ਜਗਤ ਗੁਰੂ ਨਾਨਕ ਦੇਵ
ਪੰਜਾਬ ਸਟੇਟ ਓਪਨ ਯੂਨੀਵਰਸਿਟੀ
ਪਟਿਆਲਾ

# JAGAT GURU NANAK DEV
# PUNJAB STATE OPEN UNIVERSITY, PATIALA

**(Established by Act No. 19 of 2019 of the Legislature of State of Punjab)**

## The Motto of the University
## (SEWA)

**SKILL ENHANCEMENT** **EMPLOYABILITY** **WISDOM**
**ACCESSIBILITY**


## M.SC. (COMPUTER SCIENCE)
## SEMESTER-III
## COURSE: DATA MINING & VISUALIZATION LAB

## COURSE CODE: MSCS-3-02P

**ADDRESS: C/28, THE LOWER MALL, PATIALA-147001**
**WEBSITE: www.psou.ac.in**

# M.Sc. (Computer Science)
# Semester-3
# MSCS-3-02P: Data Mining & Visualization Lab

**Total Marks: 50**
**External Marks: 15**
**Internal Marks: 35**
**Credits: 2**
**Pass Percentage: 40%**

| **Course: Data Mining & Visualization Lab** | |
|---|---|
| **Course Code: MSCS-3-02P** | |
| **Course Outcomes (COs)**<br>After the completion of this course, the students will be able to: | |
| CO1 | Explore WEKA Data Mining/Machine Learning Toolkit. |
| CO2 | Perform data pre-processing tasks and Demonstrate performing association rule mining on data sets. |
| CO3 | Demonstrate the performance of Naïve-Bayes and K-Nearest Neighbor classifiers on data sets. |
| CO4 | Evaluate the performance of Naïve-Bayes and k-Nearest Neighbor classifiers through ROC Curves |
| CO5 | Explore visualization features of Weka to visualize the clusters. |

**Exp1. Explore WEKA Data Mining/Machine Learning Toolkit**

- Downloading and/or installation of WEKA data mining toolkit,
- Understand the features of WEKA toolkit such as Explorer, Knowledge Flow interface,
- Experimenter, command-line interface.
- Navigate the options available in the WEKA (ex. Select attributes panel, Preprocess panel, classify panel, Cluster panel, Associate panel and Visualize panel)
- Study the arff file format
- Explore the available data sets in WEKA.
- Load a data set (ex. Weather dataset, Iris dataset, etc.)
- Load each dataset and observe the following:
  - List the attribute names and they types
  - Number of records in each dataset

- Identify the class attribute (if any)
- Plot Histogram
- Determine the number of records for each class.
- Visualize the data in various dimensions

**Exp2. Perform data pre-processing tasks and Demonstrate performing association rule mining on data sets.**

**Exp3. Demonstrate performing classification on data sets:**

- Load each dataset into Weka and run Id3, J48 classification algorithm. Study the classifier output. Compute entropy values, Kappa statistic. Extract if-then rules from the decision tree generated by the classifier, Observe the confusion matrix.
- Load each dataset into Weka and perform Naïve-Bayes classification and k-Nearest Neighbor classification. Interpret the results obtained.
- Plot ROC Curves and Compare classification results of ID3, J48, Naïve-Bayes and k-NN classifiers for each dataset, and deduce which classifier is performing best and poor for each dataset and justify.

**Exp4. Demonstrate performing clustering of data sets:**

- Load each dataset into Weka and run simple k-means clustering algorithm with different values of k (number of desired clusters). Study the clusters formed. Observe the sum of squared errors and centroids, and derive insights.
- Explore other clustering techniques available in Weka
- Explore visualization features of Weka to visualize the clusters. Derive interesting insights and explain.

# MODULE - VII

# EXPLORING THE VISUAL DATA SPECTRUM

:: TOPICS ::

- Data Points
- Line Chart
- Bar Chart
- Pie Chart
- Area Chart
- Candlestick Chart
- Bubble Chart
- Surface Plot
- Map Chart
- Infographics

Dr. Chetan R. Dudhagara

Assistant Professor and Head
Department of Communication & Information Technology
International Agribusiness Management Institute
Anand Agricultural University
Anand, Gujarat, India

# DATA MINING AND VISUALIZATION

## Unit – VII

## EXPLORING THE VISUAL DATA SPECTRUM

**Structure**

- **Data Points**
- **Line Chart**
- **Bar Chart**
- **Pie Chart**
- **Area Chart**
- **Candlestick Chart**
- **Bubble Chart**
- **Surface Plot**
- **Map Chart**
- **Infographics**

## OBJECTIVES

The main objective of this module is to understand the data mining and visualization concepts. The various data visualization techniques and charts such as data points, line chart, bar chart, pie chart and area chart are covered in this module. The advance data visualization techniques are also covered in this module such as candlestick chart, bubble chart, surface chart, map chart and infographics.

# 1. INTRODUCTION

Data science become a buzzword that everyone talks about the data science. Data science is an interdisciplinary field that combines different domain expertise, computer programming skills, mathematics and statistical knowledge to find or extract the meaningful or unknown patterns from unstructured and structure dataset. Data science is useful for extraction, preparation, analysis and visualization of various information. Various scientific methods can be applied to get insight in data.

Data mining is a process to find the new and hidden patterns or relationship in a large data sets for solving various business or real-life problems to improve the efficiency. The various data mining software or tools are used such as Rapid Miner, SPSS, Oracle, Orange, Weka, R, Python, etc.

Data mining is used in various real-life application such as banking, marketing, retail, health care, agriculture, insurance, transportation etc.

Data visualization is a graphical representation of data and quantitative information by using various graphical tools such as graphs, charts and maps. It is more useful to understand the trends, patterns and outliers in the data.

# 2. EXPLORING DATA VISULIZATION

Charts is the representation of data in a graphical form. It helps to summarizing and presenting a large amount of data in a simple and easy to understandable formats. By placing the data in a visual context, we can easily detect or identify the patterns, trends and correlations among them.

Python provides various easy to use multiple graphics libraries for data visualization. These libraries are work with both small and large datasets.

Python has multiple graphics libraries with different features. Some of the most popular and commonly used Python data visualization libraries are Matplotlib, Pandas, Seaborn, Plotly and ggplot.

Matplotlib is a most popular, amazing and multi-platform data visualization library available in Python. It consists a wide variety of plots like data points, line chart, bar chart, pie chart, area chart etc.

## 2.1 Data Points / Scatter Plot

Data points is a mark in a diagram where each value in the dataset is representing by a dot or point. It is a set of dotted points to represent the individual data on both horizontal and vertical axis to reveal the distribution trends of data.

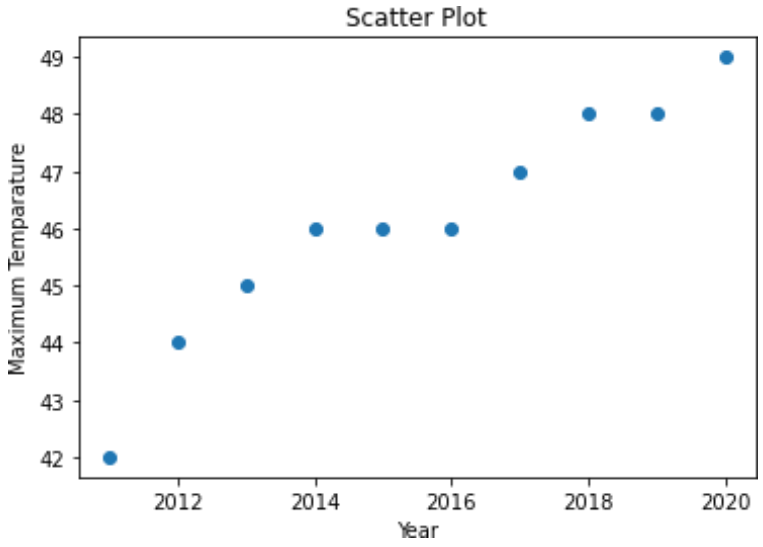
This plot is mostly used for large dataset to highlight the similarities in a dataset. It also shows the outliers and distribution of data.

The ***scatter()*** function is used to draw the scatter plot. This function plots one dot for each observation. It requires two different arrays of same length for both x-axis and y-axis. we can also set the scatter plot title and labels on both the axis.

