Matplotlib and Seaborn Guide

Installing the libraries

Before we get started, we need to install the libraries. To install Matplotlib and Seaborn, you can use the following command in your terminal or command prompt:

In [ ]:
```
pip install matplotlib seaborn
```

```
Requirement already satisfied: matplotlib in d:\anaconda\lib\site-packages (3.5.1)
Requirement already satisfied: seaborn in d:\anaconda\lib\site-packages (0.11.2)
Requirement already satisfied: pillow>=6.2.0 in d:\anaconda\lib\site-packages (from matplotlib) (9.0.1)
Requirement already satisfied: pyparsing>=2.2.1 in d:\anaconda\lib\site-packages (from matplotlib) (3.0.4)
Requirement already satisfied: numpy>=1.17 in d:\anaconda\lib\site-packages (from matplotlib) (1.21.5)
Requirement already satisfied: cycler>=0.10 in d:\anaconda\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: python-dateutil>=2.7 in d:\anaconda\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: packaging>=20.0 in d:\anaconda\lib\site-packages (from matplotlib) (21.3)
Requirement already satisfied: fonttools>=4.22.0 in d:\anaconda\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in d:\anaconda\lib\site-packages (from matplotlib) (1.3.2)
Requirement already satisfied: pandas>=0.23 in d:\anaconda\lib\site-packages (from seaborn) (1.4.2)
Requirement already satisfied: scipy>=1.0 in d:\anaconda\lib\site-packages (from seaborn) (1.7.3)
Requirement already satisfied: pytz>=2020.1 in d:\anaconda\lib\site-packages (from pandas>=0.23->seaborn) (2021
.3)
Requirement already satisfied: six>=1.5 in d:\anaconda\lib\site-packages (from python-dateutil>=2.7->matplotlib
) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

Importing the libraries

Once you have installed the libraries, you can import them into your Jupyter notebook using the following code:

In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

Matplotlib is a data visualization library in Python that provides a variety of plotting functions and tools for creating interactive plots, charts, and graphs. It is one of the most widely used libraries for data visualization and is a preferred choice for many data analysts, scientists, and researchers.

Matplotlib can be used for various purposes, including exploratory data analysis, creating informative visualizations for research papers, and producing high-quality plots for presentations. With Matplotlib, you can create various types of plots, such as line plots, scatter plots, histograms, bar plots, and many more.

Some of the common plots that can be plotted using the Pyplot module of Matplotlib are:

- **Line Plot:** A line plot is a graph that shows the relationship between two variables using lines to connect data points. It is typically used to display trends over time or to show the relationship between two continuous variables.
- **Scatter Plot:** A scatter plot is a graph that shows the relationship between two variables as a series of points. It is commonly used to explore the relationship between two continuous variables.
- **Histogram:** A histogram is a graph that shows the distribution of a set of continuous data. It is commonly used to visualize the frequency distribution of data.
- **Bar Plot:** A bar plot is a graph that shows the distribution of categorical data using rectangular bars. It is commonly used to compare the frequency of different categories.
- **Pie Chart:** A pie chart is a circular graph that shows the proportional distribution of categorical data. It is commonly used to visualize the relative frequency or proportion of different categories.

In [ ]:
```
import numpy as np
```

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

In [ ]:
```
x
```

Out[ ]:
```
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.        ])
```

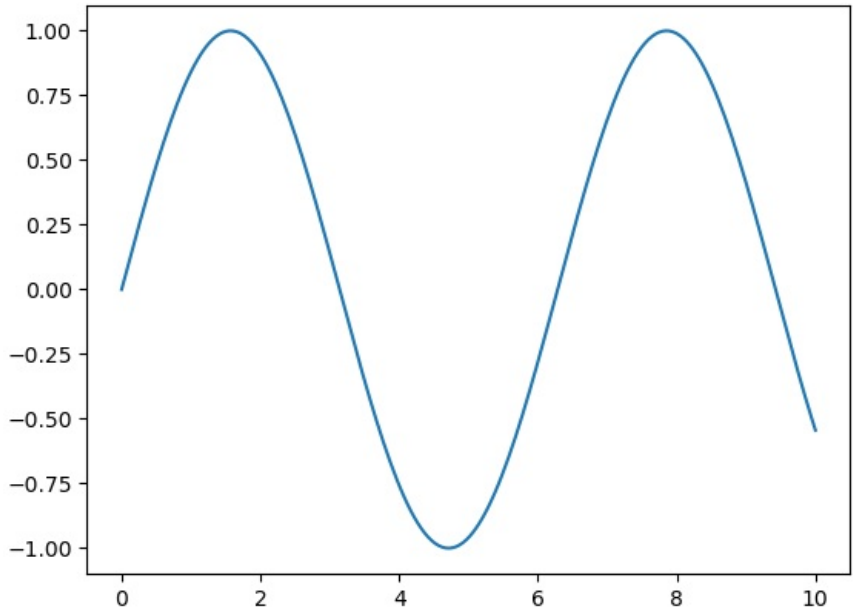
In [ ]:
```
y = np.sin(x)
```

In [ ]:
```
y
```

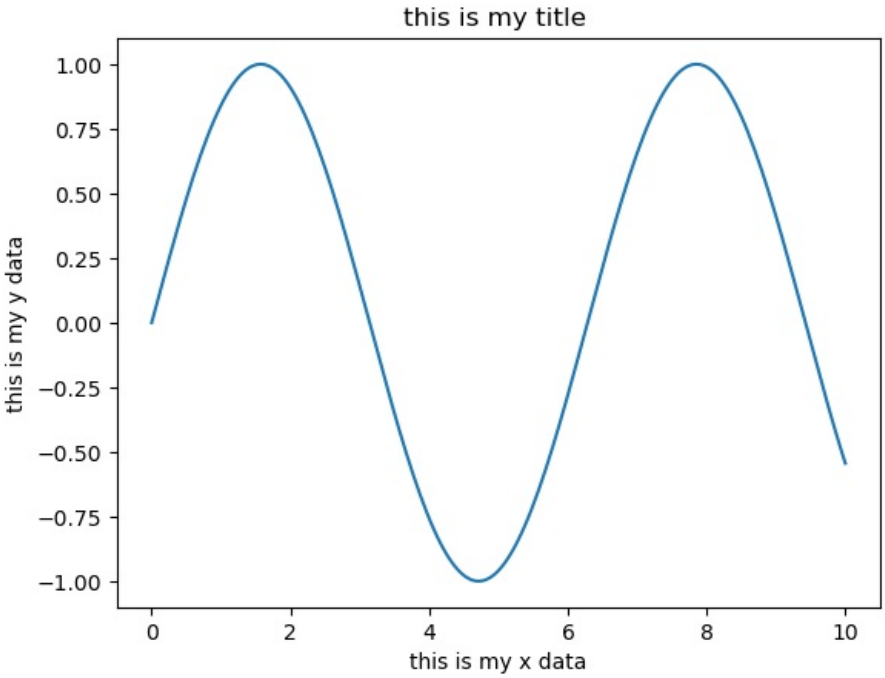
Out[ ]:
```
