Data Visualization in Python.

Python provides various libraries that come with different features for visualizing data. All these libraries come with different features and can support various types of graphs. In this tutorial, we will be discussing four such libraries.

Matplotlib Seaborn Bokeh Plotly

In [2]:
```
import pandas as pd

#reading the database
data = pd.read_csv("tips.csv")

#Printing the top 10 rows
display(data.head(10))
```

	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
5	25.29	4.71	Male	No	Sun	Dinner	4
6	8.77	2.00	Male	No	Sun	Dinner	2
7	26.88	3.12	Male	No	Sun	Dinner	4
8	15.04	1.96	Male	No	Sun	Dinner	2
9	14.78	3.23	Male	No	Sun	Dinner	2

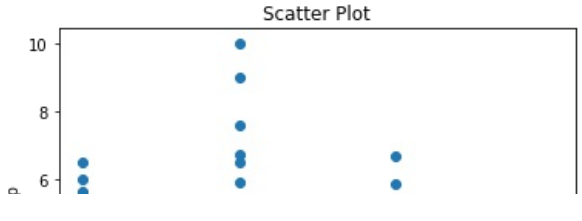
Matplotlib Library

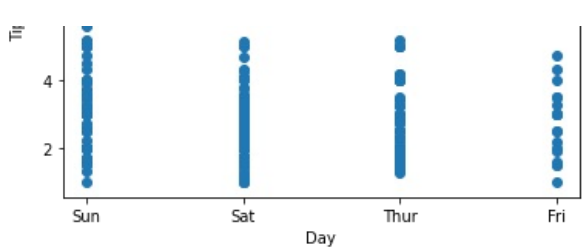
Matplotlib is an easy-to-use, low-level data visualization library that is built on NumPy arrays. It consists of various plots like scatter plot, line plot, histogram, etc. Matplotlib provides a lot of flexibility.

Scatter Plot

Scatter plots are used to observe relationships between variables and uses dots to represent the relationship between them. The scatter() method in the matplotlib library is used to draw a scatter plot.

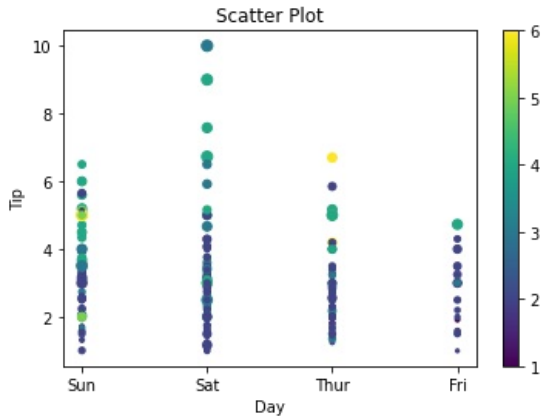
In [5]:
```
#Importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#Reading the Dataset
data = pd.read_csv("tips.csv")

#Scatter Plot with day against tip
plt.scatter(data['day'], data['tip'])

#Adding Title to the Plot
plt.title("Scatter Plot")

#Setting the Xlabel and YLabel
plt.xlabel('Day')
plt.ylabel('Tip')

#Showing the Scatter Plot
plt.show()
```





This graph can be more meaningful if we can add colors and also change the size of the points. We can do this by using the c and s parameter respectively of the scatter function. We can also show the color bar using the colorbar() method.

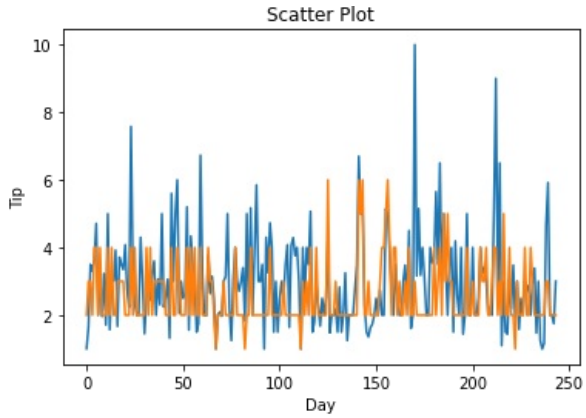
In [6]:
```
import pandas as pd
import matplotlib.pyplot as plt


# reading the database
data = pd.read_csv("tips.csv")

# Scatter plot with day against tip
plt.scatter(data['day'], data['tip'], c=data['size'],
                        s=data['total_bill'])

# Adding Title to the Plot
plt.title("Scatter Plot")

# Setting the X and Y labels
plt.xlabel('Day')
plt.ylabel('Tip')

plt.colorbar()

plt.show()
```



