BASIC VISUALIZATIONS

AGENDA

- Introduction to Data Visualizations
- Matplotlib vs Seaborn
- Matplotlib plots attributes
- Line charts
- Bar Graphs
- Histograms
- Scatter plots
- Heatmaps

INTRODUCTION TO DATA VISUALIZATIONS

- While working with data, it can be difficult to truly understand your data when it’s just in tabular form. To understand what exactly our data conveys, and to better clean it and select suitable models for it, we need to visualize it or represent it in pictorial form.

MATPLOTLIB VS SEABORN

Matplotlib	Seaborn
It is used for basic graph plotting like line charts, bar graphs, etc.	It is mainly used for statistics visualization and can perform complex visualizations with fewer commands.
It mainly works with datasets and arrays.	It works with entire datasets.
Seaborn is considerably more organized and functional than Matplotlib and treats the entire dataset as a solitary unit.	Matplotlib acts productively with data arrays and frames. It regards the aces and figures as objects.
Seaborn has more inbuilt themes and is mainly used for statistical analysis.	Matplotlib is more customizable and pairs well with Pandas and Numpy for Exploratory Data Analysis.

MATPLOTLIB PLOTS ATTRIBUTES

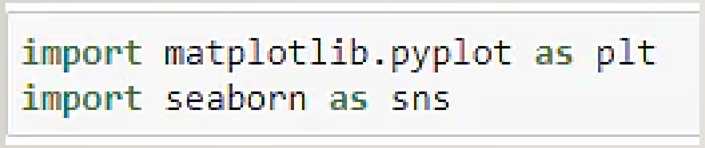
Method	Description
plot()	it creates the plot at the background of computer, it doesn’t displays it. We can also add a label as it’s argument that by what name we will call this plot – utilized in legend()
show()	it displays the created plots
xlabel()	it labels the x-axis
ylabel()	it labels the y-axis
title()	it gives the title to the graph
gca()	it helps to get access over the all the four axes of the graph
gca().spines[‘right/left/top/bottom’].set_visible(True/False)	it access the individual spines or the individual boundaries and helps to change theoir visibility

MATPLOTLIB PLOTS ATTRIBUTES

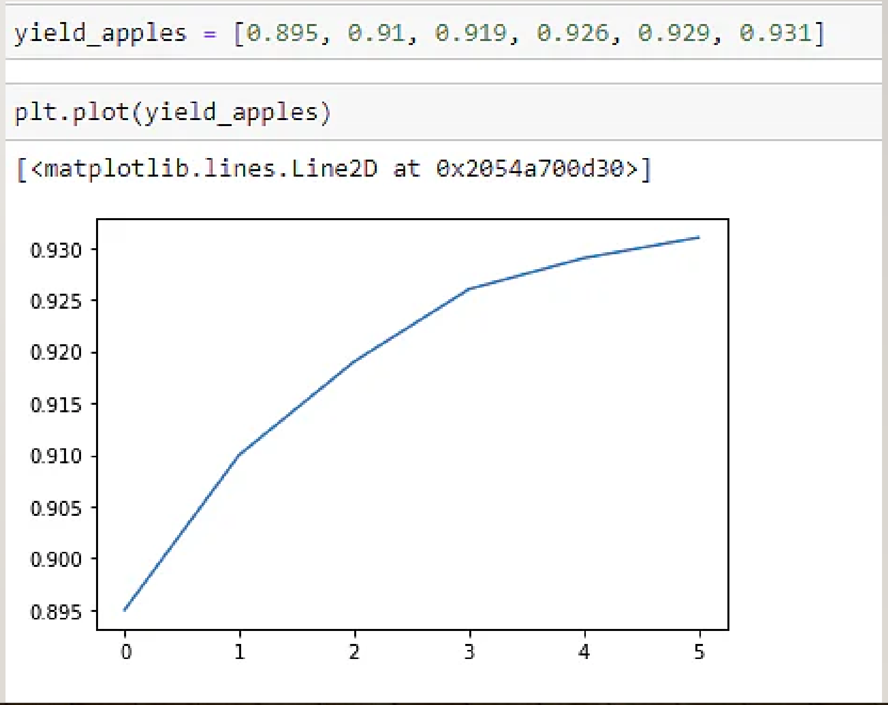
xticks()	it decides how the markings are to be made on the x-axis
yticks()	it decides how the markings are to be made on the y-axis
gca().legend()	pass a list as it’s arguments of all the plots made, if labels are not explicitly specified then add the values in the list in the same order as the plots are made
annotate()	it is use to write comments on the graph at the specified position
figure(figsize = (x, y))	whenever we want the result to be displayed in a separate window we use this command, and figsize argument decides what will be the initial size of the window that will be displayed after the run
subplot(r, c, i)	it is used to create multiple plots in the same figure with r signifies the no of rows in the figure, c signifies no of columns in a figure and i specifies the positioning of the particular plot
set_xticks	it is used to set the range and the step size of the markings on x – axis in a subplot
set_yticks	it is used to set the range and the step size of the markings on y – axis in a subplot

LINE CHARTS

- A Line chart is a graph that represents information as a series of data points connected by a straight line. In line charts, each data point or marker is plotted and connected with a line or curve.



