Data Visualization Using Python

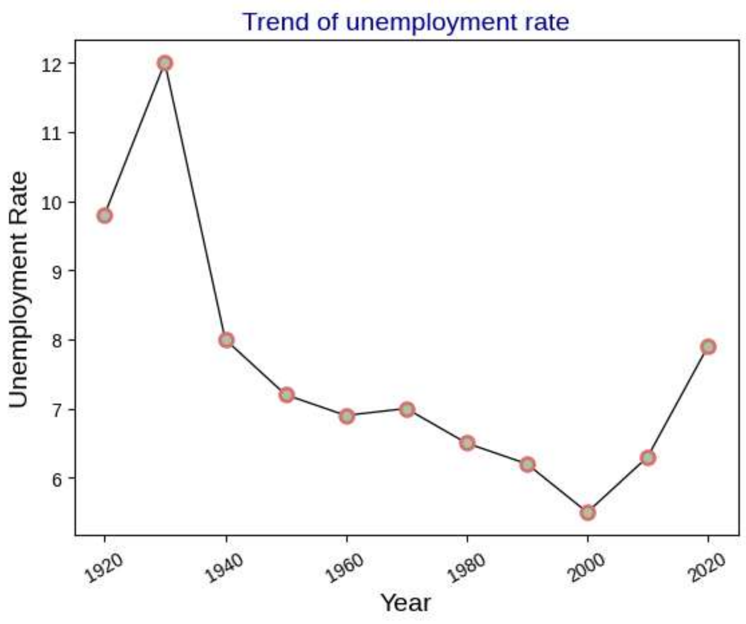
```
#importing the libraries

import numpy as np #linear algebra
import pandas as pd #data preprocessing
import matplotlib.pyplot as plt #visualization similar to ggplot2 on R
import seaborn as sns
import plotly as py
import plotly.express as px
import plotly.graph_objects as go
```

```
# Data 1

Data1 = {'Year': [1920,1930,1940,1950,1960,1970,1980,1990,2000,2010, 2020],
        'Unemployment_Rate': [9.8,12,8,7.2,6.9,7,6.5,6.2,5.5,6.3, 7.9]
       }

df = pd.DataFrame(Data1, columns = ['Year', 'Unemployment_Rate'])
```

Line Chart

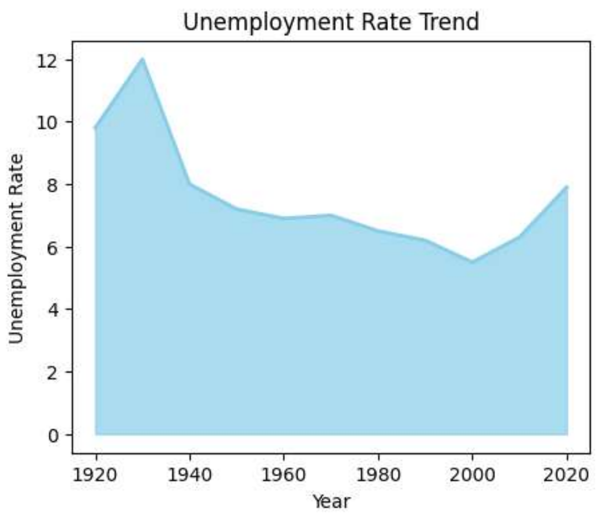
Is used in trend analysis and time series analysis. It is always a best practice to use line charts with continuous data, as the value of one variable can be found by projecting the other variable on the line at any point.

```
#Basic Line Chart

plt.plot(df.Year, df.Unemployment_Rate, marker = 'o', color = 'black',
         linewidth = 0.9, linestyle = 'solid',
         markeredgecolor = '#D37676',
         markeredgewidth = '2.0',
         markerfacecolor = '#B0C5A4', markersize = 7.0)
plt.title('Trend of unemployment rate', color = 'darkblue', size = 14)
plt.xlabel('Year', size = 14)
plt.ylabel('Unemployment Rate', size = 14)
plt.style.use('seaborn-v0_8') #plot styles
plt.grid(False) #adds grid lines in both axis
plt.xticks(rotation = 30) #rotated the x-axis tick labels by 30 degrees clockwise
plt.show()
```



```
#Area Chart

fig = plt.figure(figsize = (5,4))
plt.fill_between(df['Year'], df['Unemployment_Rate'], color='skyblue',
alpha=0.7)
plt.plot(df['Year'], df['Unemployment_Rate'], color='skyblue', linewidth = 2)
plt.xlabel("Year", size = 10)
plt.ylabel("Unemployment Rate", size = 10)
plt.title("Unemployment Rate Trend", size = 12)
```

```
Text(0.5, 1.0, 'Unemployment Rate Trend')
```