**Example:** Here we take an example of maximum temperature noted every year. The x-axis represents **"Year"** values and y-axis represents **"Max_Temp"**. (**Note:** The maximum temperature is a dummy data use for example purpose only)

```
# Importing library
import matplotlib.pyplot as plt

# Data values
Year = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018,
2019, 2020]
Max_Temp = [42,44,45,46,46,46,47,48,48,49]

# Plotting scatter plot with title and label
plt.scatter(Year, Max_Temp)
plt.title("Scatter Plot")
plt.xlabel("Year")
plt.ylabel("Maximum Temperature")
plt.show()
```

The above code will create scatter plot as follow:

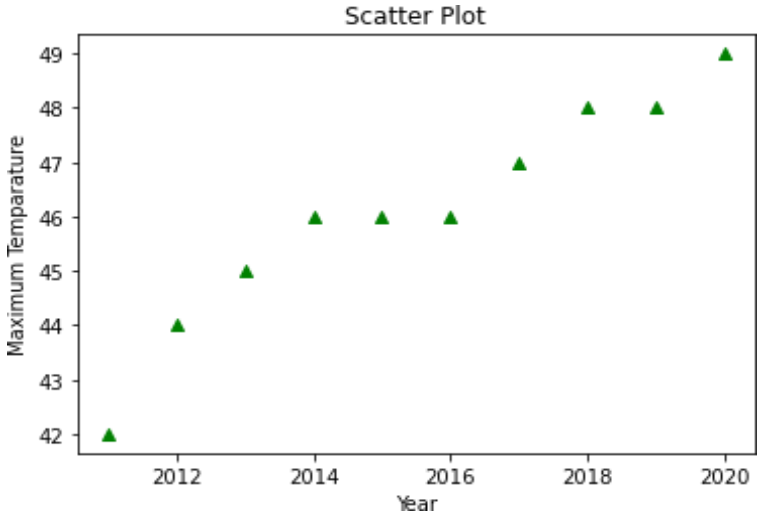



In above plot, we can see the trends of maximum temperature every year.

We can also set or change the color using ***color*** as an argument. Here we set the ***green*** color for temperature. We can also set the shape of data points using ***marker*** as an argument as follows:

```
# Importing library
import matplotlib.pyplot as plt

# Data values
Year = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018,
2019, 2020]
Max_Temp = [42,44,45,46,46,46,47,48,48,49]

# Plotting scatter plot with title and label
plt.scatter(Year, Max_Temp, color="Green", marker="^")
plt.title("Scatter Plot")
plt.xlabel("Year")
plt.ylabel("Maximum Temperature")
plt.show()
```

The above code will create scatter plot as follow:




## 2.2 Line Chart

Line chart is used to show the relation between two datasets on a different axis. There are multiple features available such as line color, line style, line width etc.

Matplotlib is a most popular library for plotting different chart. Line chart is one of them.

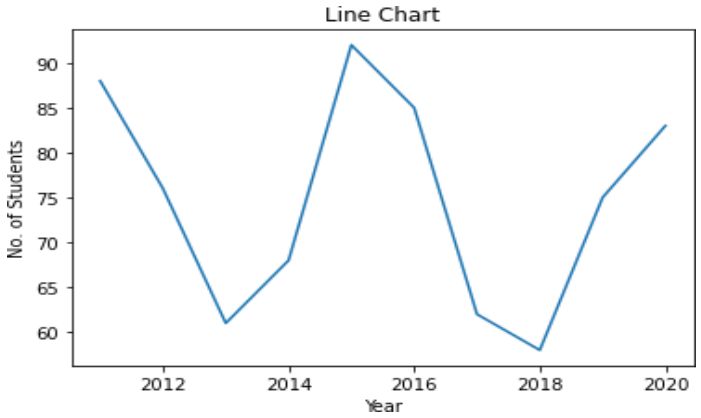
The ***plot()*** function is used to create a line chart in Python. Here we will see some examples of line chart in Python.

**Example:** Here we take an example of numbers of students enroll in specific course in different year. The x-axis represents **"Year"** values and y-axis represents **"Student"**.

```
# Importing library
from matplotlib import pyplot as plt

# Data values
```

```
Year = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018,
2019, 2020]
Student = [88,76,61,68,92,85,62,58,75,83]

# Plotting the line chart
plt.title("Line Chart")
plt.xlabel("Year")
plt.ylabel("No. of Students")
plt.plot(Year, Student)
plt.show()
```

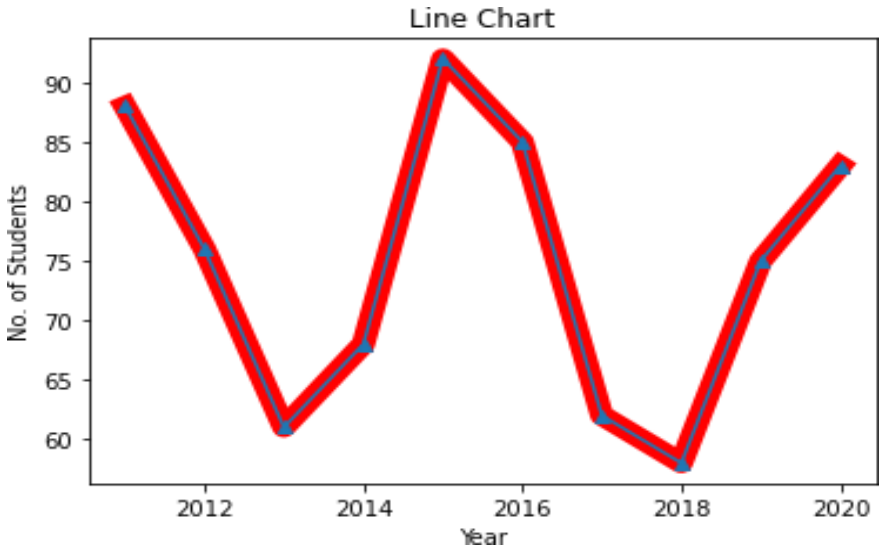
The above code will create line chart as follow:




We can set the line color using ***color*** as an argument. Here we set ***red*** color to the line. We can also set the shape of data points using ***marker*** as an argument. We can also set the line width using ***linewidth*** or ***lw*** as an argument. Here we set ***10*** to the linewidth as follows:

```
# Importing library
from matplotlib import pyplot as plt

# Data values
Year = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018,
2019, 2020]
Student = [88,76,61,68,92,85,62,58,75,83]

# Plotting the line chart
plt.title("Line Chart")
plt.xlabel("Year")
plt.ylabel("No. of Students")
plt.plot(Year, Student, color="Red", marker="^",
linewidth=10)
plt.show()
```

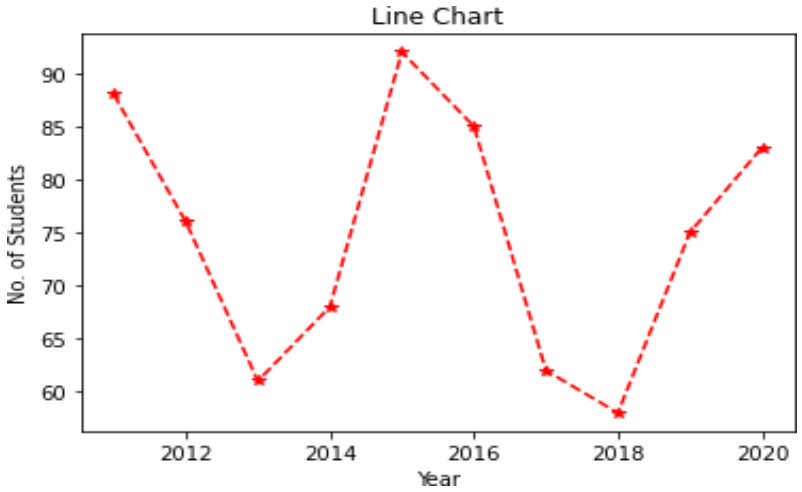
The above code will create line chart as follow:




We can set the line style also using ***linestyle*** or ***ls*** as an argument. There are various types of style available such as solid, dotted, dashed and dashdot. Here we set ***dashed*** as a linestyle in a line chart.

```
# Importing library
from matplotlib import pyplot as plt

# Data values
Year = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018,
2019, 2020]
Student = [88,76,61,68,92,85,62,58,75,83]

# Plotting the line chart
plt.title("Line Chart")
plt.xlabel("Year")
plt.ylabel("No. of Students")
plt.plot(Year, Student, marker="*", linestyle =
'dashed', color="Red")
plt.show()
```

The above code will create line chart as follow:



## 2.3 Bar Chart

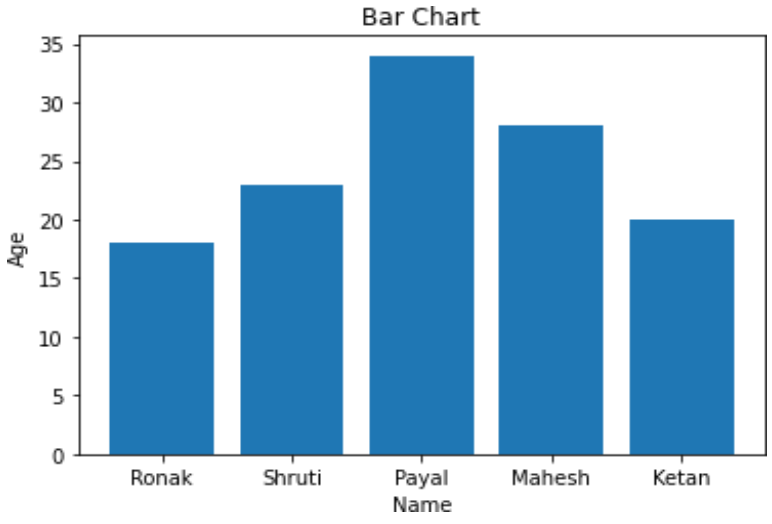
Bar chart or bar plot is representing the category of data with rectangular bars with different heights and lengths with reference to the values of that they present. The ***bar()*** function is used to create a bar chart. The bar chart can be plotted both horizontally and vertically.