Line Chart

Line Chart is used to represent a relationship between two data X and Y on a different axis. It is plotted using the plot() function. Let's see the below example.

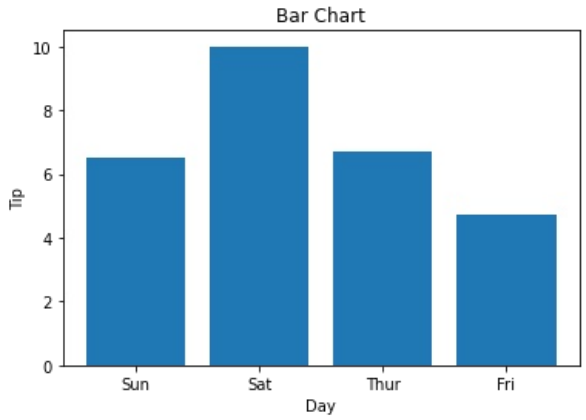
In [7]:
```
import pandas as pd
import matplotlib.pyplot as plt


# reading the database
data = pd.read_csv("tips.csv")

# Scatter plot with day against tip
plt.plot(data['tip'])
plt.plot(data['size'])

# Adding Title to the Plot
plt.title("Scatter Plot")

# Setting the X and Y labels
plt.xlabel('Day')
plt.ylabel('Tip')

plt.show()
```



Bar Plot

A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. It can be created using the bar() method.

In [8]:
```
import pandas as pd
import matplotlib.pyplot as plt


# reading the database
data = pd.read_csv("tips.csv")

# Bar chart with day against tip
plt.bar(data['day'], data['tip'])

plt.title("Bar Chart")

# Setting the X and Y labels
plt.xlabel('Day')
plt.ylabel('Tip')

# Adding the legends
plt.show()
```

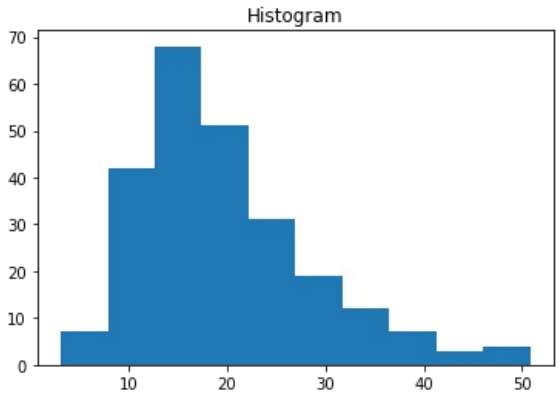


Histogram

A histogram is basically used to represent data in the form of some groups. It is a type of bar plot where the X-axis represents the bin ranges while the Y-axis gives information about frequency. The hist() function is used to compute and create a histogram. In histogram, if we pass categorical data then it will automatically compute the frequency of that data i.e. how often each value occurred.

In [11]:
```
#Importing Libraries
import pandas as pd
import matplotlib.pyplot as plt

#Reading the Dataset
data = pd.read_csv("tips.csv")

#Histogram of total Bills
plt.hist(data['total_bill'])
```

```
plt.title("Histogram")

#Adding the legends
plt.show()
```



Seaborn

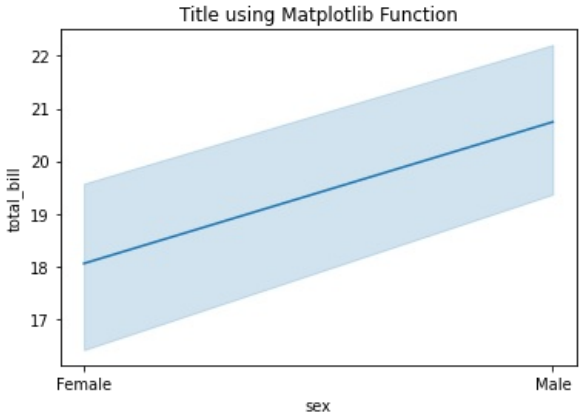
Seaborn is a high-level interface built on top of the Matplotlib. It provides beautiful design styles and color palettes to make more attractive graphs.