array([ 0.        ,  0.05023011,  0.10033341,  0.15018339,  0.19965422,
        0.24862099,  0.29696008,  0.34454944,  0.39126893,  0.43700061,
        0.481629  ,  0.52504145,  0.56712835,  0.60778345,  0.6469041 ,
        0.68439153,  0.72015112,  0.75409257,  0.78613019,  0.8161831 ,
        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])
```

In [ ]:
```
plt.plot(x,y)
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x7f54b7a42b90>]
```

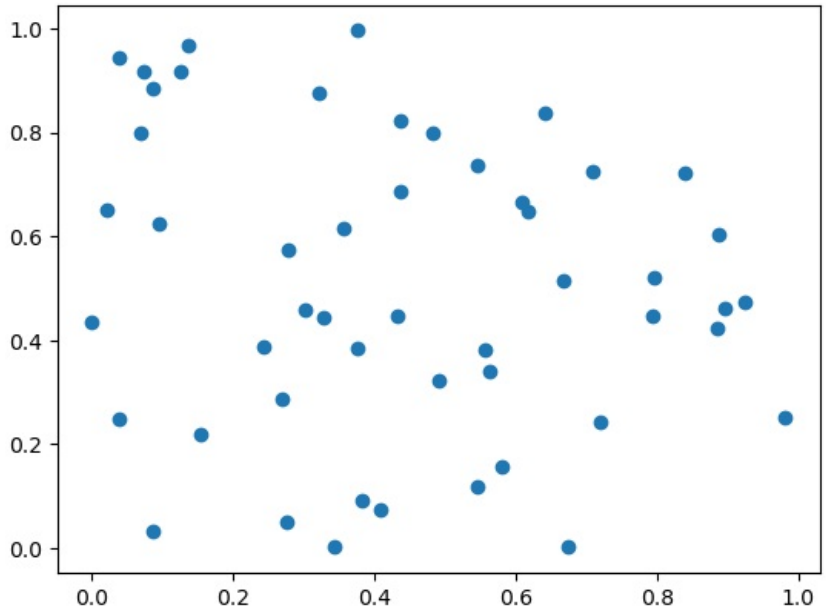


In [ ]:
```
plt.plot(x,y)
plt.xlabel("this is my x data")
plt.ylabel("this is my y data")
plt.title("this is my title ")
plt.show()
```



In [ ]:
```
x = np.random.rand(50)
y = np.random.rand(50)
plt.scatter(x,y)
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x7f54b7c7b040>
```



In [ ]:
```
x
```

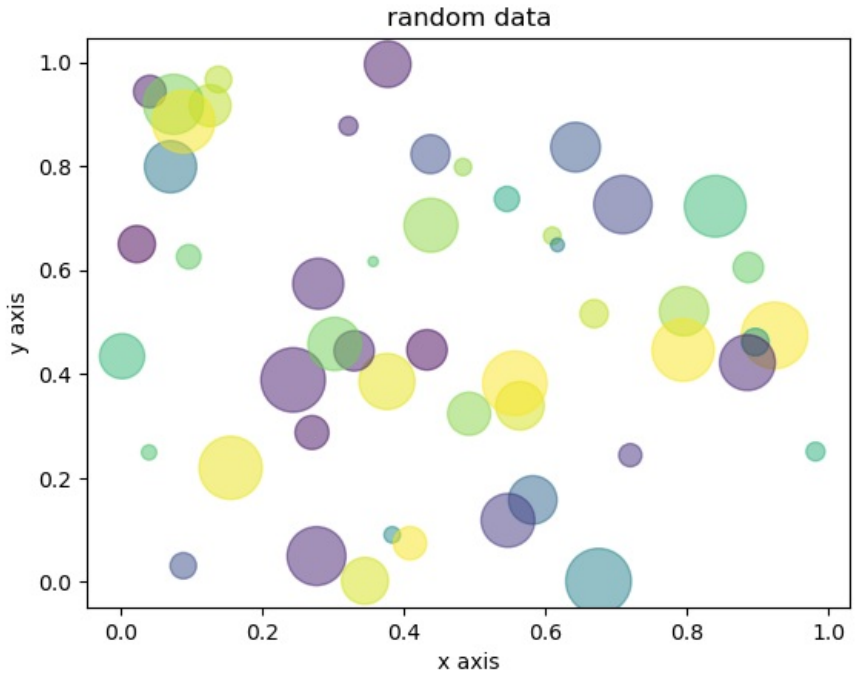
Out[ ]:
```
array([0.24304743, 0.79579648, 0.92378225, 0.3757161 , 0.32927981,
       0.04036392, 0.00115994, 0.58192904, 0.43231663, 0.7944984 ,
       0.13743848, 0.32121516, 0.27608543, 0.34461334, 0.54534119,
       0.06967744, 0.60948743, 0.54683463, 0.35624462, 0.89667967,
       0.1546341 , 0.30164764, 0.61687617, 0.88545664, 0.09532487,
       0.4832183 , 0.64243473, 0.98178848, 0.27855781, 0.12554796,
       0.67493402, 0.07395121, 0.08772712, 0.26970453, 0.56381406,
       0.02202262, 0.70943362, 0.88680141, 0.71983144, 0.38317903,
       0.43721952, 0.03930287, 0.84000702, 0.55651614, 0.49189924,
       0.6686886 , 0.40838948, 0.37679403, 0.43798099, 0.08822041])
```

In [ ]:
```
y
```

Out[ ]:
```
array([0.38854622, 0.52064332, 0.47413296, 0.38565107, 0.4445032 ,
       0.94324526, 0.43421803, 0.15772544, 0.44640805, 0.44582311,
       0.96630056, 0.87704397, 0.04980471, 0.00221969, 0.73661168,
       0.79856524, 0.66572844, 0.11856373, 0.61626211, 0.46135107,
       0.21950252, 0.45781041, 0.6482492 , 0.42182184, 0.62542343,
       0.79805752, 0.8362465 , 0.25086647, 0.57359611, 0.91612245,
       0.00157445, 0.91868597, 0.03121363, 0.28747113, 0.33903794,
       0.64976202, 0.72588497, 0.60490325, 0.24387102, 0.09069224,
       0.82279977, 0.24918024, 0.7226623 , 0.3822402 , 0.32354921,
       0.5159748 , 0.07461664, 0.99581377, 0.68611335, 0.88547652])
```

In [ ]:
```
colours = np.random.rand(50)
sizes = 1000* np.random.rand(50)
plt.scatter(x,y , c = colours , s = sizes , alpha=.5) # alpha used for intenci
plt.xlabel("x axis")
plt.ylabel("y axis")
plt.title("random data")
```

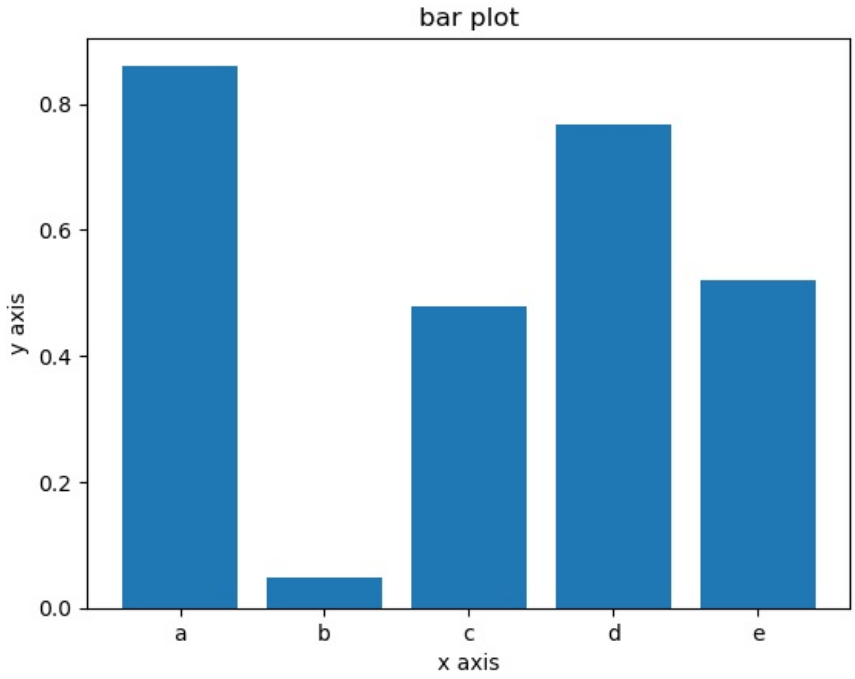
Out[ ]:
```
Text(0.5, 1.0, 'random data')
```



In [ ]:
```
x = ['a' , 'b','c','d','e']
```

In [ ]:
```
y = np.random.rand(5)
plt.bar(x,y)
plt.xlabel("x axis")
plt.ylabel("y axis")
plt.title("bar plot")
```

Out[ ]:
```
Text(0.5, 1.0, 'bar plot')
```

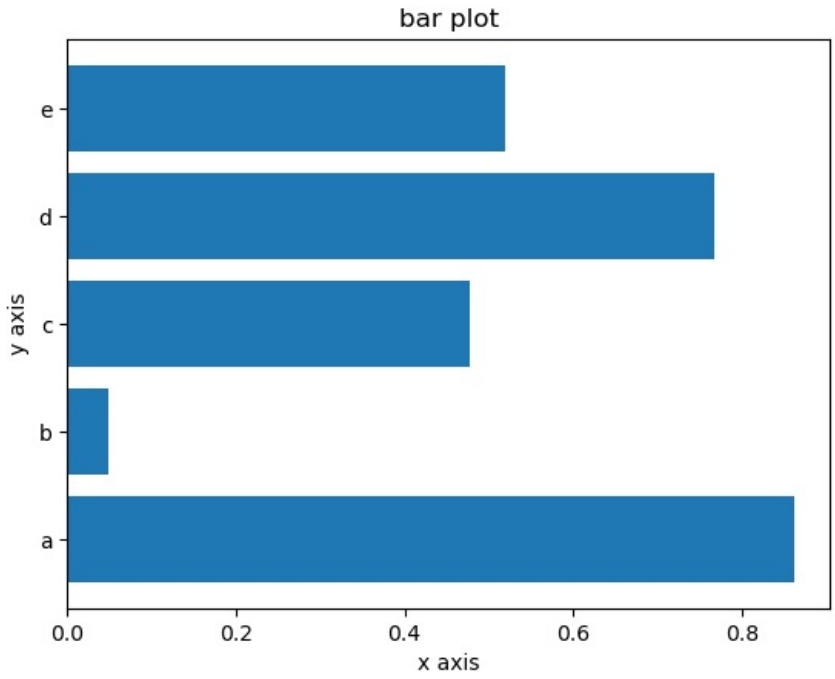


In [ ]:
```
y
```

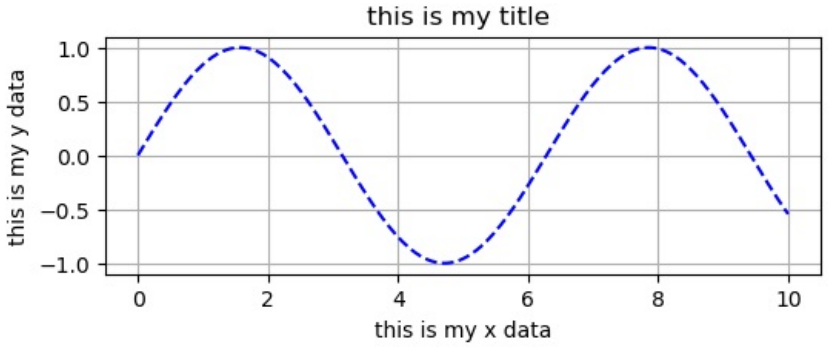
Out[ ]:
```
array([0.98399218, 0.36853523, 0.46677504, 0.03022962, 0.80728976])
```

In [ ]:
```
plt.barh(x,y)
plt.xlabel("x axis")
plt.ylabel("y axis")
plt.title("bar plot")
```

Out[ ]:
```
Text(0.5, 1.0, 'bar plot')
```

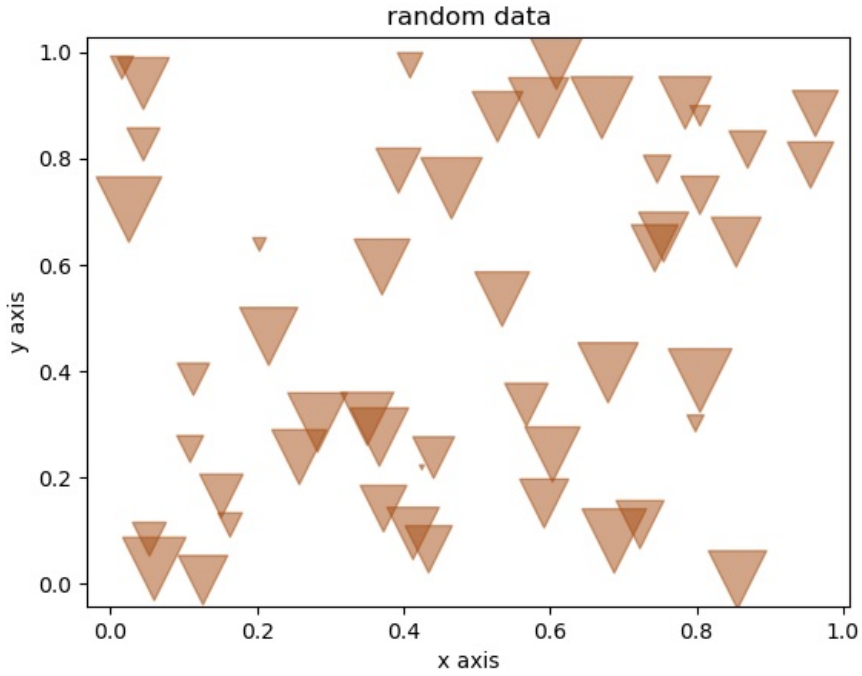


In [ ]:
```
## For sltying of graphs
plt.figure(figsize=(6,2))
plt.plot(x,y,'--b')
plt.xlabel("this is my x data")
plt.ylabel("this is my y data")
plt.title("this is my title ")
plt.grid()
plt.show()
```

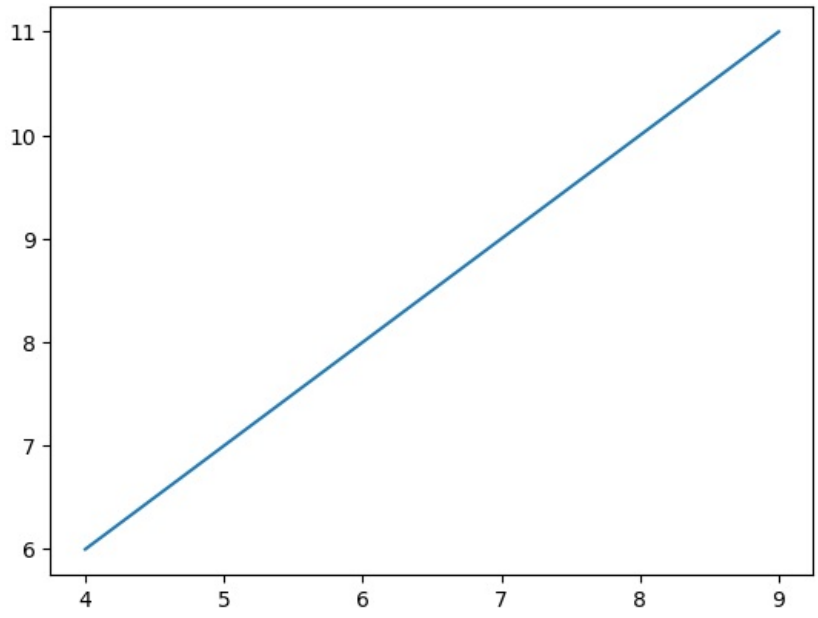


In [ ]:
```
x = np.random.rand(50)
y = np.random.rand(50)
colours = np.random.rand(50)
sizes = 1000* np.random.rand(50)
plt.scatter(x,y , c = '#A64D13' , s = sizes , alpha=.5 , marker='v')
plt.xlabel("x axis")
plt.ylabel("y axis")
plt.title("random data")
```

Out[ ]:
```
Text(0.5, 1.0, 'random data')
```

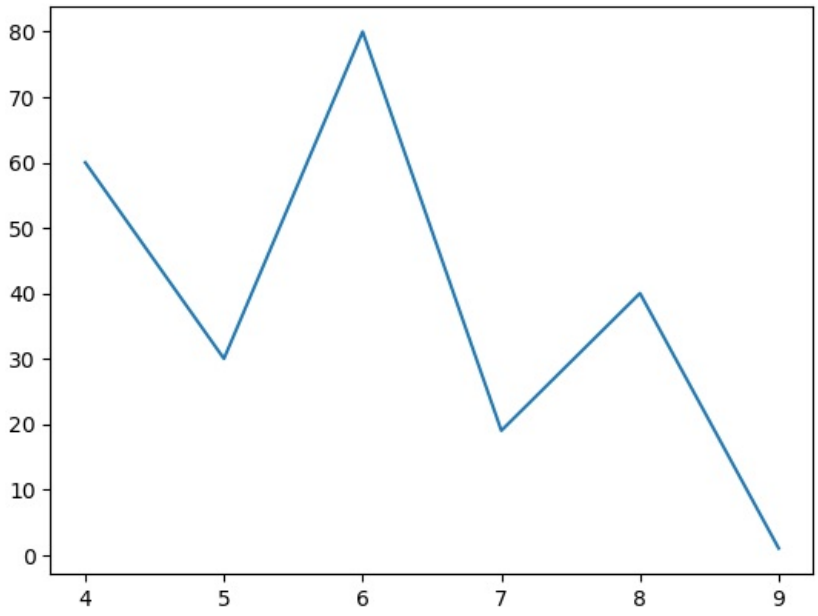


In [ ]:
```
x = [4,5,6,7,8,9]
y = [6,7,8,9,10,11]
plt.plot(x,y)
plt.show()
```

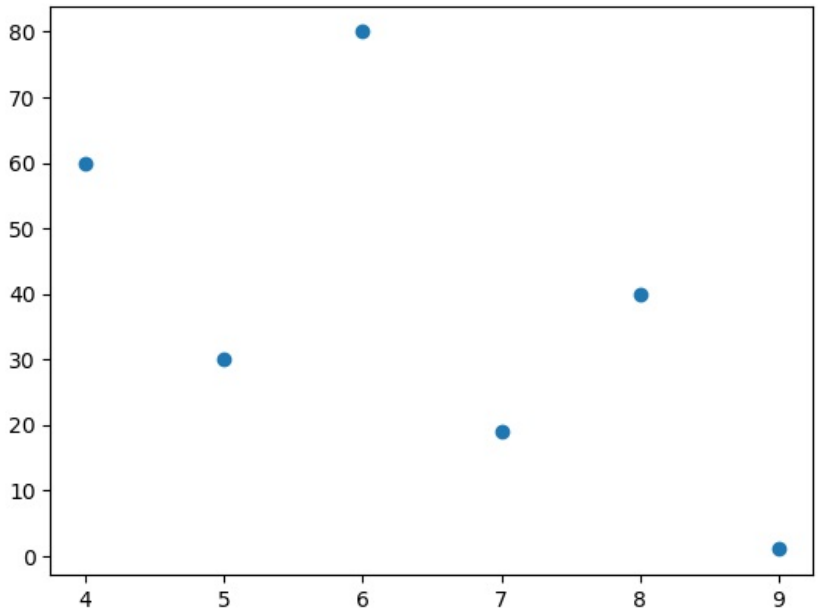


In [ ]:
```
x = [4,5,6,7,8,9]
y = [60,30,80,19,40,1]
plt.plot(x,y)
```