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



```
yield_apples = [0.895, 0.91, 0.919, 0.926, 0.929, 0.931]
```

```
plt.plot(yield_apples)
```

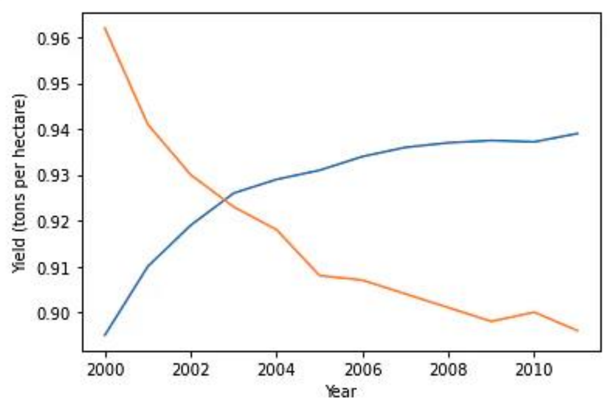
```
[<matplotlib.lines.Line2D at 0x2054a700d30>]
```

LINE CHARTS

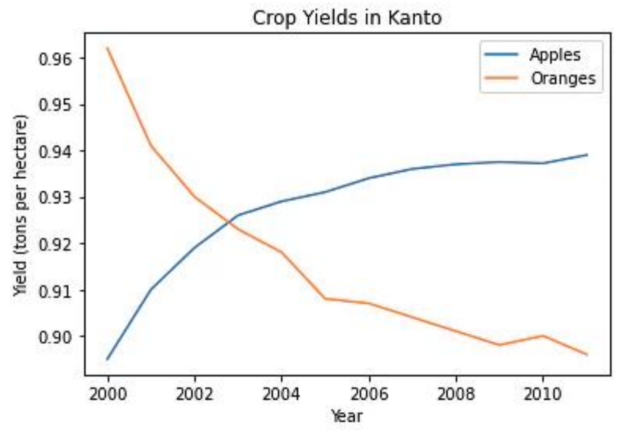
- To plot multiple datasets on the same graph, just use the plt.plot function once for each dataset. Let's use this to compare the yields of apples vs. oranges on the same graph.

```
years = range(2000, 2012)
apples = [0.895, 0.91, 0.919, 0.926, 0.929, 0.931, 0.934, 0.936, 0.937, 0.9375, 0.9372, 0.939]
oranges = [0.962, 0.941, 0.930, 0.923, 0.918, 0.908, 0.907, 0.904, 0.901, 0.898, 0.9, 0.896, ]
```

```
plt.plot(years, apples)
plt.plot(years, oranges)
plt.xlabel('Year')
plt.ylabel('Yield (tons per hectare)');
```