Seaborn is built on the top of Matplotlib, therefore it can be used with the Matplotlib as well. Using both Matplotlib and Seaborn together is a very simple process. We just have to invoke the Seaborn Plotting function as normal, and then we can use Matplotlib's customization function.

Lineplot using sns function.

In [12]:

```
#Importing Packages
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

#Reading the dataset
data = pd.read_csv("tips.csv")

#Draw Lineplot
sns.lineplot(x="sex", y="total_bill", data=data)

#Setting the title using Matplotlib Function
plt.title("Title using Matplotlib Function")

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

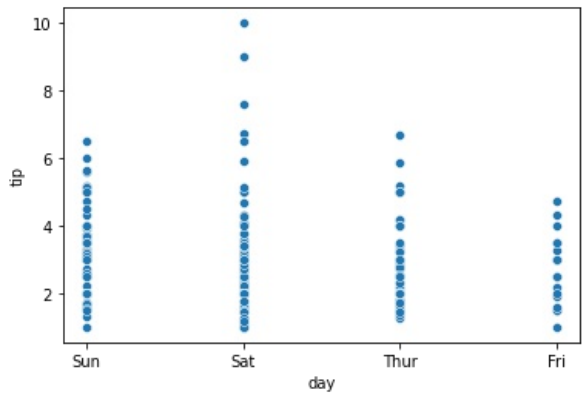


Scatter Plot

Scatter plot is plotted using the scatterplot() method. This is similar to Matplotlib, but additional argument data is required.

In [13]:

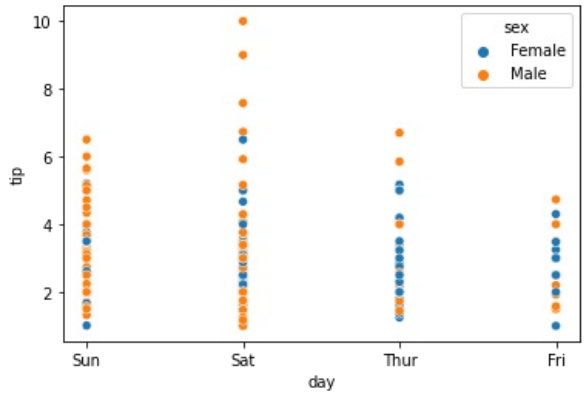
```
#Importing the Packages
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

#Loading the Dataset
data = pd.read_csv("tips.csv")

sns.scatterplot(x="day", y="tip", data=data)
plt.show()
```



You will find that while using Matplotlib it will a lot difficult if you want to color each point of this plot according to the sex. But in scatter plot it can be done with the help of hue argument.

In [16]:

```
#Importing packages
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

#Reading the dataset
data = pd.read_csv("tips.csv")

sns.scatterplot(x="day", y = "tip", data=data, hue = "sex")

plt.show()
```

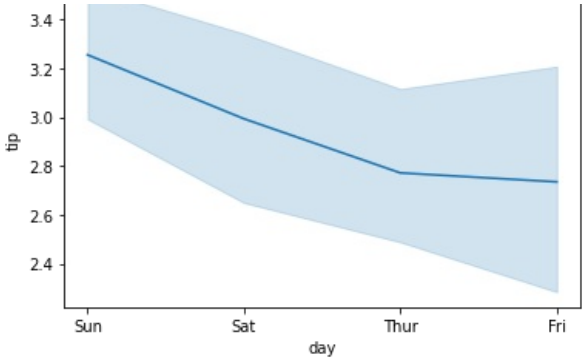


Line Plot

Line Plot in Seaborn plotted using the lineplot() method. In this, we can pass only the data argument also.

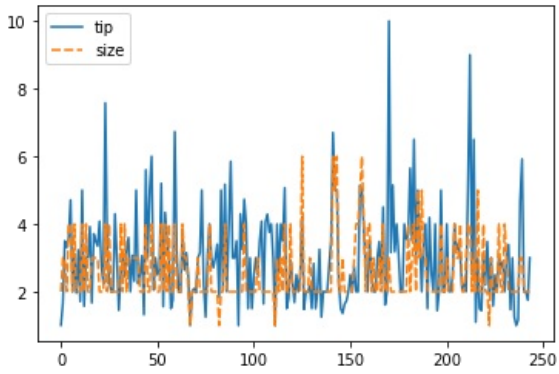
In [17]:

```
#Importing Packages

import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt

#reading the database
data = pd.read_csv("tips.csv")

#Lineplot
sns.lineplot(x='day', y='tip', data=data)
plt.show()
```



In [18]:

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd


# reading the database
data = pd.read_csv("tips.csv")

# using only data attribute
sns.lineplot(data=data.drop(['total_bill'], axis=1))
plt.show()
```

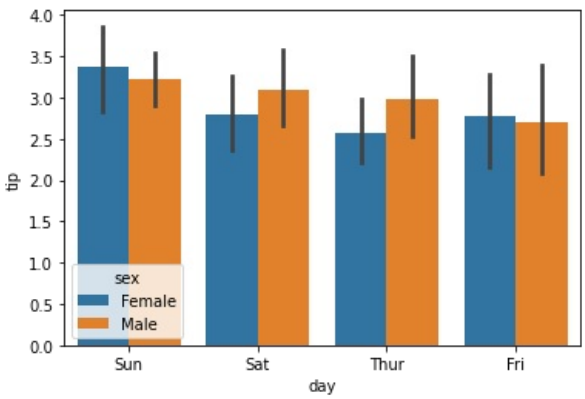


Bar Plot

Bar Plot in Seaborn can be created using the barplot() method.

In [19]:

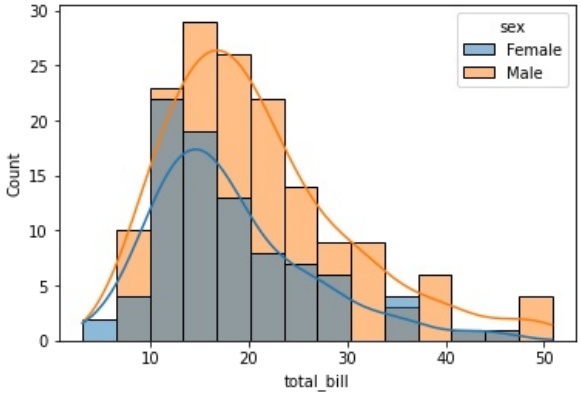
```
#Importing Packages
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

#Reading the database
data = pd.read_csv("tips.csv")

sns.barplot(x="day", y="tip", data=data, hue='sex')

plt.show()
```



Histogram

The histogram in Seaborn can be plotted using the histplot() function.

In [20]:
```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd


# reading the database
data = pd.read_csv("tips.csv")

sns.histplot(x='total_bill', data=data, kde=True, hue='sex')

plt.show()
```



After going through all these plots you must have noticed that customizing plots using Seaborn is a lot more easier than using Matplotlib. And it is also built over matplotlib then we can also use matplotlib functions while using Seaborn.

Bokeh

Let's move on to the third library of our list. Bokeh is mainly famous for its interactive charts visualization. Bokeh renders its plots using HTML and JavaScript that uses modern web browsers for presenting elegant, concise construction of novel graphics with high-level interactivity.