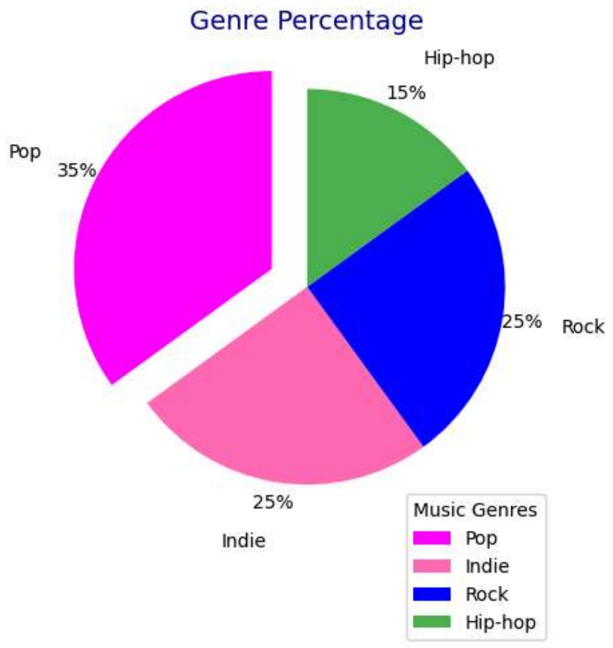


Pie Chart

Is useful in showing proportions of different categories. It is a great representation to analyse part-to-whole relationship. Pie Charts are ideal to visualize categorical data, but can also be used for discrete data (eg. shoe size, number of countries visited, etc.).

Note : The chart can get cluttered and difficult to read if the values in a discrete data are a lot.

```
y = np.array([35, 25, 25, 15])
mylabels = ["Pop", "Indie", "Rock", "Hip-hop"]
myexplode = [0.2, 0, 0, 0]
mycolors = ["magenta", "hotpink", "b", "#4CAF50"]

plt.pie(y, labels = mylabels, startangle = 90,
        explode = myexplode,
        colors = mycolors, autopct='%1.0f%%',
        pctdistance=1.1,
        labeldistance=1.3)
plt.title('Genre Percentage', color = 'darkblue', size = 14)
plt.legend(title = "Music Genres",
           bbox_to_anchor=(1,0.1))
plt.show()

#trial
#pct is changed into '%.2f'
```



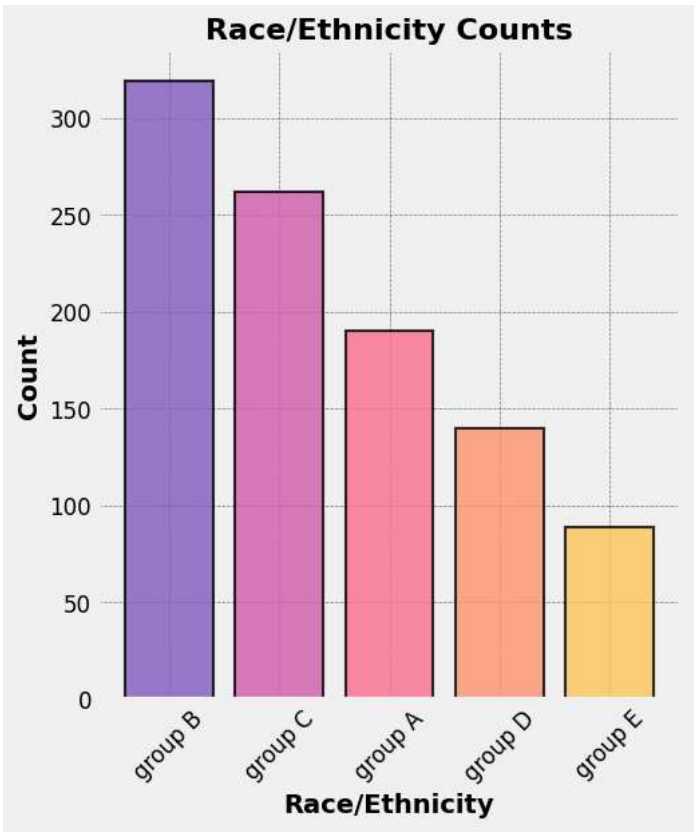
Bar Chart

Is generally used discrete or categorical data. A bar chart can be horizontal or vertical.