The bar chart describes the comparisons between distinct categories. One axis represents the particular categories being compared and another axis represent the measured values respected to those categories. The numerical values of variables in a dataset represent the height or length of bar.

**Example:** Here we take an example of students name and age. The x-axis represents **"Name"** and y-axis represents **"Age"**. Here we also set the chart title and labels on both the axis.

```
# Importing library
from matplotlib import pyplot as plt

# Data values
Name = ["Ronak", "Shruti", "Payal", "Mahesh", "Ketan"]
Age = [18, 23, 34, 28, 20]

# Labelling the axes and title
plt.title("Bar Chart")
plt.xlabel("Name")
plt.ylabel("Age")

# Plotting the bar chart
plt.bar(Name, Age)
```

The above code will create bar chart as follow:

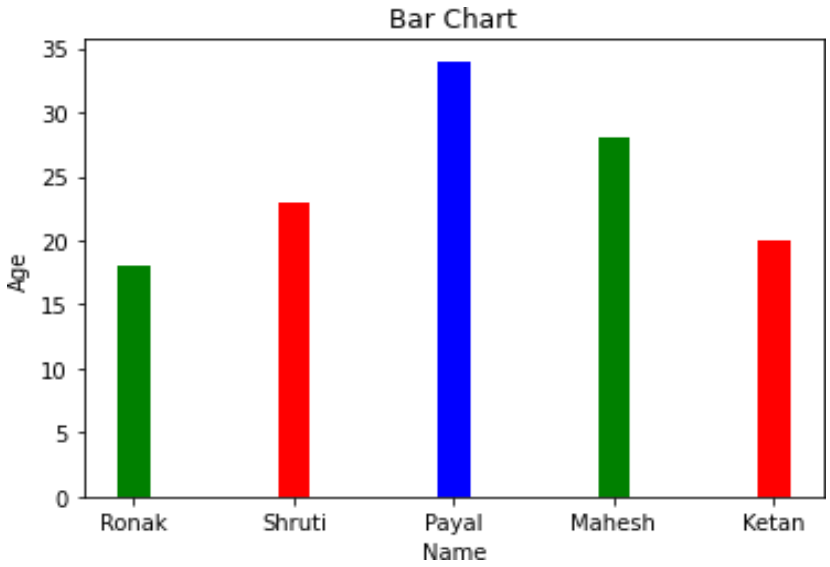



We can change the bar color also. We set ***color*** as an argument to change the color of bar. Here we set the multiple color of bar. We can set the bar width also. We set ***width*** as an argument to set the width of bar. Here we set ***0.2*** as a width of bar.

```
# Importing library
from matplotlib import pyplot as plt
```

```
# Data values
Name = ["Ronak", "Shruti", "Payal", "Mahesh", "Ketan"]
Age = [18, 23, 34, 28, 20]

# Labelling the axes and title
plt.title("Bar Chart")
plt.xlabel("Name")
plt.ylabel("Age")

# Plotting the bar chart
plt.bar(Name, Age, width=0.2, color = ["Green", "Red",
"Blue"])
```

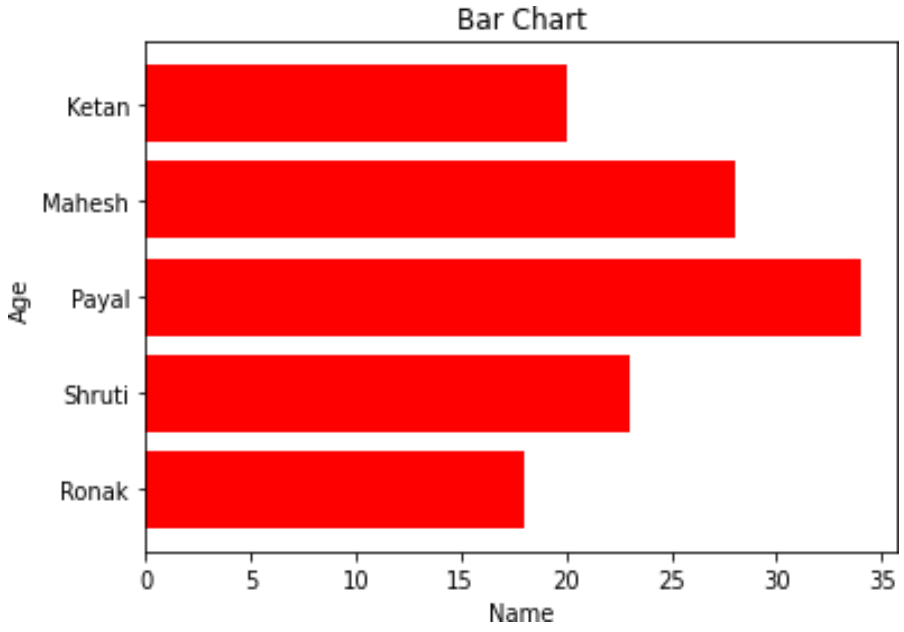
The above code will create bar chart as follow:




We can display bar horizontally also instead of vertically. We set the bar horizontally by using ***barh()*** function.

```
# Importing library
from matplotlib import pyplot as plt

# Data values
Name = ["Ronak", "Shruti", "Payal", "Mahesh", "Ketan"]
Age = [18, 23, 34, 28, 20]

# Labelling the axes and title
plt.title("Bar Chart")
plt.xlabel("Name")
plt.ylabel("Age")

# Plotting the bar chart
plt.barh(Name, Age, color="Red")
```

The above code will create bar chart as follow:



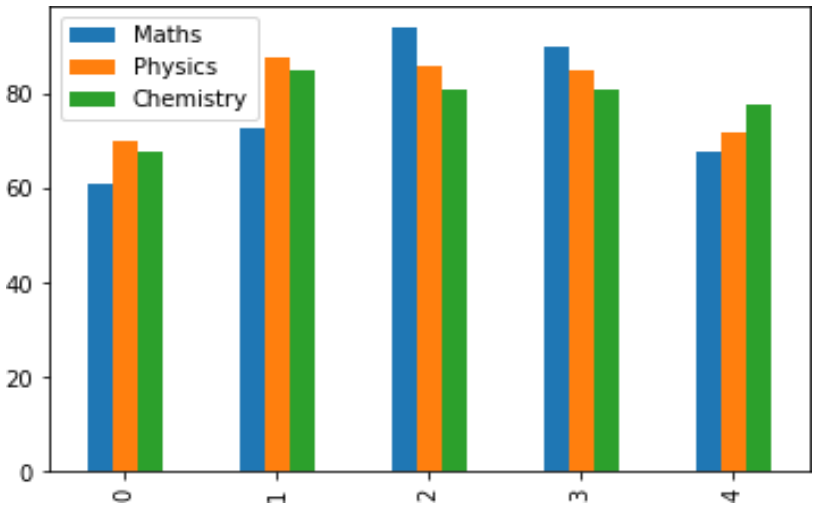

The multiple bar chart is used to represent the comparison among the different variables in a dataset.

Here, we take an example of marks of different subject with the name of students. The x-axis represents students and y-axis represents marks of different subjects.

```
# Importing library
from matplotlib import pyplot as plt
import pandas as pd

# Data values
df = pd.DataFrame({
"Name":["Ronak", "Shruti", "Payal", "Mahesh", "Ketan"],
"Maths":[61,73,94,90,68],
"Physics":[70,88,86,85,72],
"Chemistry":[68,85,81,81,78]})

# Plotting the bar chart
df.plot.bar()
```

The above code will create bar chart as follow:

Here, we show the comparisons of marks of different subjects of the students.