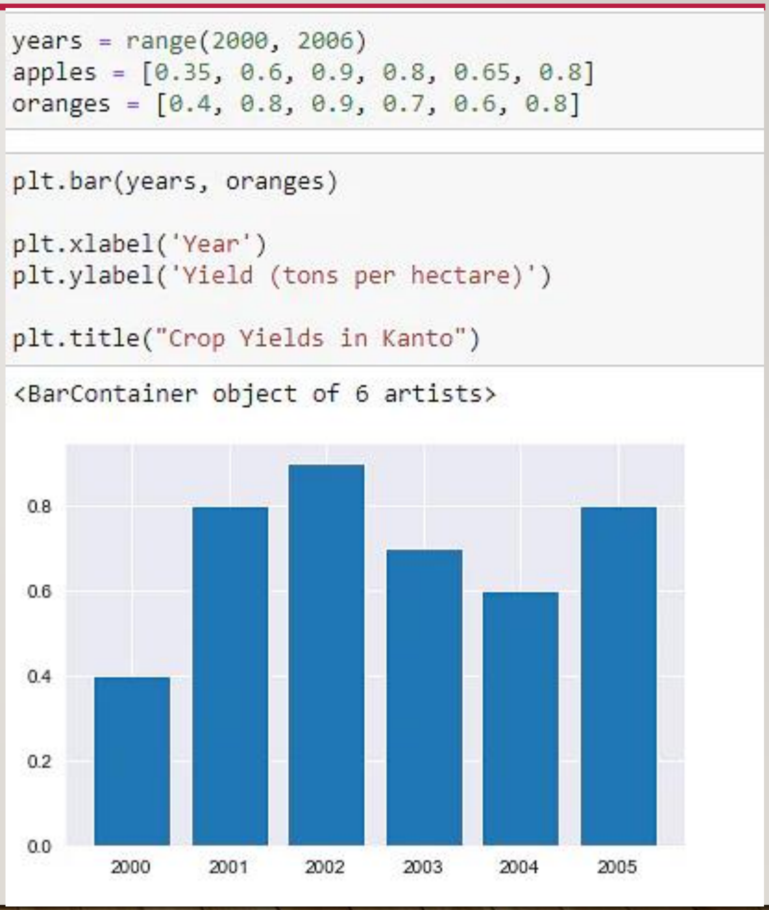


```
plt.plot(years, apples)
plt.plot(years, oranges)

plt.xlabel('Year')
plt.ylabel('Yield (tons per hectare)')

plt.title("Crop Yields in Kanto")
plt.legend(['Apples', 'Oranges']);
```



BAR GRAPHS

- When you have categorical data, you can represent it with a bar graph. A bar graph plots data with the help of bars, which represent value on the y-axis and category on the x-axis. Bar graphs use bars with varying heights to show the data which belongs to a specific category.



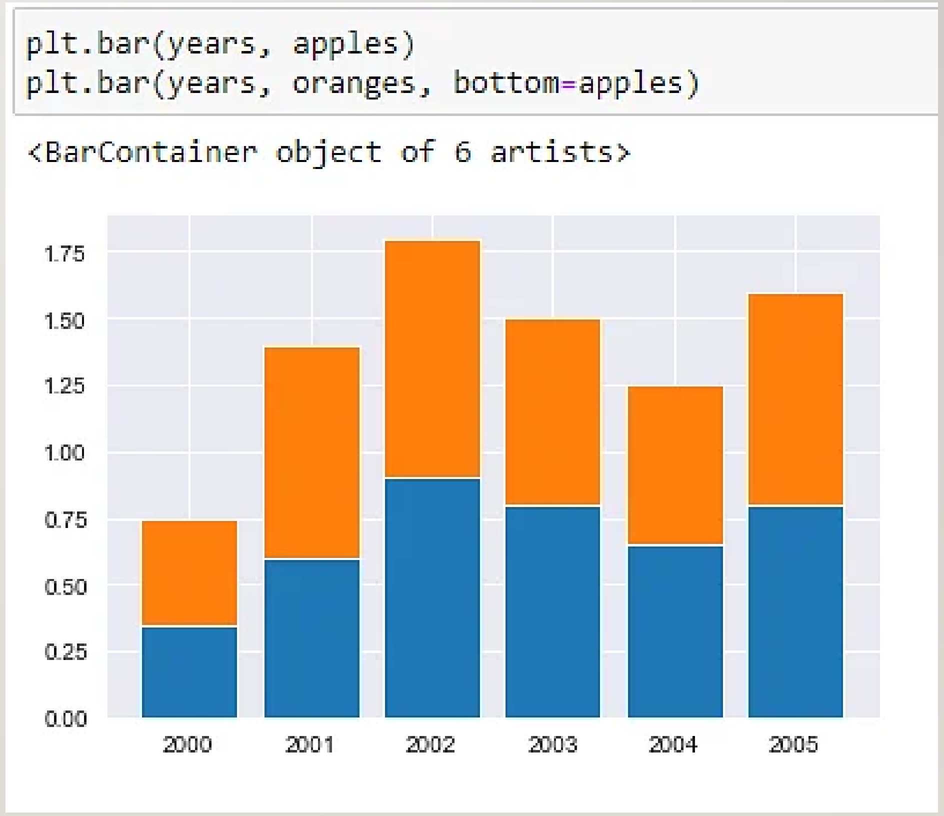
```
years = range(2000, 2006)
apples = [0.35, 0.6, 0.9, 0.8, 0.65, 0.8]
oranges = [0.4, 0.8, 0.9, 0.7, 0.6, 0.8]
```

```
plt.bar(years, oranges)

plt.xlabel('Year')
plt.ylabel('Yield (tons per hectare)')

plt.title("Crop Yields in Kanto")
```

```
<BarContainer object of 6 artists>
```

BAR GRAPHS

- We can also stack bars on top of each other. Let's plot the data for apples and oranges.