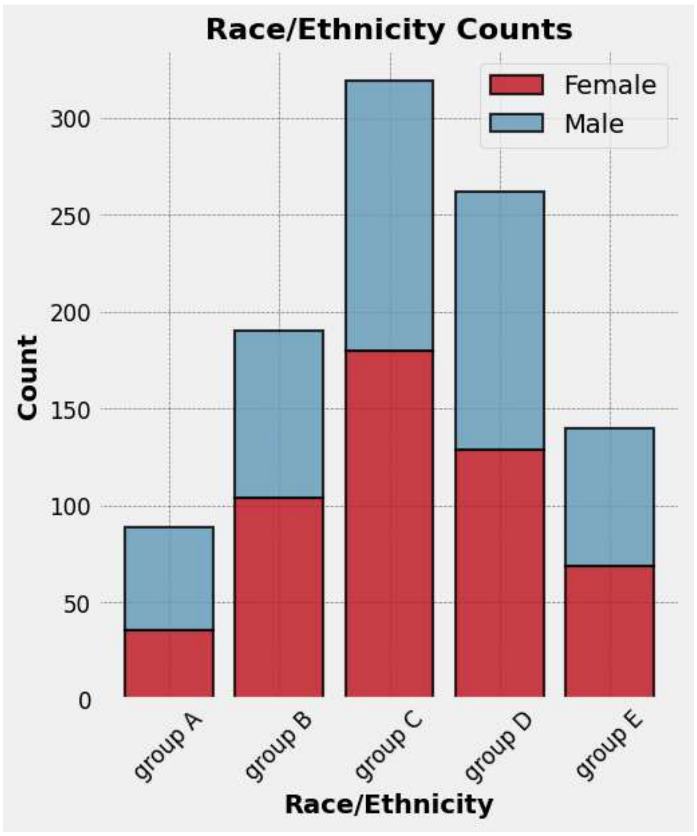
```
from google.colab import drive

drive.mount('/content/drive')
```

```
    Mounted at /content/drive
```

```
#Basic Bar Chart
df2 = pd.read_csv('/content/drive/MyDrive/Data Mining and Visualization - IUP 2024/Week 2/StudentsPerformance.csv')
df2.head()

fig = plt.figure(figsize=(5, 6))
plt.style.use('fivethirtyeight')
plt.bar(
    x=df2["race/ethnicity"].unique(),
    height=df2["race/ethnicity"].value_counts().to_list(),
    color=['#845EC2','#D65DB1','#FF6F91', '#FF9671', '#FFC75F'],         # Set the color of the bars
    edgecolor='black',    # Set the color of the bar edges
    linewidth=1.5,        # Set the width of the bar edges
    alpha=0.8             # Set the transparency of the bars
)

plt.xlabel('Race/Ethnicity', fontsize=14, fontweight='bold')  # Set the x-axis label with font size and style
plt.ylabel('Count', fontsize=14, fontweight='bold')           # Set the y-axis label with font size and style
plt.title('Race/Ethnicity Counts', fontsize=16, fontweight='bold')  # Set the chart title with font size and style
plt.xticks(fontsize=12, rotation = 45)    # Set the font size of the x-axis tick labels
plt.yticks(fontsize=12)    # Set the font size of the y-axis tick labels
plt.grid(True, linestyle='--', linewidth=0.5, alpha=0.5, color = "black")  # Display gridlines with a dashed style and reduced opacity

plt.show()
```