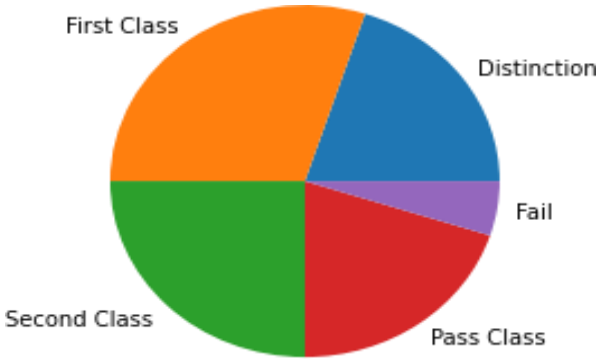

## 2.4 Pie Chart

The pie chart is a circular analytical or statistical type of graphs that represents the data in a circular plot. The total area in a pie chart is divided into different region or slices to symbolize the numerical percentage. The slices of pie are called wedges.

**Example:** Here we take an example of results of the students. The results of students are in different categories such as distinction, first class, second class, pass class and fail. Here area represent the percentage value of results of the students respectively.

```
# Importing library
import matplotlib.pyplot as plt

# Labels and area covered by each label
result = ["Distinction", "First Class", "Second Class",
"Pass Class", "Fail"]
area = [20,30,25,20,5]

# Plotting pie chart
plt.pie(area, labels=result)
plt.show()
```

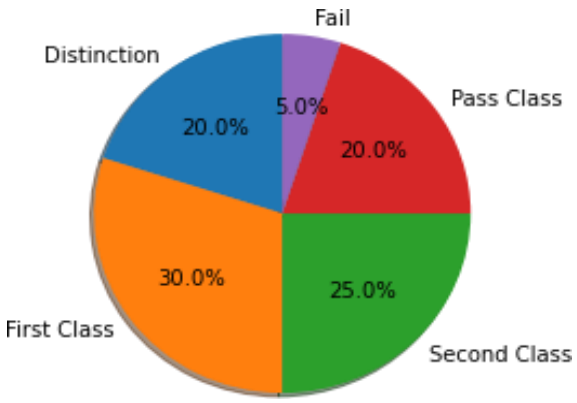
The above code will create pie chart as follow:



We can set the ***startangle*** argument to start the rotation of pie chart by given degrees. Here we set the ***startangle*** is ***90***. We can also set the ***autopct*** argument to show the percentage value. We can also set the ***shadow*** of pie chart.

```
# Importing library
import matplotlib.pyplot as plt

# Labels and area covered by each label
result = ["Distinction", "First Class", "Second Class",
"Pass Class", "Fail"]
area = [20,30,25,20,5]

# Plotting pie chart
plt.pie(area, labels=result, startangle=90,
autopct='%1.1f%%', shadow=True )
plt.show()
```

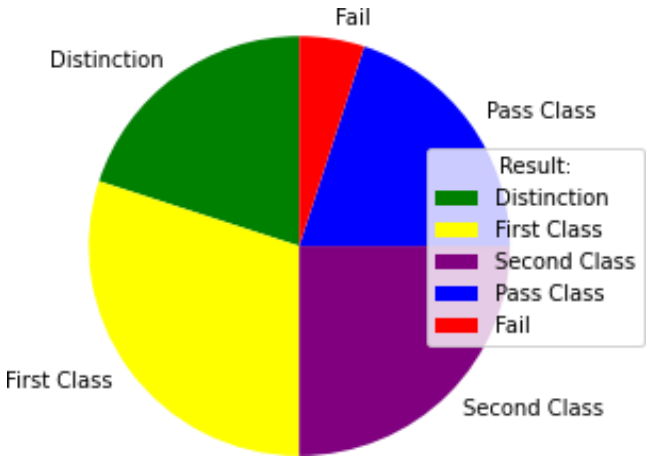
The above code will create pie chart as follow:




We can set the different colors of each slices in a pie chart using ***colors*** argument. We can also set the legend and location of legend in pie chart.

```
# Importing library
import matplotlib.pyplot as plt

# Labels and area covered by each label
result = ["Distinction", "First Class", "Second Class",
"Pass Class", "Fail"]
area = [20,30,25,20,5]
mycolor = ["Green", "Yellow", "Purple", "Blue", "Red"]

# Plotting pie chart
plt.pie(area, labels=result, startangle=90,
colors=mycolor)
plt.legend(title="Result:", loc="right")
plt.axis('equal')
plt.show()
```

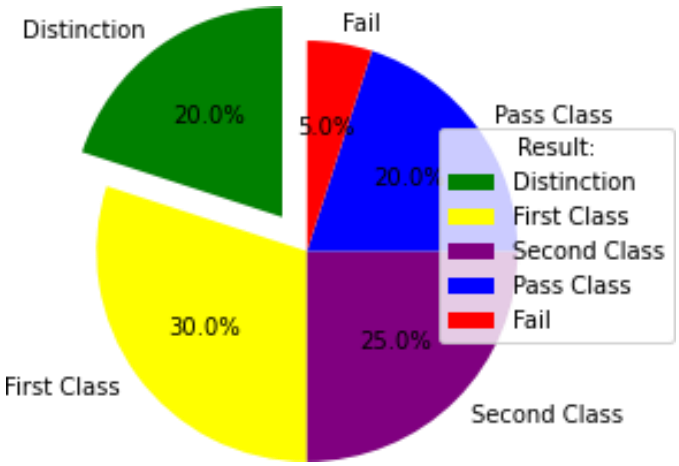
The above code will create pie chart as follow:




We can set the ***explode*** argument in a pie chart to set the fraction of radius with which we offset each wedge.

```
# Importing library
import matplotlib.pyplot as plt

# Labels and area covered by each label
result = ["Distinction", "First Class", "Second Class",
"Pass Class", "Fail"]
area = [20,30,25,20,5]
mycolor = ["Green", "Yellow", "Purple", "Blue", "Red"]
myexplodes = [0.2,0,0,0,0]

# Plotting pie chart
plt.pie(area, labels=result, startangle=90,
colors=mycolor, explode=explodes, autopct='%1.1f%%')
plt.legend(title="Result:", loc="right")
plt.axis('equal')
plt.show()
```

The above code will create pie chart as follow:




## 2.5 Area Chart

An area chart is similar as a line chart excepts that the area between an x-axis and a line is filled with color or shading. The matplotlib library is used to build the area chart. There are two different functions is used to build an area chart with matplotlib.

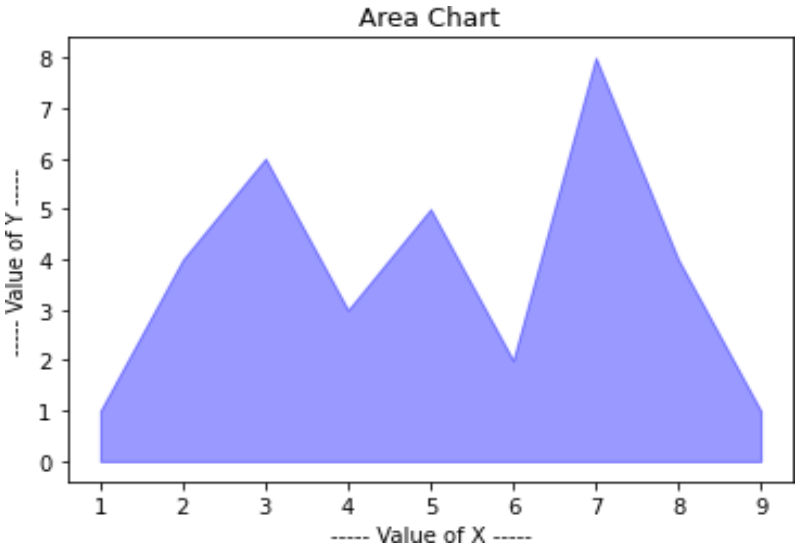
- fill_between() function
- stackplot() function

The ***fill_between()*** function is used to plot an area chart.