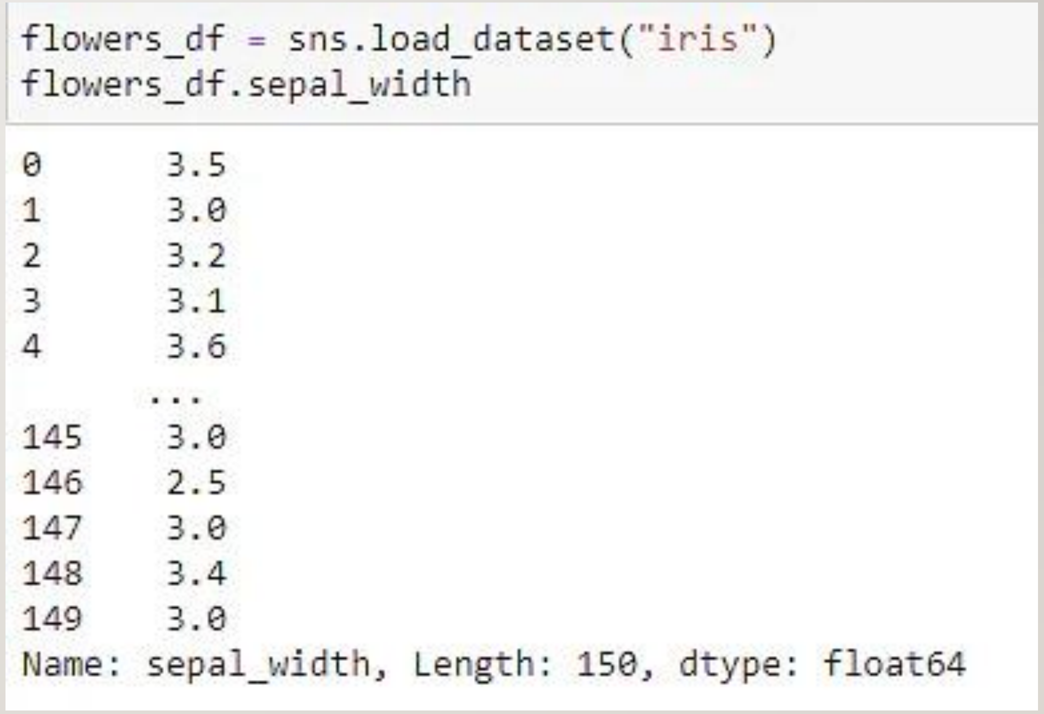


```
plt.bar(years, apples)
plt.bar(years, oranges, bottom=apples)
```

```
<BarContainer object of 6 artists>
```

HISTOGRAMS

- A Histogram is a bar representation of data that varies over a range. It plots the height of the data belonging to a range along the y-axis and the range along the x-axis. Histograms are used to plot data over a range of values. They use a bar representation to show the data belonging to each range. Let's again use the 'Iris' data which contains information about flowers to plot histograms.

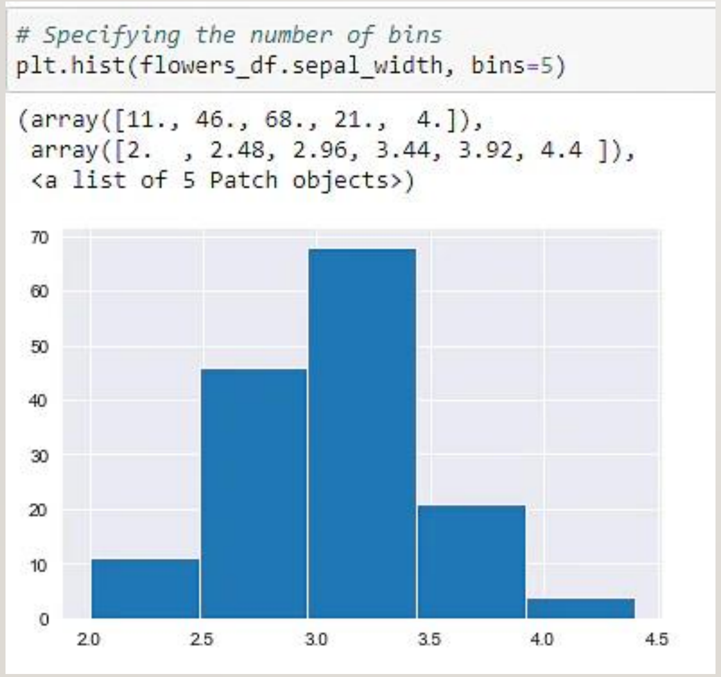


```
flowers_df = sns.load_dataset("iris")
flowers_df.sepal_width
```

```
0      3.5
1      3.0
2      3.2
3      3.1
4      3.6
      ...
145    3.0
146    2.5
147    3.0
148    3.4
149    3.0
Name: sepal_width, Length: 150, dtype: float64
```