Scatter Plot

In [9]:
```
# importing the modules
from bokeh.plotting import figure, output_file, show
from bokeh.palettes import magma
import pandas as pd


# instantiating the figure object
graph = figure(title = "Bokeh Scatter Graph")

# reading the database
data = pd.read_csv("tips.csv")

color = magma(256)

# plotting the graph
graph.scatter(data['total_bill'], data['tip'], color=color)

# displaying the model
show(graph)
```

```
C:\Anaconda\lib\site-packages\bokeh\models\sources.py:177: BokehUserWarning:

ColumnDataSource's columns must be of the same length. Current lengths: ('fill_color', 256), ('x', 244), ('y', 24
4)

C:\Anaconda\lib\site-packages\bokeh\models\sources.py:177: BokehUserWarning:

ColumnDataSource's columns must be of the same length. Current lengths: ('fill_color', 256), ('line_color', 256),
('x', 244), ('y', 244)
```

Line Chart

A line plot can be created using the line() method of the plotting module.

In [2]:
```
#Importing the Libraries
from bokeh.plotting import figure, output_file, show
import pandas as pd

#Instantiating the figure Chart
graph = figure(title = "Bokeh Bar Chart")

#Reading the Database
data = pd.read_csv("tips.csv")

#Count of each unique Value of
#Tip Column
df = data['tip'].value_counts()

#Plotting the Graph
graph.line(df, data['tip'])

#Displaying the Model
show(graph)
```

```
BokehUserWarning: ColumnDataSource's columns must be of the same length. Current lengths: ('x', 123), ('y', 244)
```

Bar Chart

Bar Chart can be of two types horizontal bars and vertical bars. Each can be created using the hbar() and vbar() functions of the plotting interface respectively.

In [3]:
```
#Importing the Packages
from bokeh.plotting import figure, output_file, show
import pandas as pd

#Instantiating the figure object
graph = figure(title="Bokeh Chart File")

#Reading the Database
data = pd.read_csv("tips.csv")

#Plotting the Graph
graph.vbar(data['total_bill'], top=data['tip'])

#Plotting the Graph
show(graph)
```

Interactive Data Visualization.

In [9]:
```
#Importing the Packages
from bokeh.plotting import figure, output_file, show
import pandas as pd

#Instantiating the Figure Object
graph = figure(title="Bokeh Chart Title")

#Reading the Database
data = pd.read_csv("tips.csv")

#Plotting the Graoh
graph.vbar(data['total_bill'],top=data['tip'],legend_label='Bill Vs Tips', color = "green")

graph.vbar(data['tip'],top=data['size'],legend_label='Tips Vs Bill', color = "red")

graph.legend.click_policy = "hide"

#Plot Show
show(graph)
```

Adding Widgets

Adding Widgets

Bokeh provides GUI features similar to HTML forms like buttons, sliders, checkboxes, etc. These provide an interactive interface to the plot that allows changing the parameters of the plot, modifying plot data, etc. Let's see how to use and add some commonly used widgets.

Buttons: This widget adds a simple button widget to the plot. We have to pass a custom JavaScript function to the CustomJS() method of the models class. CheckboxGroup: Adds a standard check box to the plot. Similarly to buttons we have to pass the custom JavaScript function to the CustomJS() method of the models class. RadioGroup: Adds a simple radio button and accepts a custom JavaScript function.

In [10]:
```
from bokeh.io import show
from bokeh.models import Button, CheckboxGroup, RadioGroup, CustomJS

button = Button(label="GFG")

button.js_on_click(CustomJS(
        code="console.log('button: click!', this.toString())"))

# Labels for checkbox and radio
# buttons
L = ["First", "Second", "Third"]

# the active parameter sets checks the selected value
# by default
checkbox_group = CheckboxGroup(labels=L, active=[0, 2])

checkbox_group.js_on_click(CustomJS(code="""
        console.log('checkbox_group: active=' + this.active, this.toString())
"""))

# the active parameter sets checks the selected value
# by default
radio_group = RadioGroup(labels=L, active=1)

radio_group.js_on_click(CustomJS(code="""
        console.log('radio_group: active=' + this.active, this.toString())
"""))

show(button)
show(checkbox_group)
show(radio_group)
```

Sliders: Adds a slider to the plot. It also needs a custom JavaScript function.

