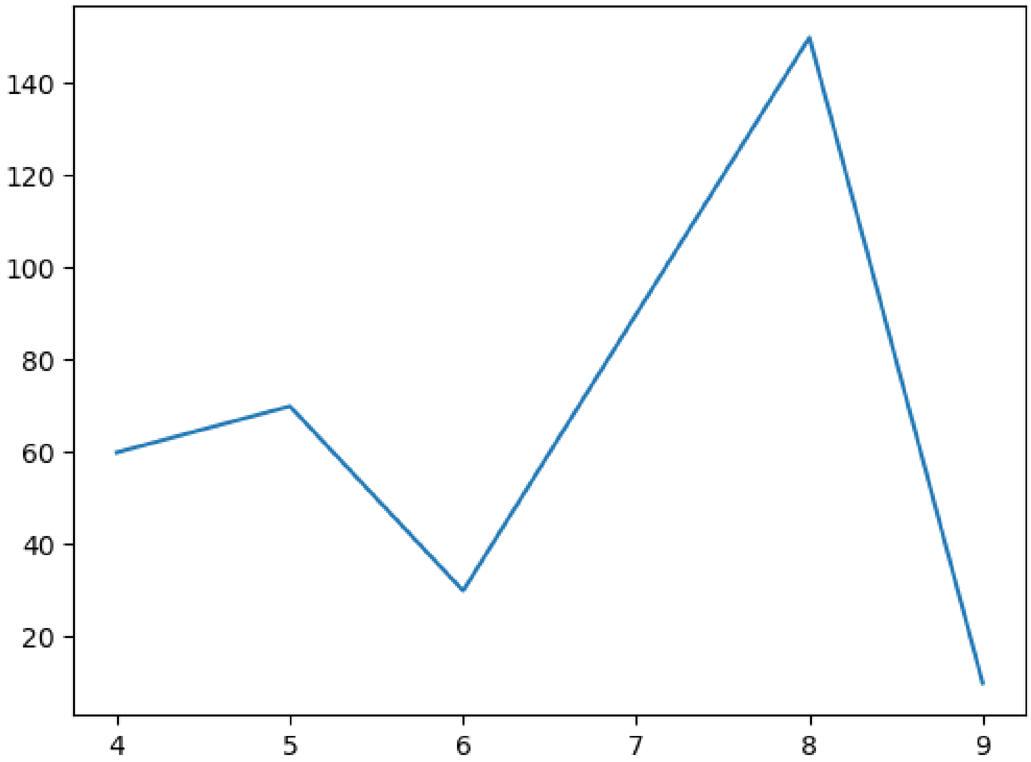
[36]:

```
x=[4,5,6,7,8,9]
y=[6,7,8,9,10,11]
plt.plot(x,y)
plt.show()
```



[42]:

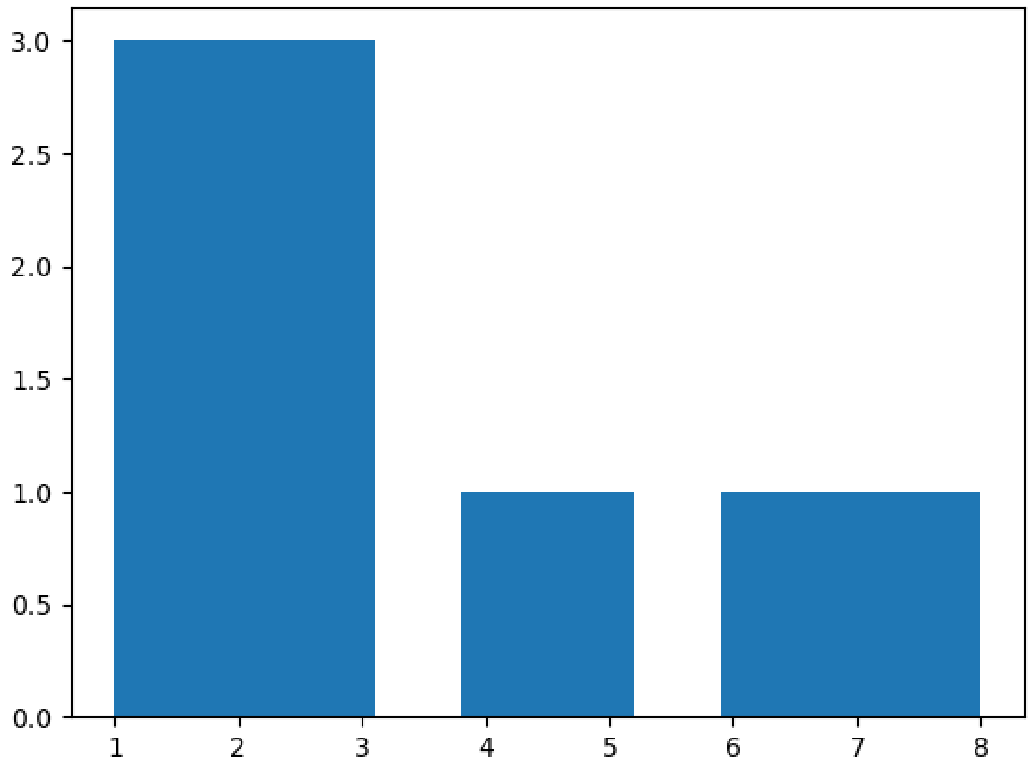
```
x=[4,5,6,7,8,9]
y=[60,70,30,90,150,10]
plt.plot(x,y)
plt.show()
```