```
plt.show()
```

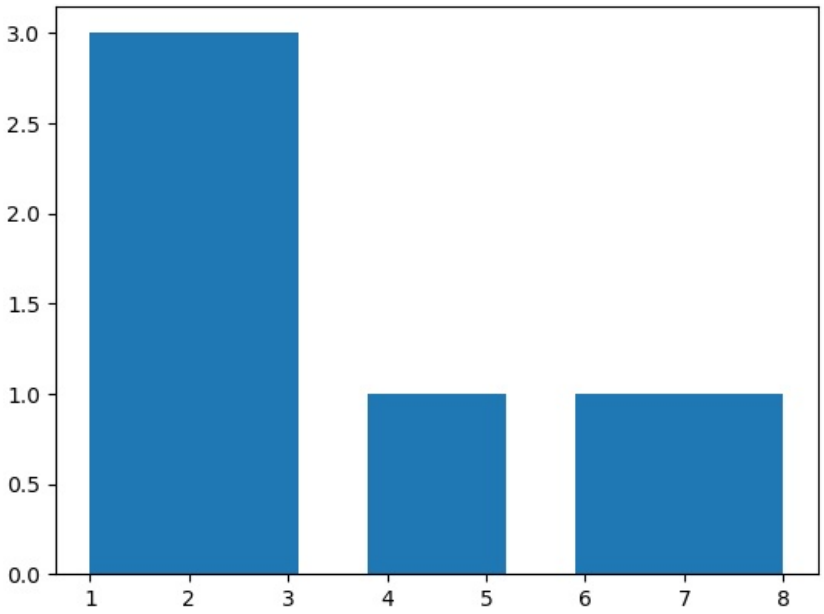


In [ ]:
```
x = [4,5,6,7,8,9]
y = [60,30,80,19,40,1]
plt.scatter(x,y)
plt.show()
```

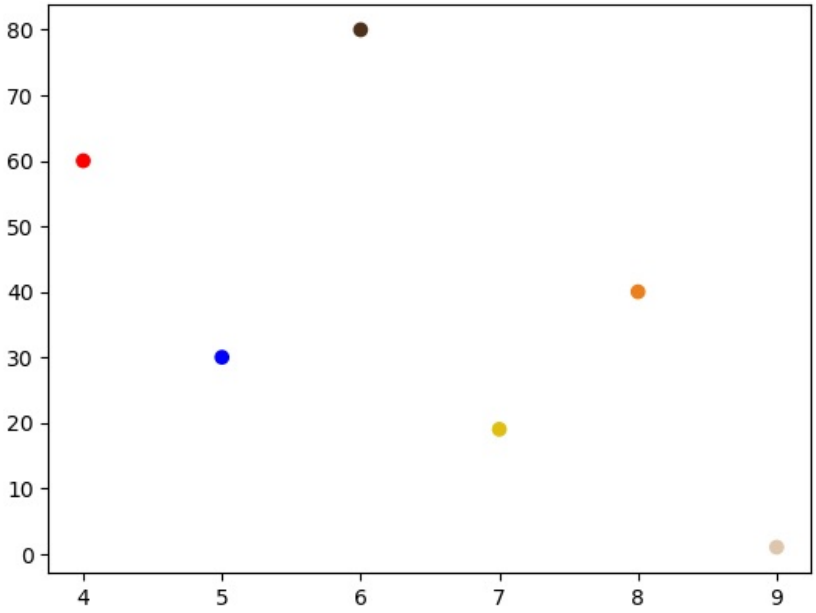


In [ ]:
```
data = [1,2,3,4,5,1,2,3,6,7,8,1,2,3]
```

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

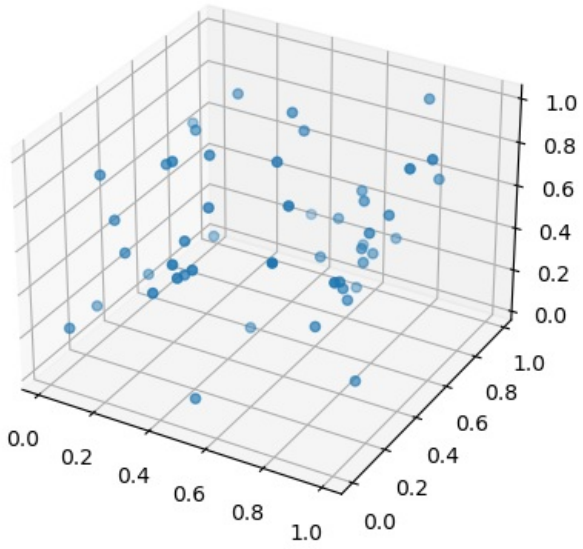


In [ ]:
```
x = [4,5,6,7,8,9]
y = [60,30,80,19,40,1]
colour = ['red','blue' , "#4E301D" , "#DFC010" , "#EC801C" , "#DFC6AE"]
plt.scatter(x,y , c = colour)
plt.show()
```



In [ ]:
```
x = np.random.rand(50)
y = np.random.rand(50)
z = np.random.rand(50)