In [11]:
```
from bokeh.io import show
from bokeh.models import CustomJS, Slider

slider = Slider(start=1, end=20, value=1, step=2, title="Slider")

slider.js_on_change("value", CustomJS(code="""
        console.log('slider: value=' + this.value, this.toString())
"""))

show(slider)
```

Plotly

This is the last library of our list and you might be wondering why plotly. Here's why –

Potly has hover tool capabilities that allow us to detect any outliers or anomalies in numerous data points. It allows more customization. It makes the graph visually more attractive.

Scatter Plot

Scatter plot in Plotly can be created using the scatter() method of plotly.express. Like Seaborn, an extra data argument is also required here.

In [10]:
```
import plotly.express as px
import pandas as pd

#Reading the Database
data = pd.read_csv("tips.csv")

#Plotting the scatter plot
fig = px.scatter(data, x="day", y="tip",color="sex")
```

```
#Showing the Plot
fig.show()
```

Line Chart

Line plot in Plotly is much accessible and illustrious annexation to plotly which manage a variety of types of data and assemble easy-to-style statistic. With px.line each data position is represented as a vertex

In [11]:

```
import plotly.express as px
import pandas as pd

#Reading the Database
data = pd.read_csv("tips.csv")

#Plotting the scatter plot
fig = px.line(data, y="tip", color="sex")

#Showing the Plot
fig.show()
```

Bar Chart

Bar Chart in Plotly can be created using the bar() method of plotly.express class.

In [5]:
```
import plotly.express as px
import pandas as pd

# reading the database
data = pd.read_csv("tips.csv")

# plotting the scatter chart
fig = px.bar(data, x='day', y='tip', color='sex')

# showing the plot
fig.show()
```

Histogram

In plotly, histograms can be created using the histogram() function of the plotly.express class.

In [6]:
```
import plotly.express as px
import pandas as pd

#Reading the Database
data = pd.read_csv("tips.csv")

#Plotting the Graph
fig = px.histogram(data, x = "total_bill", color = "sex")

#Plot the figure
fig.show()
```

Adding Buttons: In plotly, actions custom Buttons are used to quickly make actions directly from a record. Custom Buttons can be added to page layouts in CRM, Marketing, and Custom Apps. There are also 4 possible methods that can be applied in custom buttons:

restyle: modify data or data attributes relayout: modify layout attributes update: modify data and layout attributes animate: start or pause an animation

In [7]:

```
import plotly.graph_objects as px
import pandas as pd

# reading the database
data = pd.read_csv("tips.csv")


plot = px.Figure(data=[px.Scatter(
        x=data['day'],
        y=data['tip'],
        mode='markers',)
])

# Add dropdown
plot.update_layout(
        updatemenus=[
                dict(
                        type="buttons",
                        direction="left",
                        buttons=list([
                                dict(
                                        args=["type", "scatter"],
                                        label="Scatter Plot",
                                        method="restyle"
                                ),
                                dict(
                                        args=["type", "bar"],
                                        label="Bar Chart",
                                        method="restyle"
                                )
                        ]),
                ),
        ]
)

plot.show()
```

Creating Sliders and Selectors:

In plotly, the range slider is a custom range-type input control. It allows selecting a value or a range of values between a specified minimum and maximum range. And the range selector is a tool for selecting ranges to display within the chart. It provides buttons to select pre-configured ranges in the chart. It also provides input boxes where the minimum and maximum dates can be manually input

In [8]:

```
import plotly.graph_objects as px
import pandas as pd

# reading the database
data = pd.read_csv("tips.csv")

plot = px.Figure(data=[px.Scatter(
        y=data['tip'],
        mode='lines',)
])

plot.update_layout(
        xaxis=dict(
                rangeselector=dict(
                        buttons=list([
                                dict(count=1,
                                        step="day",
                                        stepmode="backward"),
                        ])
                ),
                rangeslider=dict(
                        visible=True
                ),
        )
)

plot.show()
```

Conclusion

In this tutorial, I have plotted the tips dataset with the help of the four different plotting modules of Python namely Matplotlib, Seaborn, Bokeh, and Plotly. Each module showed the plot in its own unique way and each one has its own set of features like Matplotlib provides more flexibility but at the cost of writing more code whereas Seaborn being a high-level language provides allows one to achieve the same goal with a small amount of code. Each module can be used depending on the task we want to do.