CASE - 7

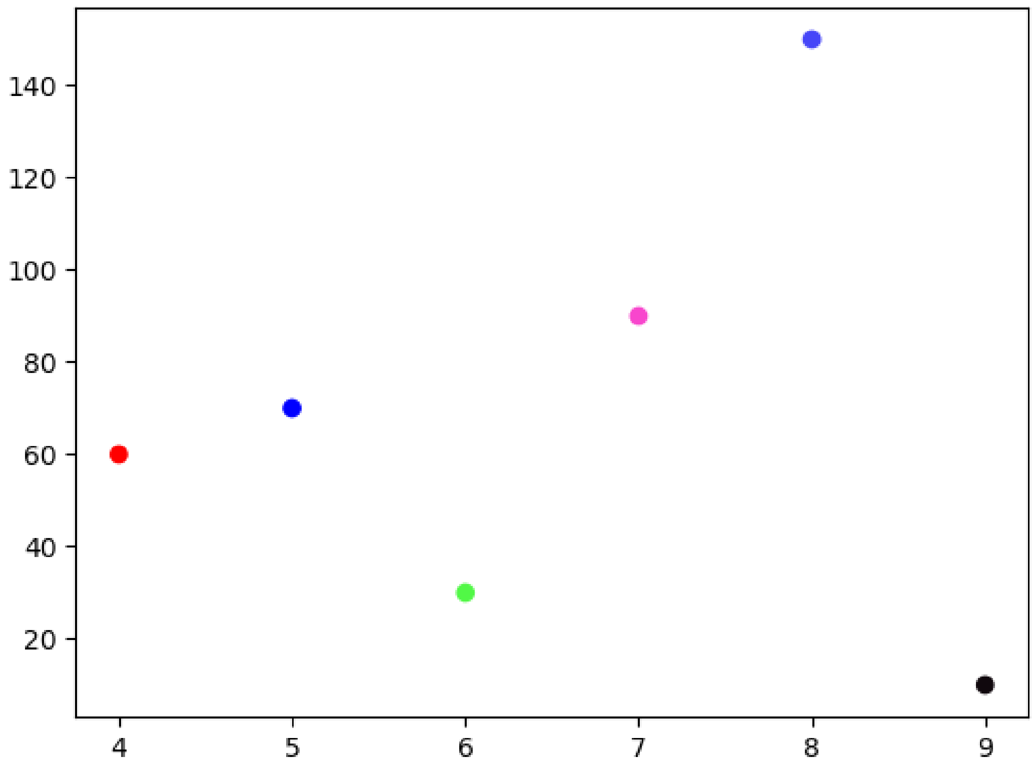
```
[52]: data=[1,2,3,4,1,2,3,7,8,1,2,3,5,6]
```

```
[53]: plt.hist(data)
      plt.show()
```



[57]:

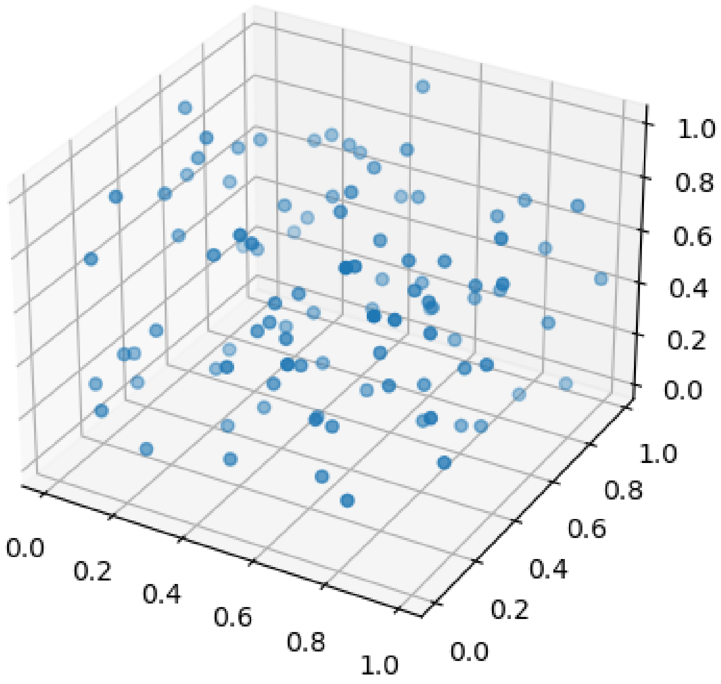
```
x=[4,5,6,7,8,9]
y=[60,70,30,90,150,10]
colour=["red","blue","#51f846","#f846cd","#4646f8","#10080d"]
plt.scatter(x,y,c=colour)
plt.show()
```



CASE - 8

[65]:

```
# 3D DATA PROJECTION

x=np.random.rand(100)
y=np.random.rand(100)
z=np.random.rand(100)

fig=plt.figure()

# THIS FUNCTION IS USED FOR MAKING 3D PROJECTION GRAPGH

ax=fig.add_subplot(projection="3d")
ax.scatter(x,y,z)
plt.show()
```



THANK YOU SO MUCH !!

YOURS VIRAT TIWARI :)