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



Seaborn

In [ ]:
```
import seaborn as sns
```

Que 1: Name any five plots that we can plot using the Seaborn library. Also, state the uses of each plot ?

Ans) Seaborn is a popular data visualization library in Python that is built on top of Matplotlib. It provides a high-level interface for creating informative and visually appealing statistical graphics. **Some of the plots that can be created using Seaborn along with their uses are:**

- **Scatter Plot:** A scatter plot is used to show the relationship between two continuous variables. It is useful for identifying patterns or trends in the data and detecting outliers or unusual observations.
- **Line Plot:** A line plot is used to visualize the trend of a continuous variable over time or across different categories. It is useful for identifying patterns or trends in the data and for comparing the values of a variable across different categories.
- **Bar Plot:** A bar plot is used to compare the values of a categorical variable. It is useful for visualizing the frequency, count, or proportion of different categories and for identifying differences or similarities between categories.
- **Heatmap:** A heatmap is used to visualize the distribution of a continuous variable across different categories. It is useful for identifying patterns or trends in the data and for identifying the categories that have high or low values of the variable.
- **Box Plot:** A box plot is used to visualize the distribution of a continuous variable and to identify outliers. It is useful for comparing the distribution of a variable between different categories and for identifying potential outliers or extreme values.

Overall, Seaborn provides a wide range of plots that can be used for exploratory data analysis, hypothesis testing, and communication of results

In [ ]:
```
iris = sns.load_dataset('iris')
```

In [ ]:
```
iris
```

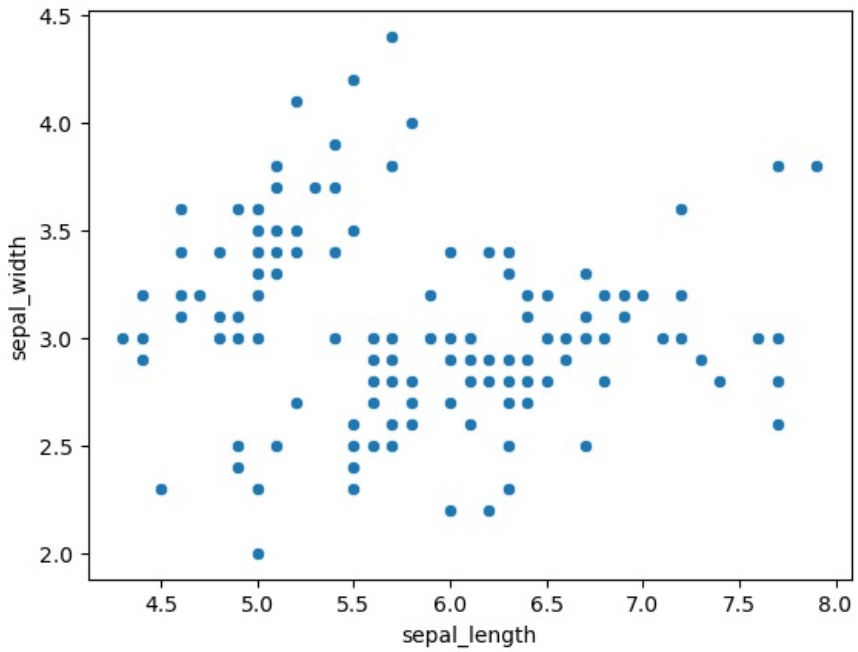
Out[ ]:

	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

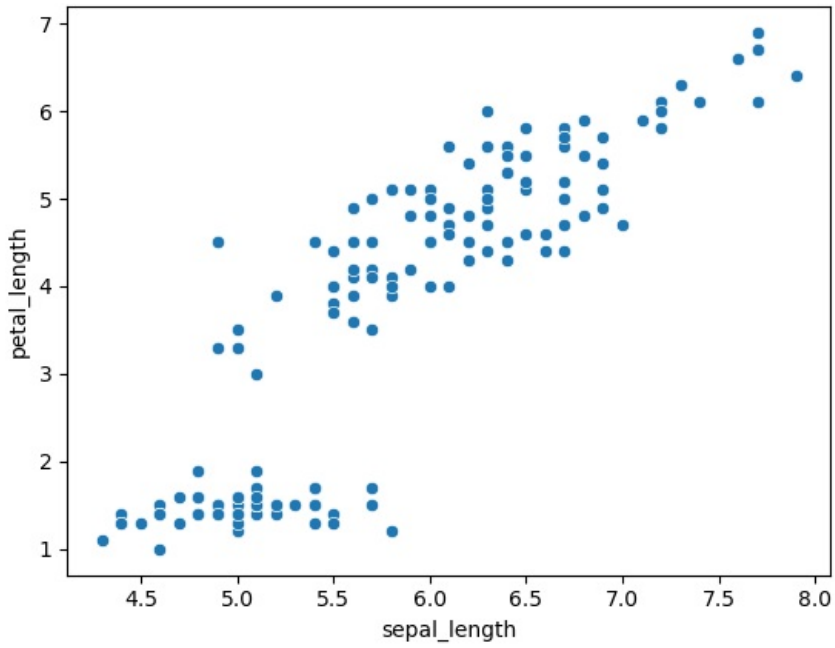
In [ ]:
```
sns.scatterplot(x = iris.sepal_length , y = iris.sepal_width)
```

Out[ ]:
```
<AxesSubplot: xlabel='sepal_length', ylabel='sepal_width'>
```



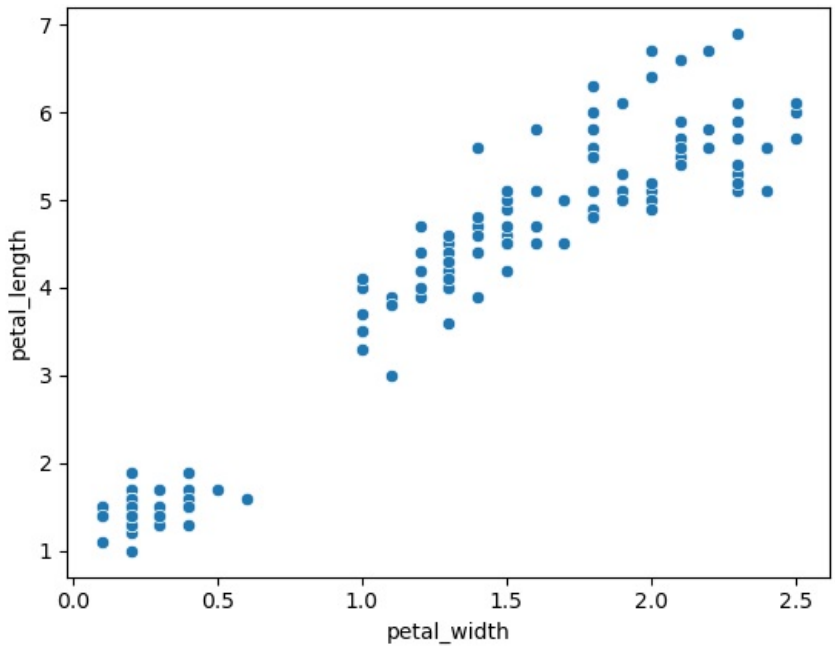
In [ ]:
```
sns.scatterplot(x = iris.sepal_length , y = iris.petal_length)
```