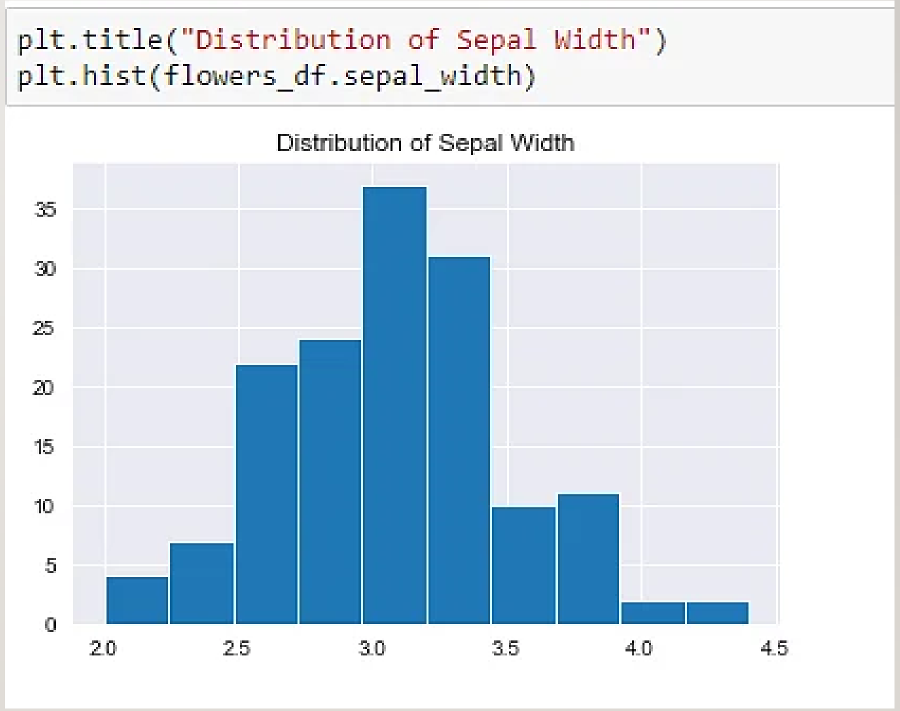
HISTOGRAMS

- Now, let's plot a histogram using the hist() function.



```
# Specifying the number of bins
plt.hist(flowers_df.sepal_width, bins=5)
```

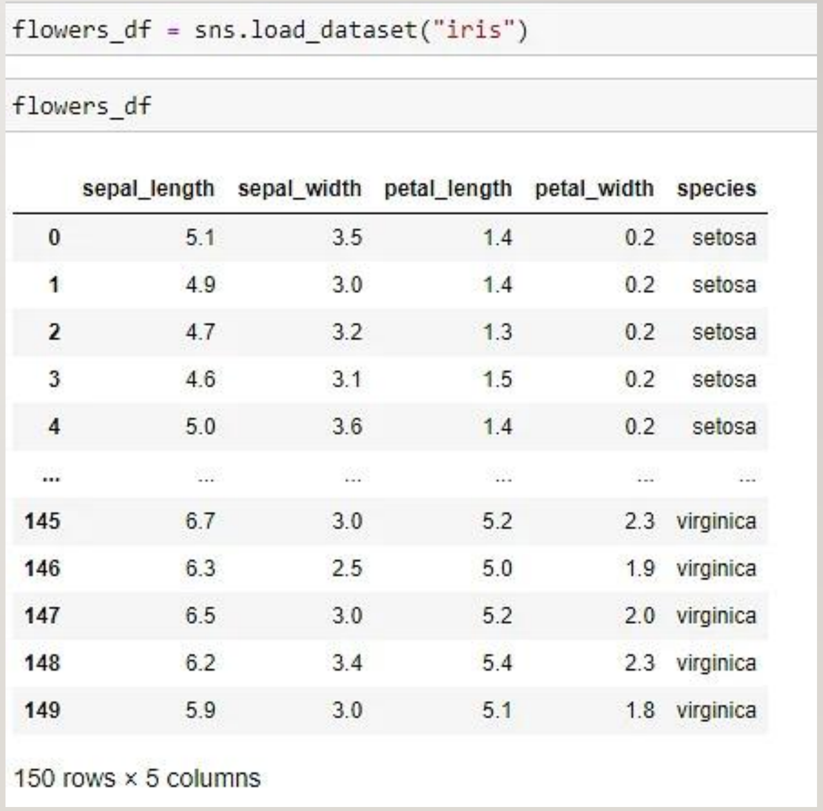
```
(array([11., 46., 68., 21.,  4.]),
 array([2.  , 2.48, 2.96, 3.44, 3.92, 4.4 ]),
 <a list of 5 Patch objects>)
```



```
plt.title("Distribution of Sepal Width")
plt.hist(flowers_df.sepal_width)
```