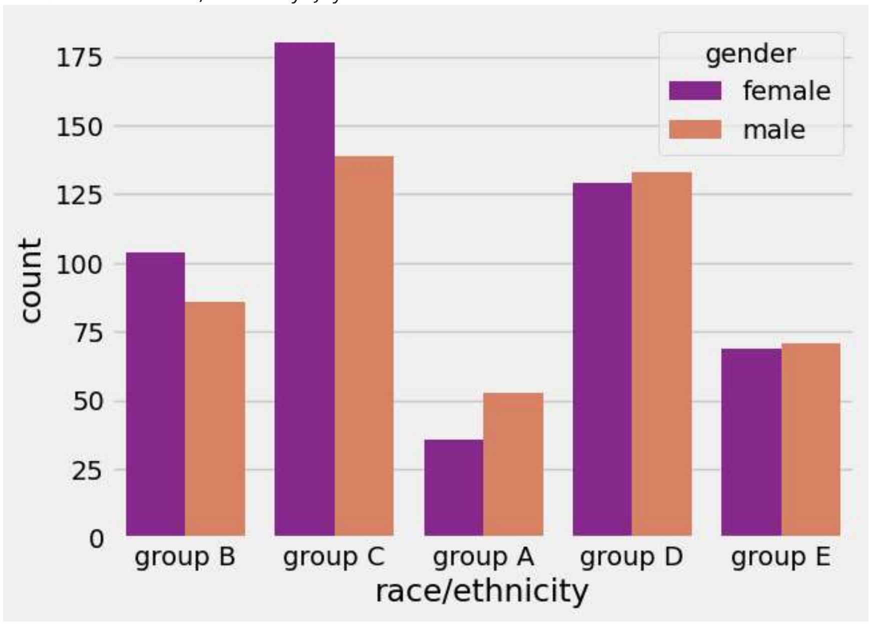


```
#Stacked Bar Chart
grouped_data = df2.groupby(['race/ethnicity', 'gender']).size().unstack()

fig = plt.figure(figsize=(5, 6))
plt.style.use('fivethirtyeight')
plt.bar(
    x=grouped_data.index,
    height= grouped_data["female"],
    label = "Female",
    color='#C1121F',      # Set the color of the bars
    edgecolor='black',    # Set the color of the bar edges
    linewidth=1.5,        # Set the width of the bar edges
    alpha=0.8             # Set the transparency of the bars
)

plt.bar(
    x=grouped_data.index,
    height= grouped_data["male"],
    bottom = grouped_data["female"],
    label = "Male",
    color='#669BBC',      # Set the color of the bars
    edgecolor='black',    # Set the color of the bar edges
    linewidth=1.5,        # Set the width of the bar edges
    alpha=0.8             # Set the transparency of the bars
)

plt.xlabel('Race/Ethnicity', fontsize=14, fontweight='bold')  # Set the x-axis label with font size and style
plt.ylabel('Count', fontsize=14, fontweight='bold')           # Set the y-axis label with font size and style
plt.title('Race/Ethnicity Counts', fontsize=16, fontweight='bold')  # Set the chart title with font size and style
plt.xticks(fontsize=12, rotation = 45)    # Set the font size of the x-axis tick labels
plt.yticks(fontsize=12)    # Set the font size of the y-axis tick labels
plt.grid(True, linestyle='--', linewidth=0.5, alpha=0.5, color = "black")# Display gridlines with a dashed style and reduced opacity
plt.legend()

plt.show()
```

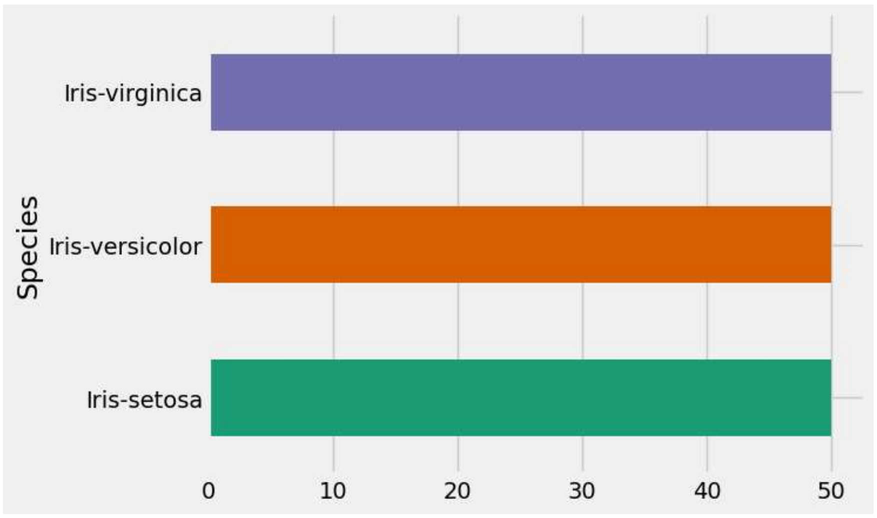


```
sns.countplot(data = df2, x = "race/ethnicity", hue = "gender",
              palette = "plasma")
```

```
<Axes: xlabel='race/ethnicity', ylabel='count'>
```