Out[ ]:
```
<AxesSubplot: xlabel='sepal_length', ylabel='petal_length'>
```



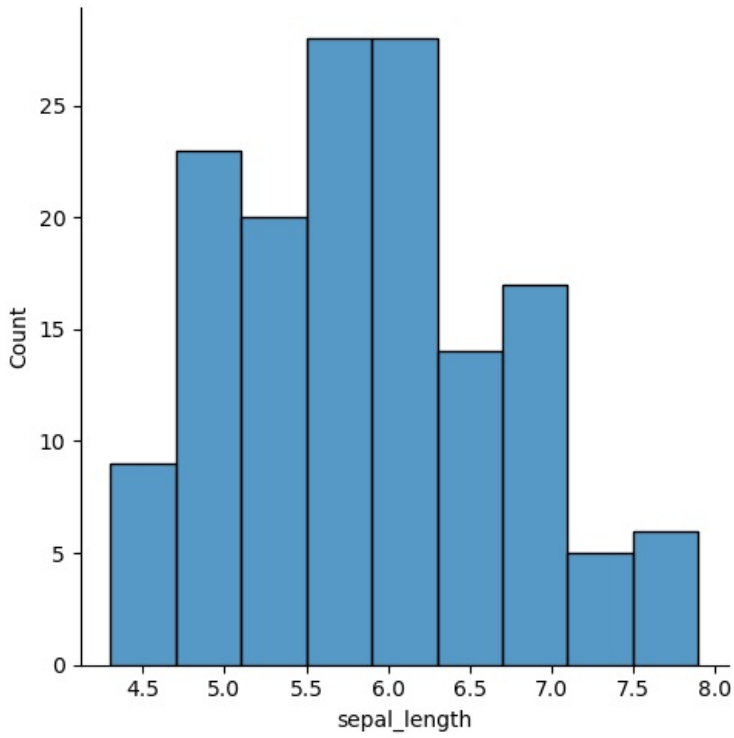
In [ ]:
```
sns.scatterplot(x = iris.petal_width , y = iris.petal_length)
```

Out[ ]:
```
<AxesSubplot: xlabel='petal_width', ylabel='petal_length'>
```



In [ ]:
```
sns.displot(iris['sepal_length'])
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7f681934ce80>
```



In [ ]:
```
tips = sns.load_dataset("tips")
```

In [ ]:
```
tips
```

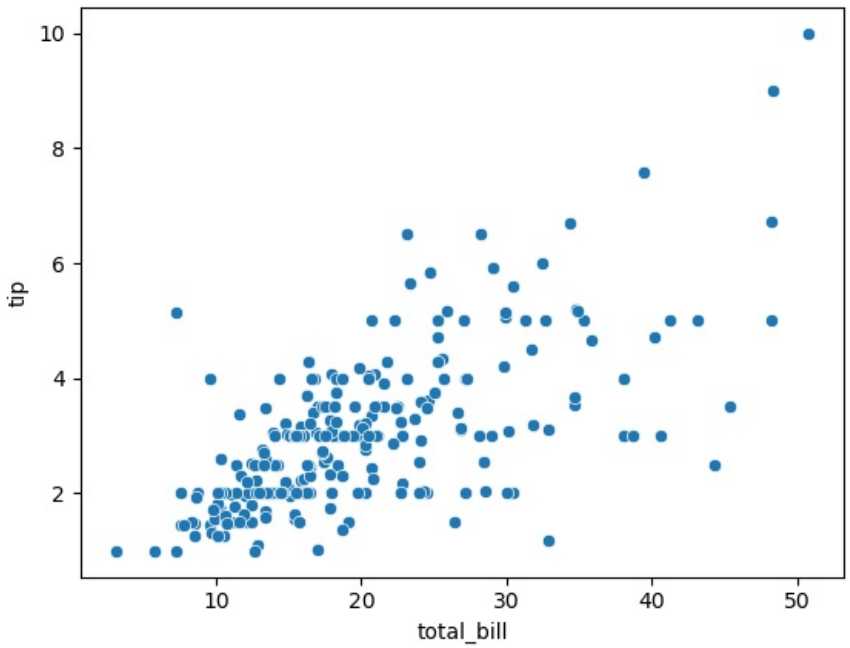
Out[ ]:

	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
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

In [ ]:
```
sns.scatterplot(x =tips.total_bill , y = tips.tip )
```

Out[ ]:
```
<AxesSubplot: xlabel='total_bill', ylabel='tip'>
```



In [ ]:
```
tips.head()
```

Out[ ]:

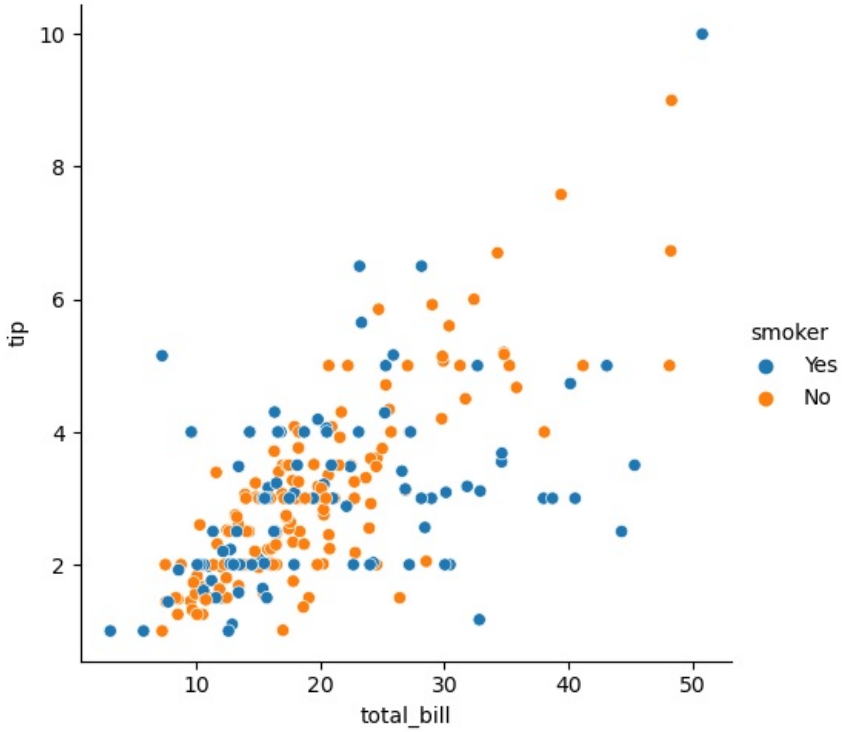
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

In [ ]:
```
tips['smoker'].value_counts()
```

Out[ ]:
```
No     151
Yes     93
Name: smoker, dtype: int64
```

In [ ]:
```
sns.relplot(x =tips.total_bill , y = tips.tip , data = tips , hue = 'smoker' )
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7f68180e1ed0>
```



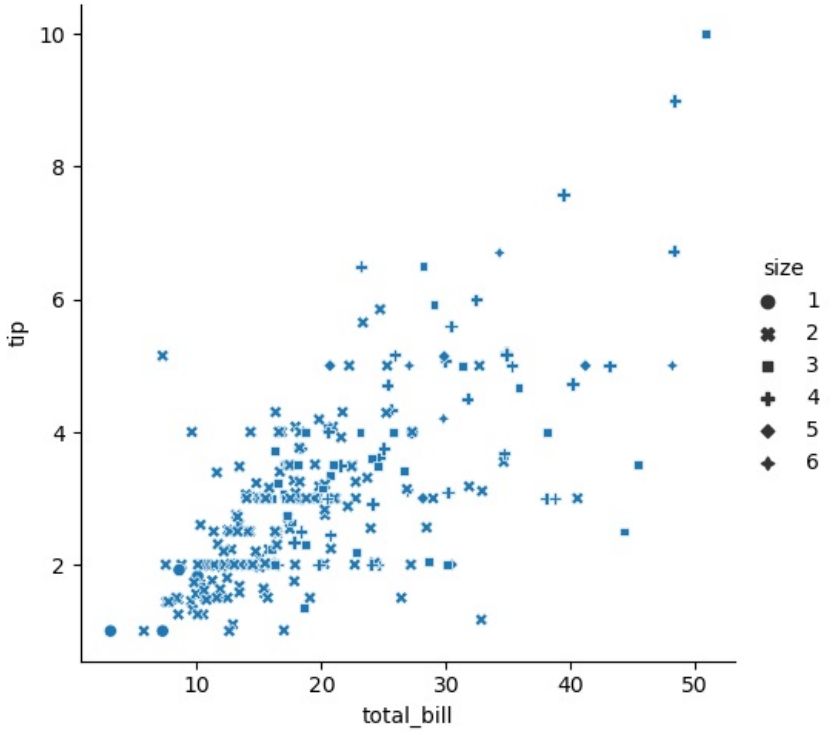
In [ ]:
```
tips.head()
```

Out[ ]:

	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

In [ ]:
```
sns.relplot(x =tips.total_bill , y = tips.tip , data = tips , style = 'size' )
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7f6817d8a470>
```

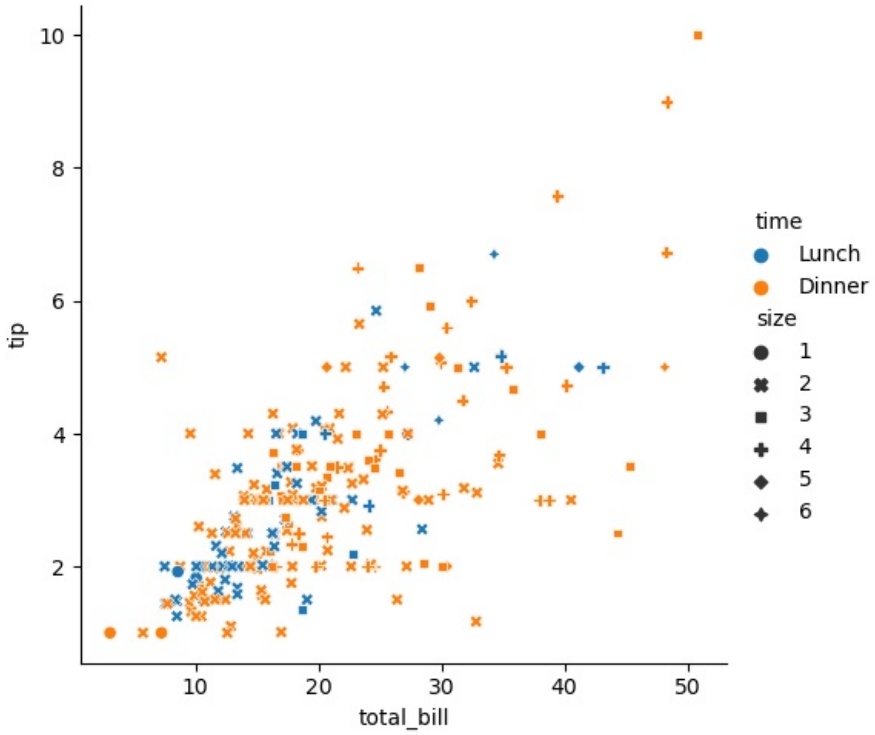


In [ ]:
```
tips['size'].value_counts()
```

Out[ ]:
```
2    156
3     38
4     37
5      5
1      4
6      4
Name: size, dtype: int64
```