SCATTER PLOTS

- Scatter plots are used when we have to plot two or more variables present at different coordinates. The data is scattered all over the graph and is not confined to a range. Two or more variables are plotted in a Scatter Plot, with each variable being represented by a different color.



```
flowers_df = sns.load_dataset("iris")
```

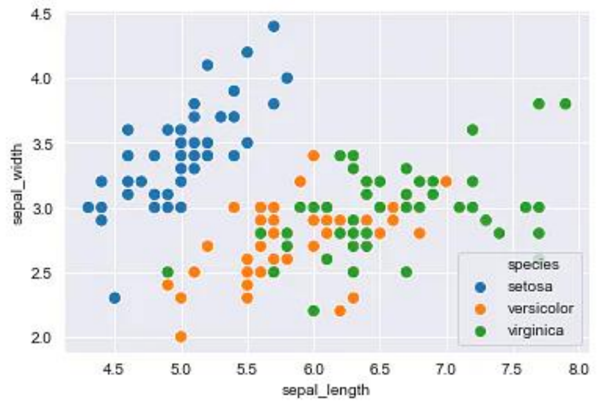
```
flowers_df
```

	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

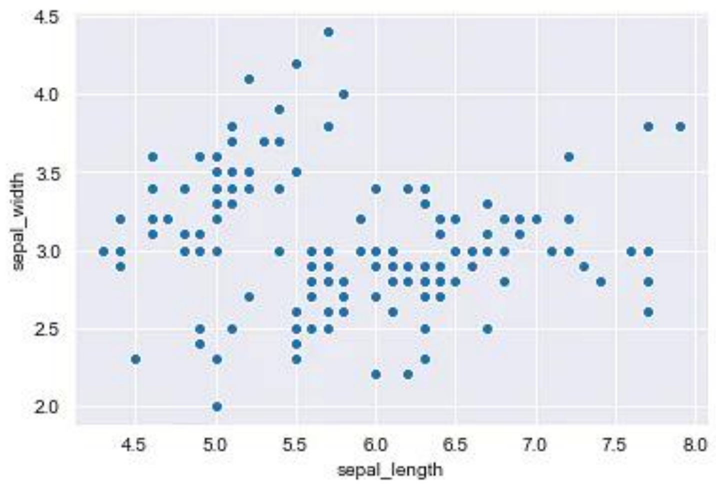
SCATTER PLOTS

```
sns.scatterplot(x=flowers_df.sepal_length, y=flowers_df.sepal_width, hue=flowers_df.species, s=70);
```



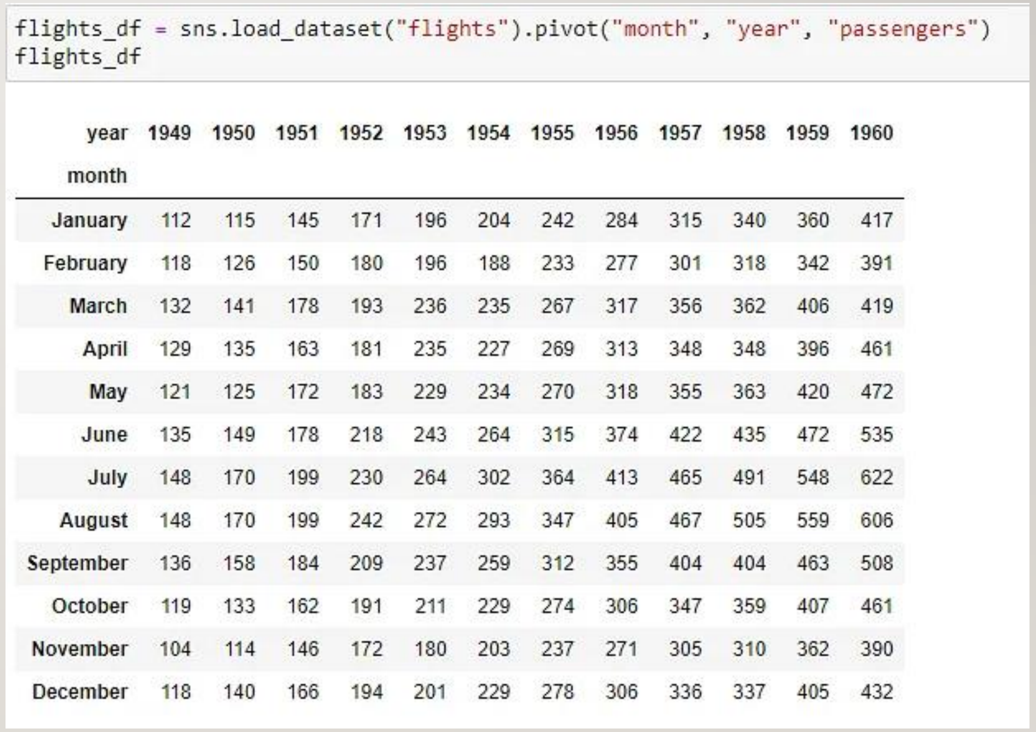
```
sns.scatterplot(x=flowers_df.sepal_length, y=flowers_df.sepal_width)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2054a958820>
```