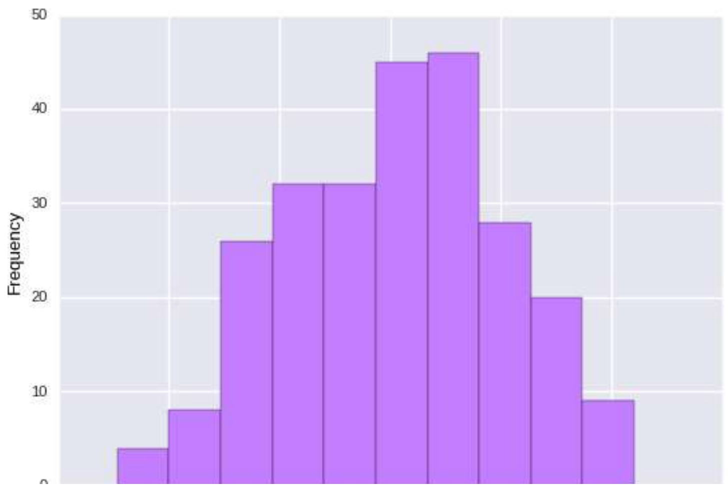
```
df1 = pd.read_csv('/content/drive/MyDrive/Data Mining and Visualization - IUP 2024/Week 2/Iris.csv')

# Bar Chart

from matplotlib import pyplot as plt
import seaborn as sns

df1.groupby('Species').size().plot(kind='barh', color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```



Histogram

Is used to observe the frequency distribution of a variable. The variable is divided into different intervals and the length of bars of the histogram show the frequency of the variable for a particular interval. Histograms can be used to observe discrete as well as continuous data.

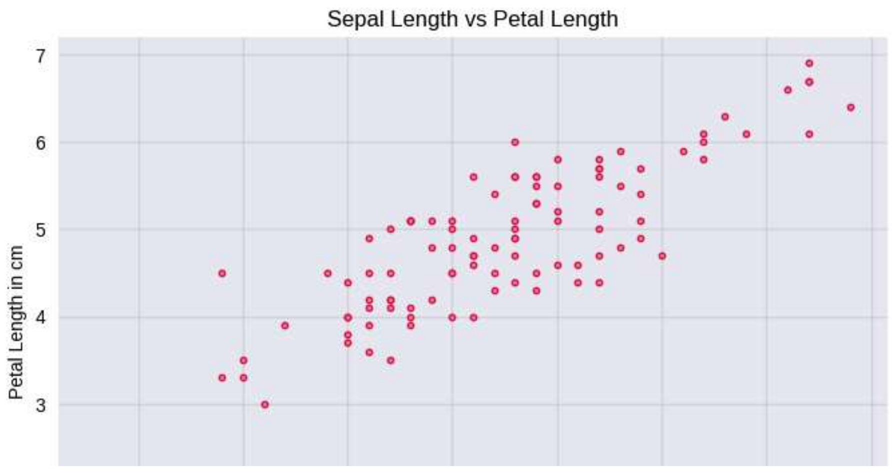
```
import numpy as np

#Getting the data
x = np.random.normal(170, 10, 250)
#print(x)

plt.style.use('seaborn-v0_8')
plt.hist(x, color = '#C77DFF')
plt.xlabel("Student Height", size = 12, color = "black")
plt.ylabel("Frequency", size = 12, color = "black")
plt.show()
```



Scatter Plot

Is used to study the correlation between the two variables. If one variable increases or decreases with other, they are positively correlated. If one variable increases as the other decreases and vice versa, they are negatively correlated. If there is no such relation, they aren't related.

```
df3 = pd.read_csv('/content/drive/MyDrive/Data Mining and Visualization - IUP 2024/Week 2/Iris.csv')

plt.scatter(df3.SepalLengthCm, df3.PetalLengthCm, marker = "o",
            color = "#FF758F", linewidths = 1, edgecolors = "#C9184A", s = 10)
plt.style.use('seaborn-v0_8')
plt.xlabel("Sepal Length in cm", size = 10, color = "black")
plt.ylabel("Petal Length in cm", size = 10, color = "black")
plt.title("Sepal Length vs Petal Length", size =12, color = "black")
plt.xticks(color = "black")
plt.yticks(color = "black")
plt.grid(color = "grey", alpha = 0.2)
plt.show()
```


Sepal Length vs Petal Length
Petal Length in cm