**Example:** Here we take an example of two dataset x and y. The x-axis represents the x values and y-axis represents the y values. Here we also set the chart title, labels on both the axis and ***color*** as a ***Blue***.

```
# Importing library
import matplotlib.pyplot as plt
import numpy as np

# Dataset
x=range(1,10)
y=[1,4,6,3,5,2,8,4,1]

# Titles of chart
plt.title("Area Chart")
plt.xlabel("----- Value of X----- ")
plt.ylabel("----- Value of Y----- ")

# Plotting area chart
plt.fill_between(x,y,color="Blue",alpha=0.4)
plt.show()
```

The above code will create area chart as follow:

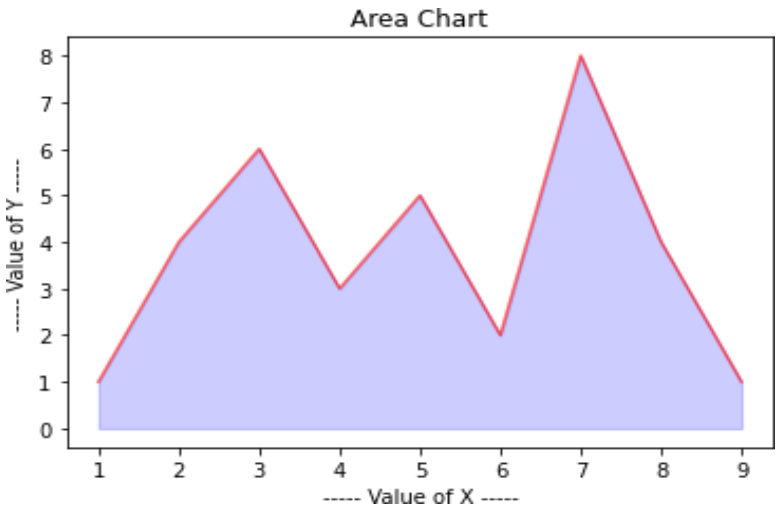



We can change the line color also. We set ***color*** as an argument to change the color. Here we set ***Red*** as a color of line.

```
# Importing library
import matplotlib.pyplot as plt
import numpy as np
```

```
# Dataset
x=range(1,10)
y=[1,4,6,3,5,2,8,4,1]
# Titles of chart
plt.title("Area Chart")
plt.xlabel("----- Value of X----- ")
plt.ylabel("----- Value of Y----- ")
# Plotting area chart
plt.fill_between( x, y, color="Blue", alpha=0.2)
plt.plot(x, y, color="Red", alpha=0.6)
plt.show()
```

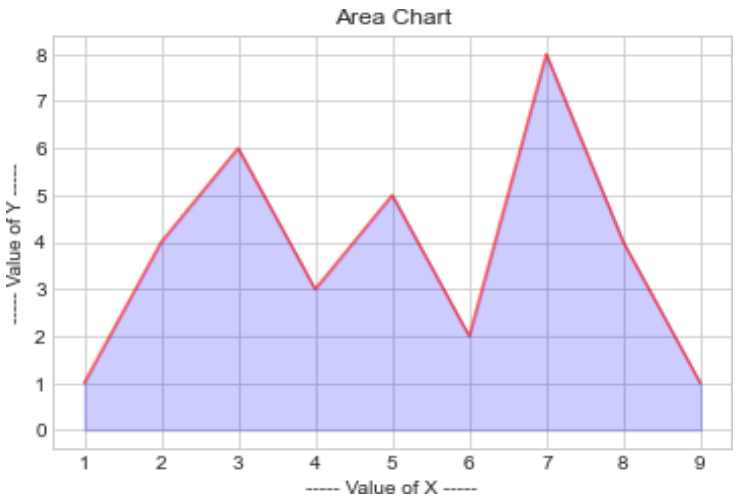
The above code will create area chart as follow:




We can also set and use the plotting style. The various plotting style can be used such as ***fivethirtyeight, seaborn-pastel, seaborn-whitegrid, dark_background*** etc. Here we set ***seaborn-whitegrid*** as a plotting style.

```
# Importing library
import matplotlib.pyplot as plt
import numpy as np
# Dataset
x=range(1,10)
y=[1,4,6,3,5,2,8,4,1]
# Titles of chart
plt.title("Area Chart")
plt.xlabel("----- Value of X----- ")
plt.ylabel("----- Value of Y----- ")
# Plotting style
plt.style.use('dark_background')
# Plotting area chart
plt.fill_between( x, y, color="Blue", alpha=0.2)
plt.plot(x, y, color="Red", alpha=0.6)
plt.show()
```

The above code will create area chart as follow:

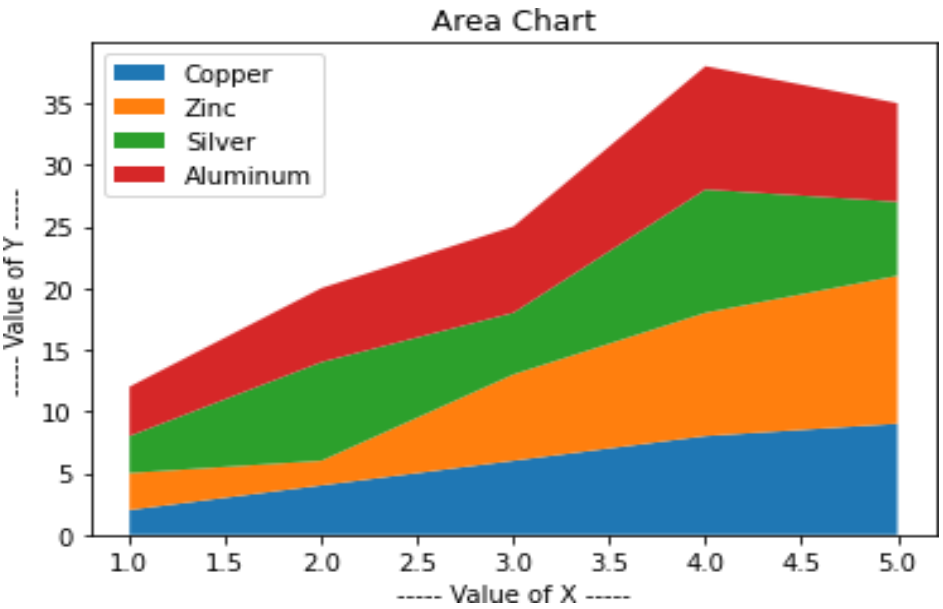



The ***stackplot()*** function is also used to plot an area chart.

**Example:** Here we take an example of metals such as copper, zinc, silver and aluminum. The x-axis represents the x values and y-axis represents the different metals. Here we also set the chart title, labels on both the axis and chart legend. Here we set ***upper left*** as a location for chart legend.

```
# Importing library
import matplotlib.pyplot as plt
import numpy as np

# Dataset
x=range(1,6)
Copper=[2,4,6,8,9]
Zinc=[3,2,7,10,12]
Silver=[3,8,5,10,6]
Aluminum=[4,6,7,10,8]

# Titles of chart
plt.title("Area Chart")
plt.xlabel("----- Value of X----- ")
plt.ylabel("----- Value of Y----- ")

# Plotting stacked area chart
plt.stackplot(x, Copper, Zinc, Silver, Aluminum,
labels=['Copper','Zinc','Silver','Aluminum'])
plt.legend(loc='upper left')
```

The above code will create area chart as follow:




## 2.6 Candlestick Chart

The candlestick chart is also known as Japanese chart. This type of charts mostly used for technical analysis in trading for visualizing of price with specified time period. The ***mpl_finance*** module of matplotlib is used to create this chart in Python.

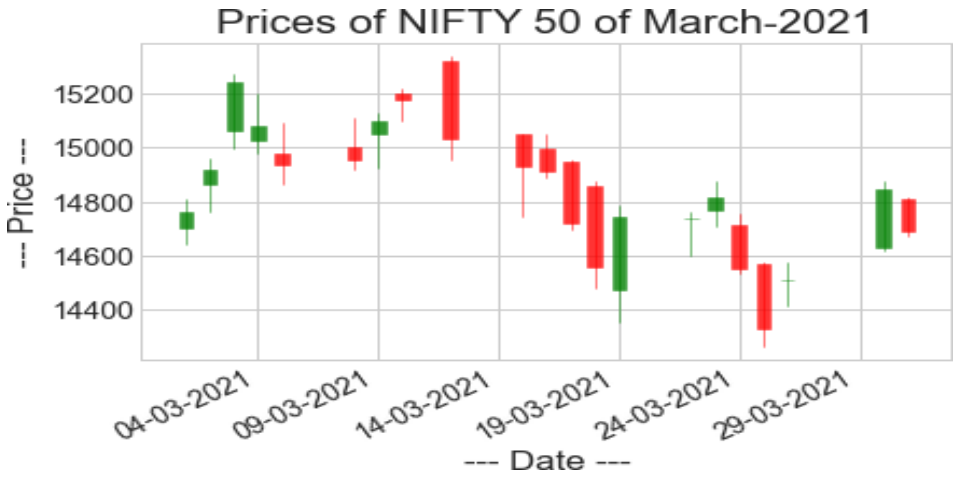
**Example:** Here we take an example of historical index data of Nifty 50 which was downloaded from NSE website. They have four points such as Open, High, Low and Close (OHLC). Here we set ***colorup*** as a ***green*** and ***colordown*** as a ***red*** argument.

```
# Importing library
import matplotlib.pyplot as plt
from mpl_finance import candlestick_ohlc
import pandas as pd
import matplotlib.dates as mpdates

# Dataset reading and extracting
df = pd.read_csv('Nifty50-March.csv')
df = df[['Date', 'Open', 'High', 'Low', 'Close']]

# Converting datetime object and applying map function
df['Date'] = pd.to_datetime(df['Date'])
df['Date'] = df['Date'].map(mpdates.date2num)

# Creating subplots
fig, ax = plt.subplots()

# Plotting candlestick chart
candlestick_ohlc(ax, df.values, width = 0.6, colorup =
'green', colordown = 'red', alpha = 0.8)
ax.grid(True)

# Plotting style
plt.style.use('seaborn-whitegrid')

# Setting title and labels of chart
plt.title('Prices of NIFTY 50 of March-2021')
```

```
ax.set_xlabel('--- Date ---')
ax.set_ylabel('--- Price ---')

# Date formatting
dt_format = mpdates.DateFormatter('%d-%m-%Y')
ax.xaxis.set_major_formatter(dt_format)
fig.autofmt_xdate()

# Show plot
plt.show()
```