HEAT MAPS

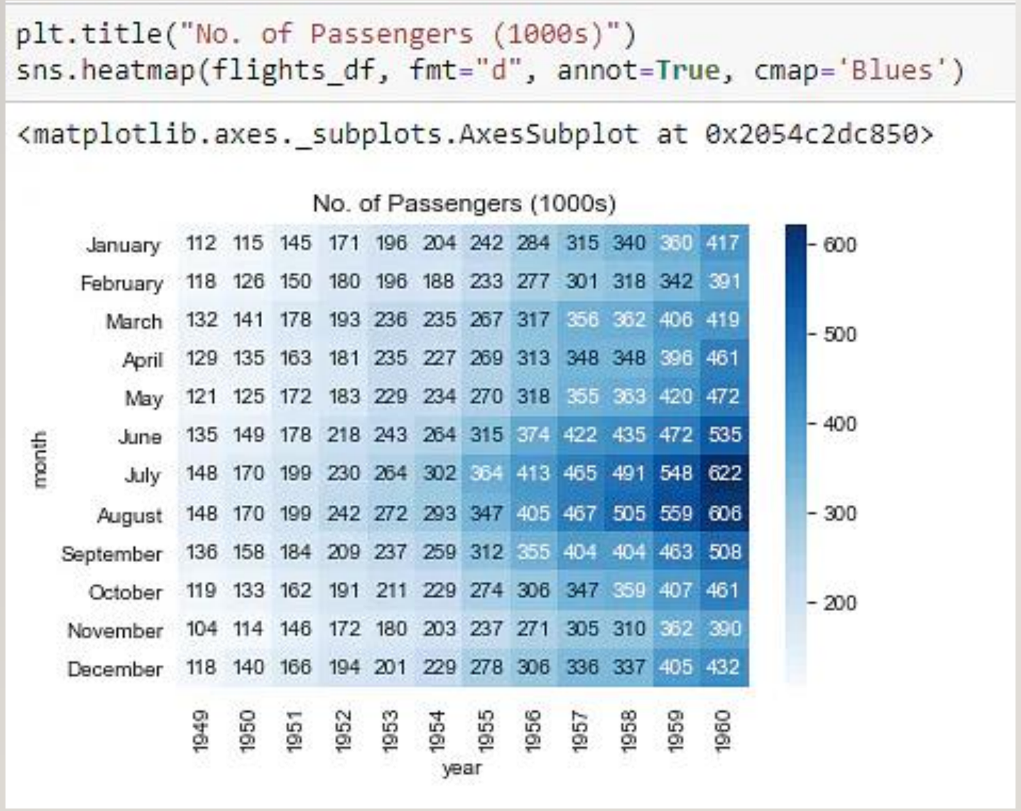
- Heatmaps are used to see changes in behavior or gradual changes in data. It uses different colors to represent different values. Based on how these colors range in hues, intensity, etc., tells us how the phenomenon varies. Let's use heatmaps to visualize monthly passenger footfall at an airport over 12 years from the flights dataset in Seaborn.



```
flights_df = sns.load_dataset("flights").pivot("month", "year", "passengers")
flights_df
```

year	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960
month												
January	112	115	145	171	196	204	242	284	315	340	360	417
February	118	126	150	180	196	188	233	277	301	318	342	391
March	132	141	178	193	236	235	267	317	356	362	406	419
April	129	135	163	181	235	227	269	313	348	348	396	461
May	121	125	172	183	229	234	270	318	355	363	420	472
June	135	149	178	218	243	264	315	374	422	435	472	535
July	148	170	199	230	264	302	364	413	465	491	548	622
August	148	170	199	242	272	293	347	405	467	505	559	606
September	136	158	184	209	237	259	312	355	404	404	463	508
October	119	133	162	191	211	229	274	306	347	359	407	461
November	104	114	146	172	180	203	237	271	305	310	362	390
December	118	140	166	194	201	229	278	306	336	337	405	432