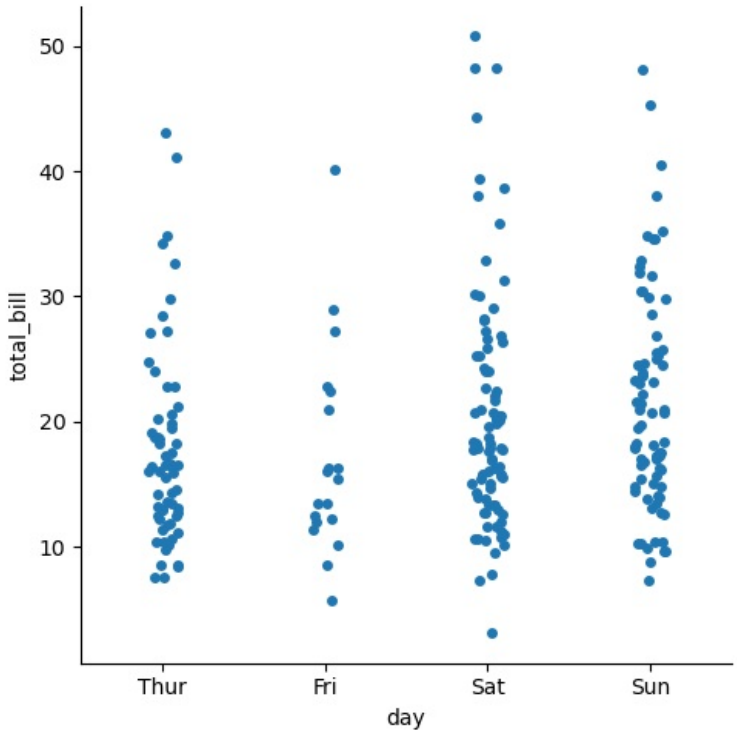
In [ ]:
```
sns.relplot(x =tips.total_bill , y = tips.tip , data = tips , style = 'size',hue = "time" )
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7f6817d89b40>
```



In [ ]:
```
sns.catplot(x = 'day' , y = 'total_bill' , data  = tips)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7f68179b4850>
```



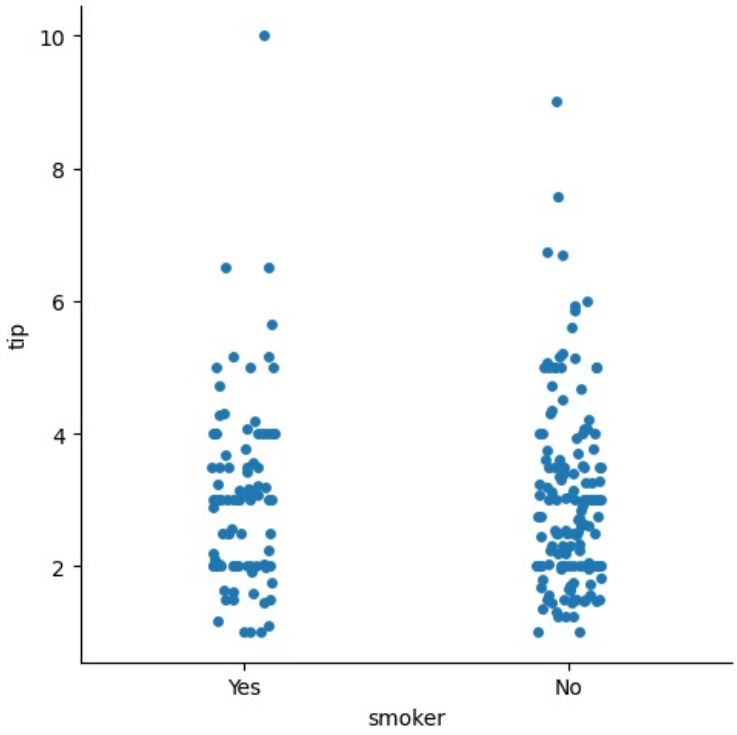
In [ ]:
```
tips.head()
```

Out[ ]:

	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

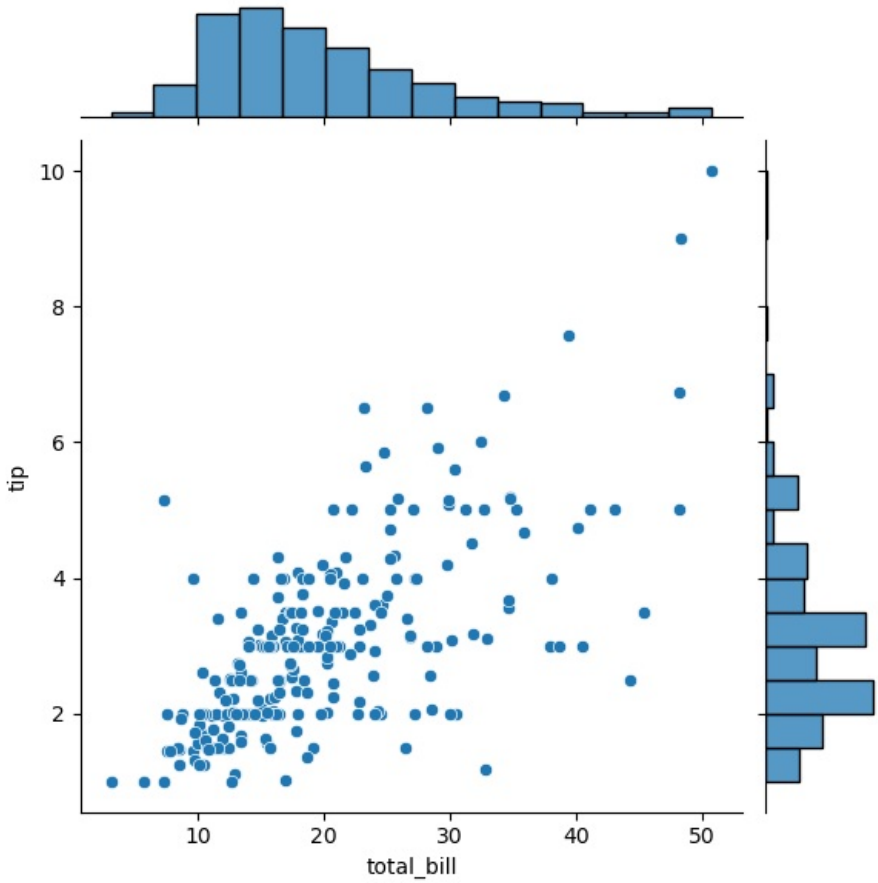
In [ ]:
```
sns.catplot(x = 'smoker' , y = 'tip' , data  = tips)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x7f68179f4c10>
```



In [ ]:
```
sns.jointplot(x = tips.total_bill , y = tips.tip)
```

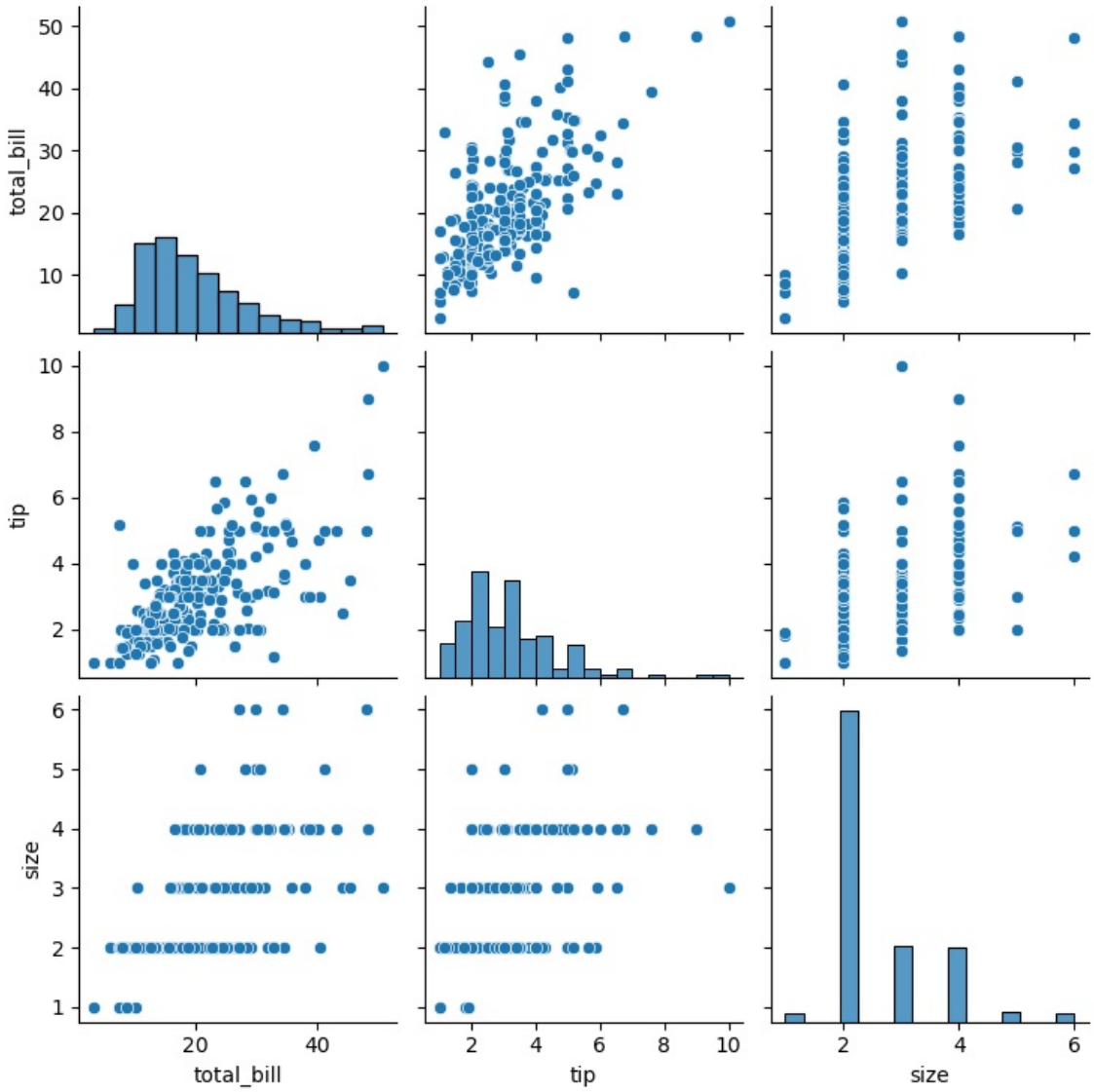
Out[ ]:
```
<seaborn.axisgrid.JointGrid at 0x7f6817dace50>
```