**Note:** The above historical index data of Nifty 50 was downloaded from NSE website.

The above code will create candlestick chart as follow:




## 2.7 Bubble Chart

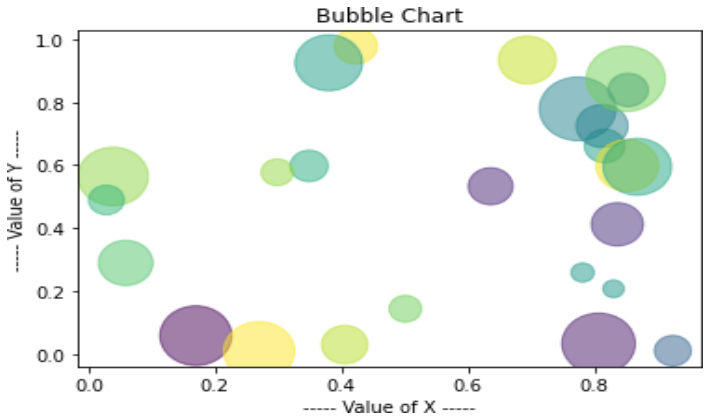
The bubble chart is very similar to the scatter plot. It displays the data as a cluster of circles. The ***scatter()*** function of matplotlib library is used to create a bubble chart in Python. To create bubble chart, it required the data of xy coordinates, size and color of bubbles.

**Example:** Here we create random numbers for xy coordinates and size of bubbles. Here we set the random colors for bubbles.

```
# Importing library
import matplotlib.pyplot as plt
import numpy as np

# Dataset
x = np.random.rand(25)
y = np.random.rand(25)
z = np.random.rand(25)
colors = np.random.rand(25)

# Titles of chart
plt.title("Bubble Chart")
plt.xlabel("----- Value of X----- ")
plt.ylabel("----- Value of Y----- ")
```

```
# Plotting bubble chart
plt.scatter(x, y, s=z*2000, c=colors, alpha=0.5)
plt.show()
```

The above code will create bubble chart as follow:




**Example:** Here we used the ***iris*** dataset to create bubble chart.

```
# Importing libraries
import matplotlib.pyplot as plt
import plotly.express as px

# Create data from iris dataset
df = px.data.iris()

# Plotting bubble chart
plt = px.scatter(df, x="sepal_width", y="sepal_length",
color="species", size='petal_length',
hover_data=['petal_width'])
plt.show()
```

The above code will create bubble chart as follow:


## 2.8 Surface Plot

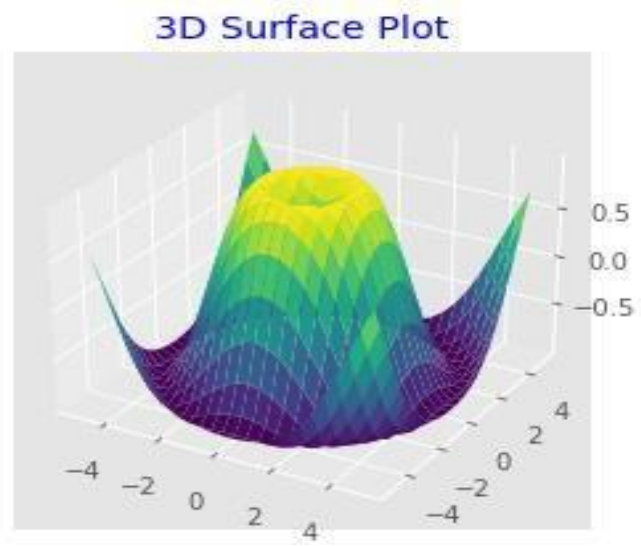
This plot is like as wireframe plot, but each face of the wireframe is filled polygon.

This plot shows the functional relationship between one dependent variable and two independent variables. The ***plot_surface()*** function is used to create a surface plot. Here we create a 3D surface plot.

**Example:** Here we draw a 3D surface plot.

```
# Importing library
import numpy as np
import matplotlib.pyplot as plt

# 3D projection
fig = plt.figure()
ax = plt.axes(projection="3d")

# Function
def func(x, y):
    return np.sin(np.sqrt(x * x + y * y))

# All three axis
x = np.linspace(-5, 5, 25)
y = np.linspace(-5, 5, 25)
X, Y = np.meshgrid(x, y)
Z = func(X, Y)

# 3D surface plotting
ax.plot_surface(X, Y, Z, rstride=1, cstride=1,
cmap='viridis', edgecolor='none')
ax.set_title("3D Surface Plot", color="blue")
plt.show()
```

The above code will plot as follow:




## 2.9 Map Chart

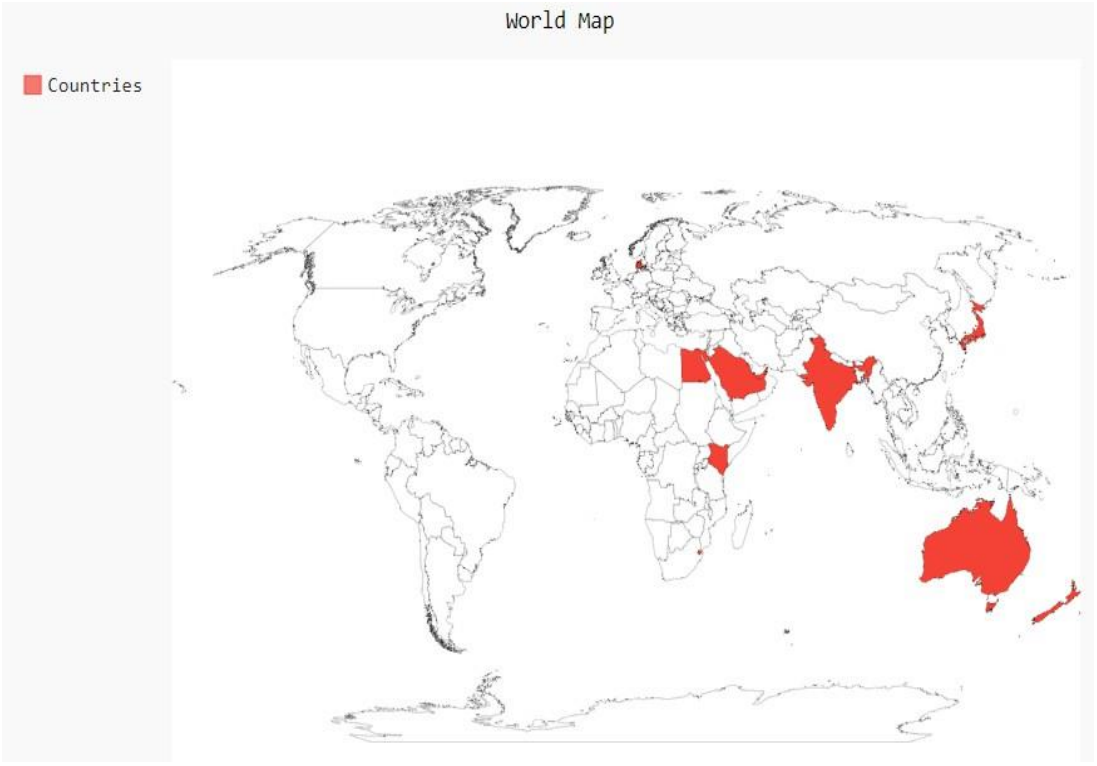
Map chart is used to represent the different types of information in a context, often geographical using one or more layers. It helps to compare the values and show the

categories across geographical regions. The map chart is mostly used to represent geographical regions in a data such as regions, states, countries etc. The map chart contains interactive shapes or display separate markers on an image or map background.