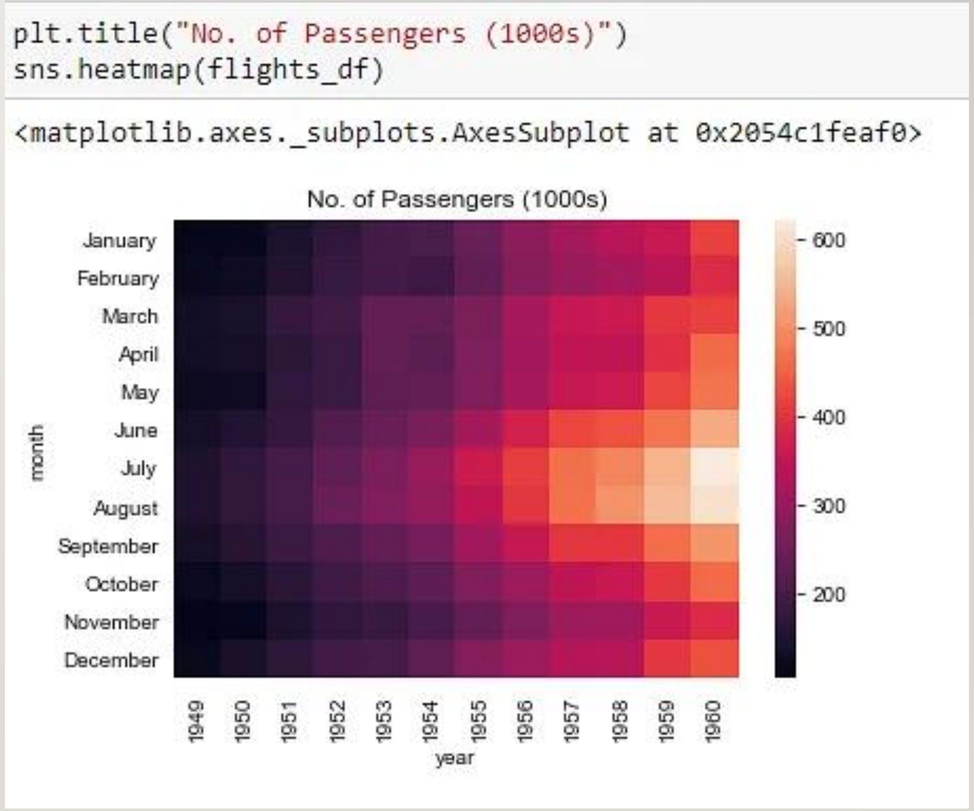
HEAT MAPS



```
plt.title("No. of Passengers (1000s)")
sns.heatmap(flights_df, fmt="d", annot=True, cmap='Blues')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2054c2dc850>
```

month	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960
January	112	115	145	171	196	204	242	284	315	340	360	417
February	118	126	150	180	196	188	233	277	301	318	342	391
March	132	141	178	193	236	235	267	317	356	362	406	419
April	129	135	163	181	235	227	269	313	348	348	396	461
May	121	125	172	183	229	234	270	318	355	363	420	472
June	135	149	178	218	243	264	315	374	422	435	472	535
July	148	170	199	230	264	302	364	413	465	491	548	622
August	148	170	199	242	272	293	347	405	467	505	559	606
September	136	158	184	209	237	259	312	355	404	404	463	508
October	119	133	162	191	211	229	274	306	347	359	407	461
November	104	114	146	172	180	203	237	271	305	310	362	390
December	118	140	166	194	201	229	278	306	336	337	405	432



```
plt.title("No. of Passengers (1000s)")
sns.heatmap(flights_df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2054c1feaf0>
```

CONCLUSION

In this lecture, The Complete Guide to Data Visualization in Python, we gave an overview of data visualization in python and discussed how to create Line Charts, Bar Graphs, Histograms, Scatter Plot, and Heat Maps using various data visualization packages offered by Python like Matplotlib and Seaborn.

Thank you