In [ ]:
```
sns.pairplot(tips)
```

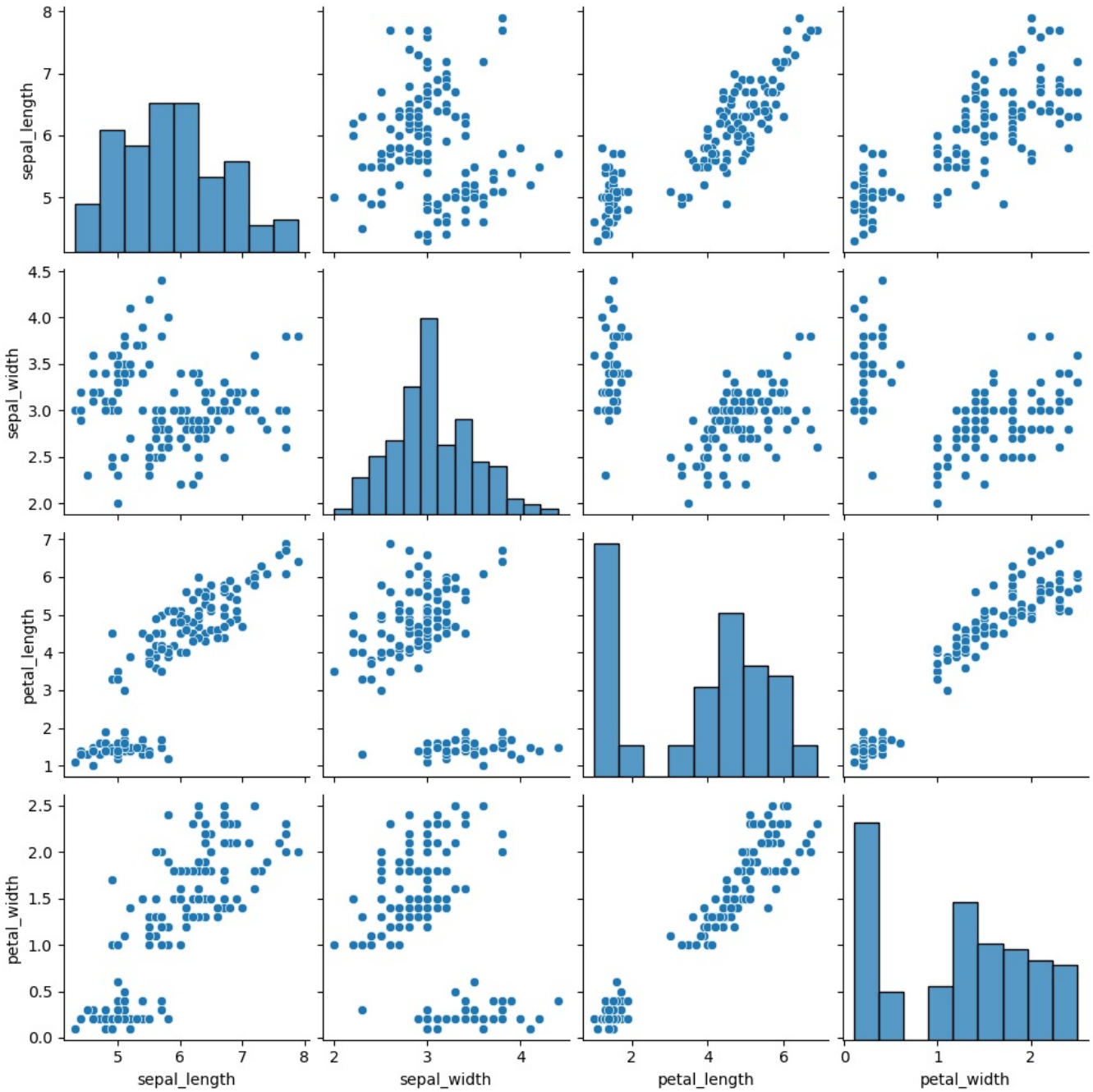
In [ ]:
```
sns.pairplot(tips)
```

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x7f68188a6d10>
```



In [ ]:
```
sns.pairplot(iris)
```

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x7f681833d300>
```



Advance guide for both Visualization Libraries

Basic plot with Matplotlib

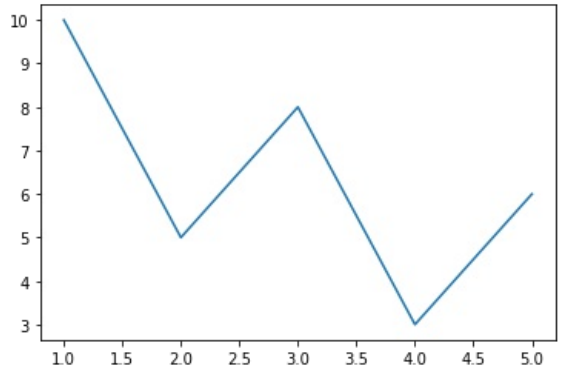
Matplotlib provides a wide range of options for creating different types of visualizations. To create a basic plot using Matplotlib, you can use the following code:

In [ ]:
```
import matplotlib.pyplot as plt

# Create some data
x = [1, 2, 3, 4, 5]
y = [10, 5, 8, 3, 6]

# Create a plot
plt.plot(x, y)

# Show the plot
plt.show()
```



This code will create a simple line plot with the given data.

Basic plot with Seaborn

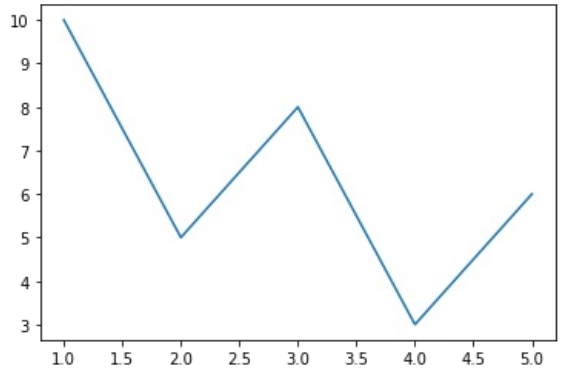
Seaborn is built on top of Matplotlib and provides a higher-level interface for creating beautiful statistical graphics. To create a basic plot using Seaborn, you can use the following code:

In [ ]:
```
import seaborn as sns

# Create some data
x = [1, 2, 3, 4, 5]
y = [10, 5, 8, 3, 6]

# Create a plot
sns.lineplot(x=x, y=y)

# Show the plot
plt.show()
```



This code will create a simple line plot with the given data using Seaborn.

Customizing plots

Both Matplotlib and Seaborn provide a wide range of options for customizing your plots. You can change the colors, add labels and titles, change the plot style, and much more.