**Example:** Here we draw a world map of selected country.

```
# Importing library
import pygal

# Create map
wm = pygal.maps.world.World()

# Title of map
wm.title = 'World Map'

# List of countries
wm.add('Countries',
{'ae','au','dk','eg','in','jp','ke','nz','sa','sz'})

# Save the map
wm.render_to_file('map1.svg')
```

The above code will plot world map with highlighted selected countries as follow:

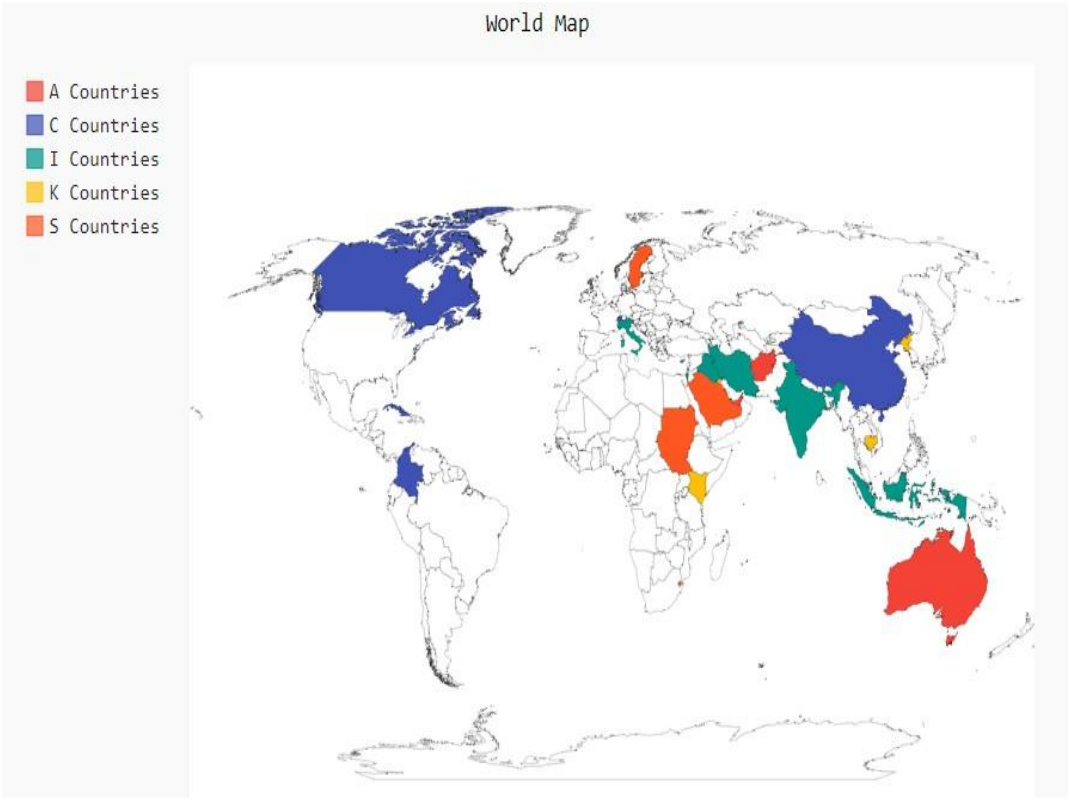



We can also set the different colors for different series of countries. Here we created a world map with selected series of countries as follows:

```
# Importing library
import pygal
```

```
# Create map
wm = pygal.maps.world.World()

# Title of map
wm.title = 'World Map'

# List of countries
wm.add('A Countries', ['ae','af','au'])
wm.add('C Countries', ['ca','ch','cn','co','cu','cy'])
wm.add('I Countries', ['id','il','in','iq','ir','it'])
wm.add('K Countries', ['ke','kh','kp','kw'])
wm.add('S Countries', ['sa','sd','se','sg','sz'])

# Save the map
wm.render_to_file('map2.svg')
```

The above code will plot world map with highlighted selected countries as follow:

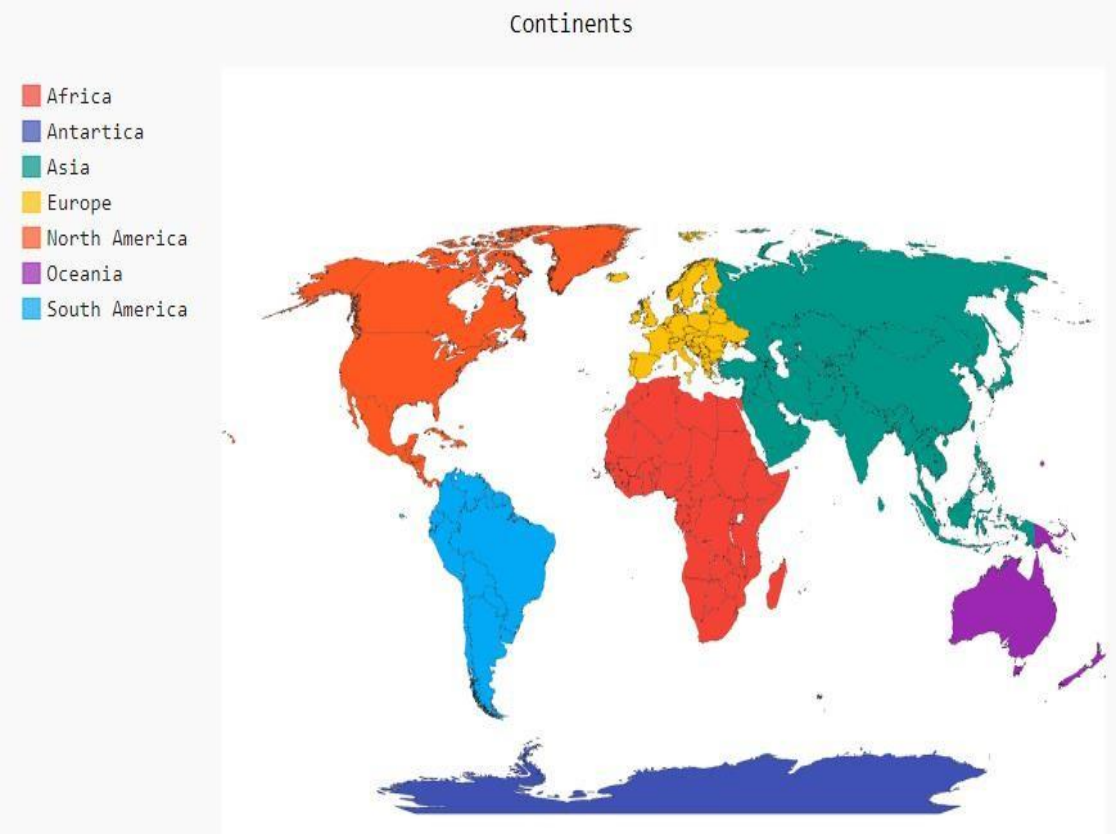



We can also display the different continents in the world. Here we created a world map with continents as follows:

```
# Importing library
import pygal

# Create map
wm = pygal.maps.world.SupranationalWorld()
```

```
# Title of map
wm.title = 'Continents'

# List of continents
wm.add('Africa', [('africa', 1)])
wm.add('Antartica', [('antartica', 1)])
wm.add('Asia', [('asia', 1)])
wm.add('Europe', [('europe', 1)])
wm.add('North America', [('north_america', 1)])
wm.add('Oceania', [('oceania', 1)])
wm.add('South America', [('south_america', 1)])

# Save the map
wm.render_to_file('map3.svg')
```

The above code will plot world map with different continents as follow:




## 2.10 Infographics

Infographics (information graphics) is a graphical visual representation of data, information or knowledge intended to represent information quickly, clearly and easy to understandable formats. It combines visual imagery, data charts and less

text together. It improves the ability of human to see the trends, patterns and relationships in a data.

The reasons to use infographics are:

- Easy to read
- Easy to share
- Easy for marketing
- Easy to summarize content

There are main two basic functions of infographics: static and interactive. The static infographics generally constructed using template. It is very simple solutions when the data and information are remains unchange. The interactive infographics is more complex in terms of designing and programming. It is continuously updating and used to engage audience to give attention and represent the more updated and advances information to the users.

The various types of designing are used for infographics. It is selected based on the types of information that you want to represent clearly. The most common types of infographics are list, comparison, geographic, statistical, informational, data visualization, timeline, interactive, etc.

## 3. SUMMARY

The students will learn many things in this module such as data mining and data visualization concepts. They will be able to create a various type of charts or plots using Python.

- Ability to do understand the data mining and data visualization concept.
- Ability to plot the various types of charts including data points, line chart, bar chart, pie chart and area chart.
- Ability to plot advance data visualization charts including candlestick chart, bubble chart, surface chart, map chart and infographics.

## 4. REFERENCES


**Books**

1. Davy Cielen, Arno D. B. Meysman, Mohamed Ali : Introducing Data Science, Manning Publications Co.
2. Stephen Klosterman (2019) : Data Science Projects with Python, Packt Publishing
3. Jake VanderPlas (2017) : Python Data Science Handbook: Essential Tools for Working with Data, O'Reilly
4. Wes McKinnery and Pandas Development Team (2021) : pandas : powerful Python data analysis toolkit, Release 1.2.3

**Web References**

1. https://www.oreilly.com
2. https://www.geeksforgeeks.org

3. https://www.tutorialspoint.com
4. https://www.w3schools.com
5. https://pandas.pydata.org
6. https://pbpython.com
7. https://matplotlib.org
8. https://www.logianalytics.com
9. https://seleritysas.com
10. https://www.semrush.com

## QUESTIONS

### Short Answer:

1. What is chart?
2. List types of charts.
3. What is scatter plot?
4. What is line chart?
5. What is candlestick chart?
6. What is surface plot?
7. What is infographics?