Changing colors

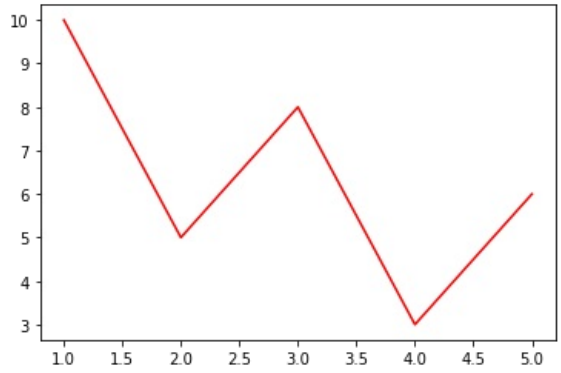
To change the colors of your plot, you can use the color parameter in Matplotlib or the palette parameter in Seaborn. **Here's an example:**

In [ ]:
```
import seaborn as sns

# Create some data
x = [1, 2, 3, 4, 5]
y = [10, 5, 8, 3, 6]

# Create a plot with a different color
sns.lineplot(x=x, y=y, color='red')

# Show the plot
plt.show()
```



Adding labels and titles

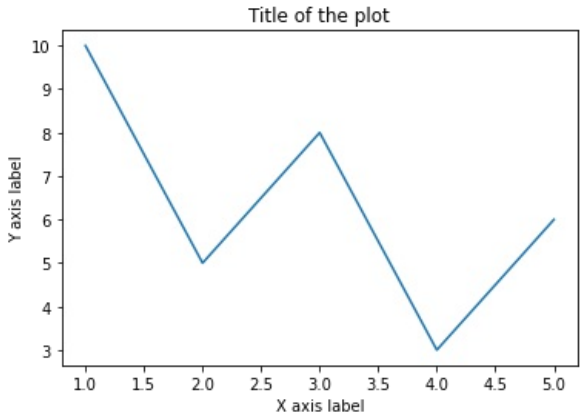
To add labels and titles to your plot, you can use the xlabel, ylabel, and title functions in Matplotlib, or the x_label, y_label, and title parameters in Seaborn. **Here's an example:**

In [ ]:
```
import seaborn as sns

# Create some data
x = [1, 2, 3, 4, 5]
y = [10, 5, 8, 3, 6]

# Create a plot with labels and a title
sns.lineplot(x=x, y=y)
plt.xlabel('X axis label')
plt.ylabel('Y axis label')
plt.title('Title of the plot')

# Show the plot
plt.show()
```



Changing plot style

Matplotlib provides a wide range of styles for your plot. You can change the style of your plot using the style.use() function. Seaborn also provides a number of built-in styles that can be used to customize your plot. **Here's an example of how to change the style of your plot:**

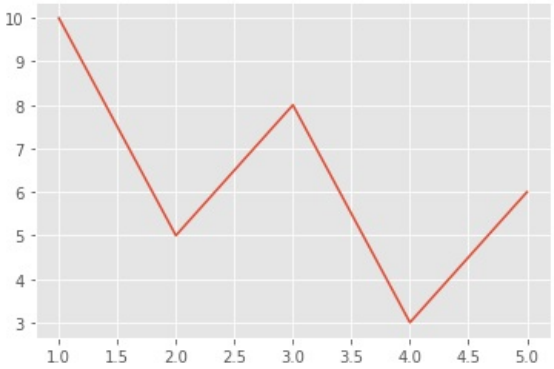
In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Set the style for Matplotlib
plt.style.use('ggplot')

# Create some data
x = [1, 2, 3, 4, 5]
y = [10, 5, 8, 3, 6]

# Create a plot with the changed style
sns.lineplot(x=x, y=y)

# Show the plot
plt.show()
```



Different types of plots

Both Matplotlib and Seaborn provide a wide range of options for creating different types of visualizations. Let's take a look at some of the most common types of plots.

Scatter plot

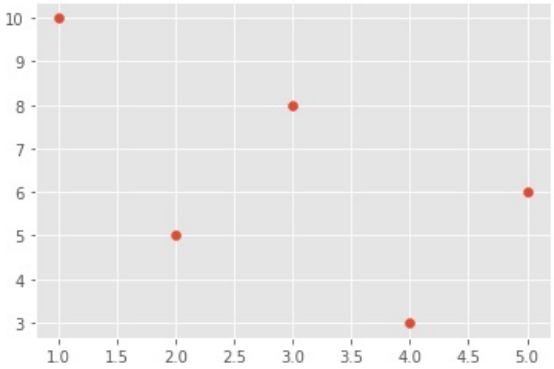
A scatter plot is used to display the relationship between two variables. Here's an example of how to create a scatter plot using Matplotlib:

In [ ]:
```
import matplotlib.pyplot as plt

# Create some data
x = [1, 2, 3, 4, 5]
y = [10, 5, 8, 3, 6]

# Create a scatter plot
plt.scatter(x, y)

# Show the plot
plt.show()
```



Bar plot

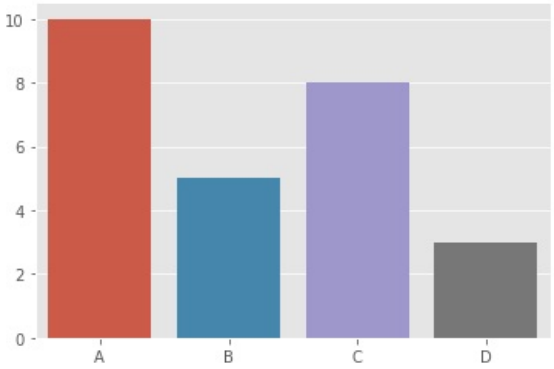
A bar plot is used to display the distribution of a categorical variable. Here's an example of how to create a bar plot using Seaborn:

In [ ]:
```
import seaborn as sns

# Create some data
x = ['A', 'B', 'C', 'D']
y = [10, 5, 8, 3]

# Create a bar plot
sns.barplot(x=x, y=y)

# Show the plot
plt.show()
```



Histogram

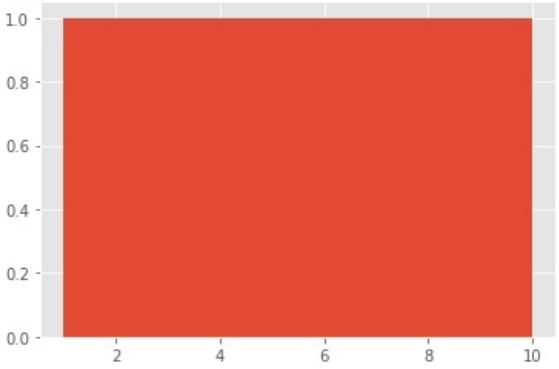
A histogram is used to display the distribution of a numerical variable. Here's an example of how to create a histogram using Matplotlib:

In [ ]:
```
import matplotlib.pyplot as plt

# Create some data
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Create a histogram
plt.hist(x)

# Show the plot
plt.show()
```



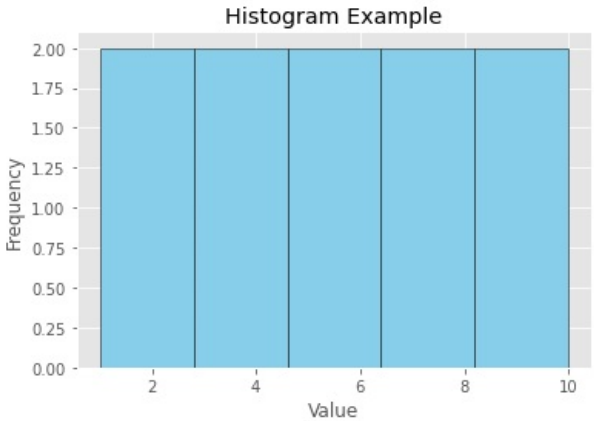
In [ ]:
```
import matplotlib.pyplot as plt

# Create some data
x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Create a histogram
plt.hist(x, bins=5, color='skyblue', edgecolor='black')

# Add axis labels and a title
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.title('Histogram Example')

# Show the plot
plt.show()
```



Heatmap

A heatmap is a graphical representation of data that uses a system of color-coding to represent different values. Heatmaps are used in

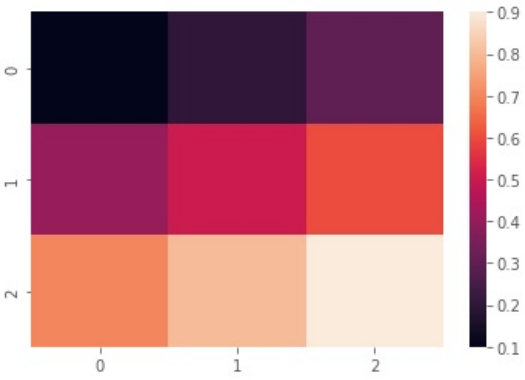
various forms of analytics but are most commonly used to show user behavior on specific webpages or webpage templates.

In [ ]:
```
import seaborn as sns

# Create some data
data = [[0.1, 0.2, 0.3],
        [0.4, 0.5, 0.6],
        [0.7, 0.8, 0.9]]

# Create a heatmap
sns.heatmap(data)

# Show the plot
plt.show()
```



Boxplot

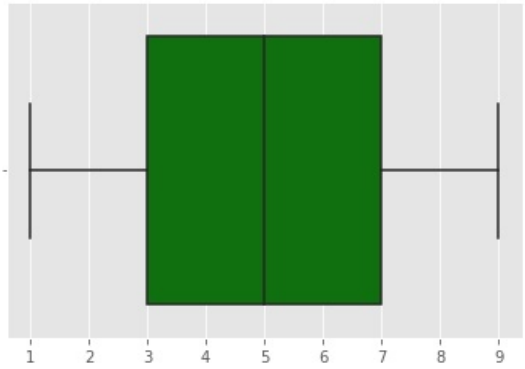
In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Create some data
data = [1, 2, 3, 4, 5, 6, 7, 8, 9]

# Create a box plot
sns.boxplot(data, color='green')

# Show the plot
plt.show()
```

```
D:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keywor
d arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Voilin Plot

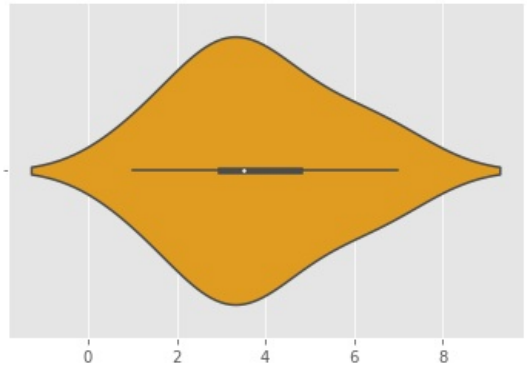
In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Create some data
data = [1, 2, 3, 3, 3, 4, 4, 5, 6, 7]

# Create a violin plot
sns.violinplot(data, color='orange')

# Show the plot
plt.show()
```

```
D:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keywor
d arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments wit
hout an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Conclusion

Matplotlib and Seaborn are powerful libraries for creating beautiful visualizations in Python. In this guide, we covered the basics of both libraries, including installation, importing, creating basic plots, customizing plots, and different types of plots. With this knowledge, you can now create beautiful visualizations to explore and communicate your data effectively.