## Long Answer:

1. Explain bar chart with example.
2. Explain pie chart with example.
3. Explain area chart with example.
4. Explain bubble chart with example.
5. Explain map chart with example.

## PRACTICALS

1. Create and display scatter plot with different arguments.
2. Create and display bar plot with different arguments.
3. Create and display pie plot with different arguments.
4. Create and display area plot with different arguments.
5. Create and display bubble plot with different arguments.
6. Create and display map chart with different arguments.
7. Create and display candlestick plot.
8. Create and display surface plot.

# MODULE - VII

# VISUALIZING DATA PROGRAMMATICALLY

## :: TOPICS ::

- Google chart
- Basics of bar chart
- Basics of pie chart
- Working with chart animation

Dr. Chetan R. Dudhagara

Assistant Professor and Head
Department of Communication & Information Technology
International Agribusiness Management Institute
Anand Agricultural University
Anand, Gujarat, India

## OBJECTIVES

The main objective of this module is to understand the data visualization concepts programmatically. The various data visualization techniques and charts such as google charts, bar chart and pie chart are covered in this module.

## 5. INTRODUCTION

Data science become a buzzword that everyone talks about the data science. Data science is an interdisciplinary field that combines different domain expertise, computer programming skills, mathematics and statistical knowledge to find or extract the meaningful or unknown patterns from unstructured and structure dataset. Data science is useful for extraction, preparation, analysis and visualization of various information. Various scientific methods can be applied to get insight in data.

Data visualization is a graphical representation of data and quantitative information by using various graphical tools such as graphs and charts. It is more useful to understand the trends, patterns and outliers in the data.

## 6. DATA VISULIZATION

Charts is the representation of data in a graphical form. It helps to summarizing and presenting a large amount of data in a simple and easy to understandable formats. By placing the data in a visual context, we can easily detect or identify the patterns, trends and correlations among them.

Python provides various easy to use multiple graphics libraries for data visualization. These libraries are work with both small and large datasets.

Python has multiple graphics libraries with different features. Some of the most popular and commonly used Python data visualization libraries are Matplotlib, Pandas, Seaborn, Plotly and ggplot.

Google is also providing very powerful tool for visualization which is called google chart. It is very simple, user-friendly and free tool for creating interactive and animated charts.

Google chart API (Application Programming Interface) is an extremely easy and simple tool to create a chart from the data and embed it into a webpage. It has numerous functions and properties for creating efficient and interactive charts. It has good cross-browser compatibility and also supports legacy browsers well.

## 7. STARTING WITH GOOGLE CHARTS

Google chart is a very popular library available for charting tool. Initially it was developed and used for its internal applications for rendering charts. Later it was released for the public use also. It helps and used to create variety of charts such as bar chart, pie chart, bubble chart, geo chart, etc.

Google charts is a combination of APIs:

- Google Chart API
- Google Visualization API

- The google chart API is used to create static visualizations from the data and embeds it into a webpage. Basic HTML programming knowledge is recommended to create chart using google chart API. The various types of charts including scatter, line, bar, pie, etc. can be created using it and embedded into webpages.
- The google visualization API is used to creates dynamic visualizations which allow to user interaction within the webpage. The charts are created using java scripting language. The various types of charts including timelines, heat maps, tree maps, etc. can be created using it and embedded into webpages.

## 3.1 Google Charts API

Google charts is a pure JavaScript based charting library to enhance web application by adding interactive charting capability. It provides variety of chart such as bar chart, pie chart, line chart, spline chart, area chart, etc.

There are two ways to use google charts

- Using downloaded google chart
- Using content delivery network (CDN)

**Using downloaded google chart:**

- Include the google charts JavaScript file in the HTML page using following script.
- **Syntax:**

```
<head>
        <script src = "googlecharts/loader.js/" >
</head>
```

**Using content delivery network:**

- Include the google charts JavaScript file in the HTML page using following script.
- **Syntax:**

```
<head>
        <script src = "https://www.gstatic.com/charts/loader.js/" >
</head>
```

## 3.2 Bar Chart

A bar chart or bar graph that presents categorical data using rectangular bars or columns with different heights or lengths proportional to the value that they represent. The bars can be plotted horizontally or vertically. The vertical bar chart is called as column chart also.

To create google bar chart in web page, the following link is added.

```
<script
src = "https://www.gstatic.com/charts/loader.js">
</script>
```

The  tag or element is used to display the chart in a web page.

```
<div id="myChart" style="width:700px; height:400px">
</div>
```

The  tag or element must have a unique name or id.

To load the google graph API, the ***corechart*** package is use. The callback function is use to call when the API is loaded.

```
google.charts.load('current',{packages:['corechart']});
google.charts.setOnLoadCallback(drawChart);
```

**Example:** To create a bar chart using google chart of the following data.

| **Class** | Distinction | First Class | Second Class | Pass Class | Fail |
|---|---|---|---|---|---|
| **No. of Student** | 12 | 28 | 41 | 14 | 5 |

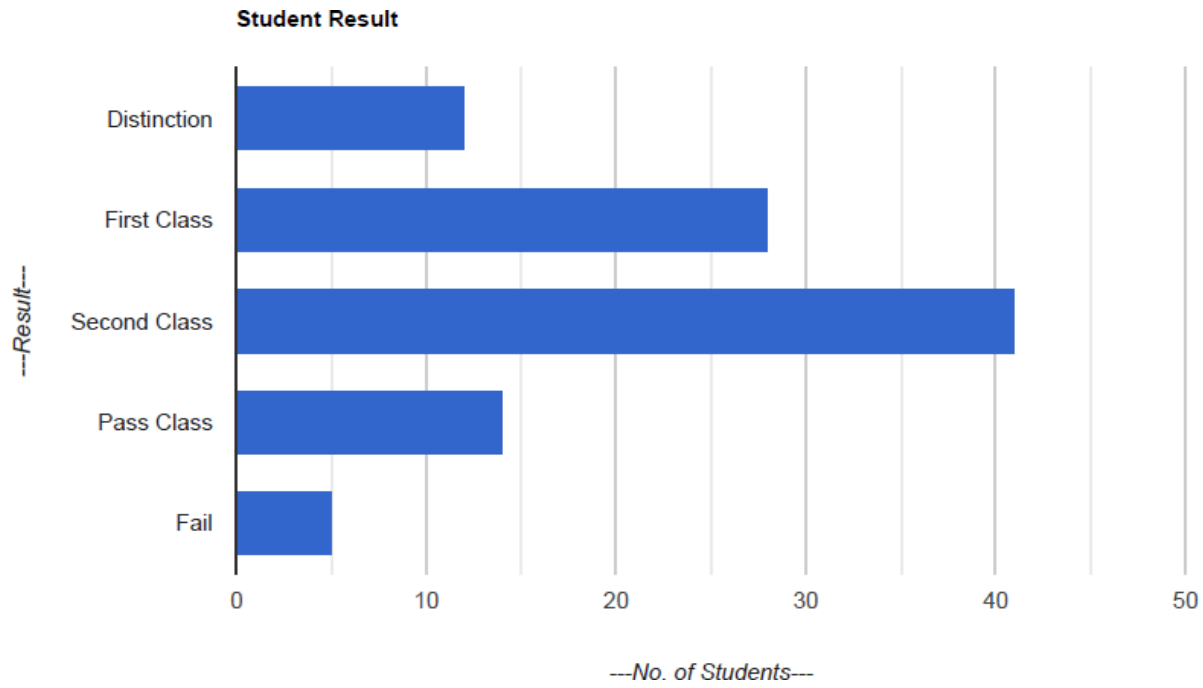
The above data contains results of 100 students.

The following code will create a bar chart using JavaScript.

```
<html>
  <head>
    <script type="text/javascript"
     src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
     google.charts.load('current', {packages:['corechart']});
     google.charts.setOnLoadCallback(drawChart);
     function drawChart()
     {
         var data = google.visualization.arrayToDataTable
         ([
           ['Result', 'No. of Students'],
           ['Distinction', 12],
           ['First Class', 28],
           ['Second Class', 41],
           ['Pass Class', 14],
           ['Fail', 5]
         ]);
         var options =
```

```
        {
            title: 'Student Result',
            hAxis: {title: '---No. of Students---'},
            vAxis: {title: '---Result---'}
        };
        var chart = new
        google.visualization.BarChart(document.getElementById
        ('barchart'));
        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
    <div id="barchart" style="width: 900px; height: 500px;"></div>
  </body>
</html>
```

The above code will create bar chart as follow:




**Example:** To create a grouped bar chart using google chart of the following data.

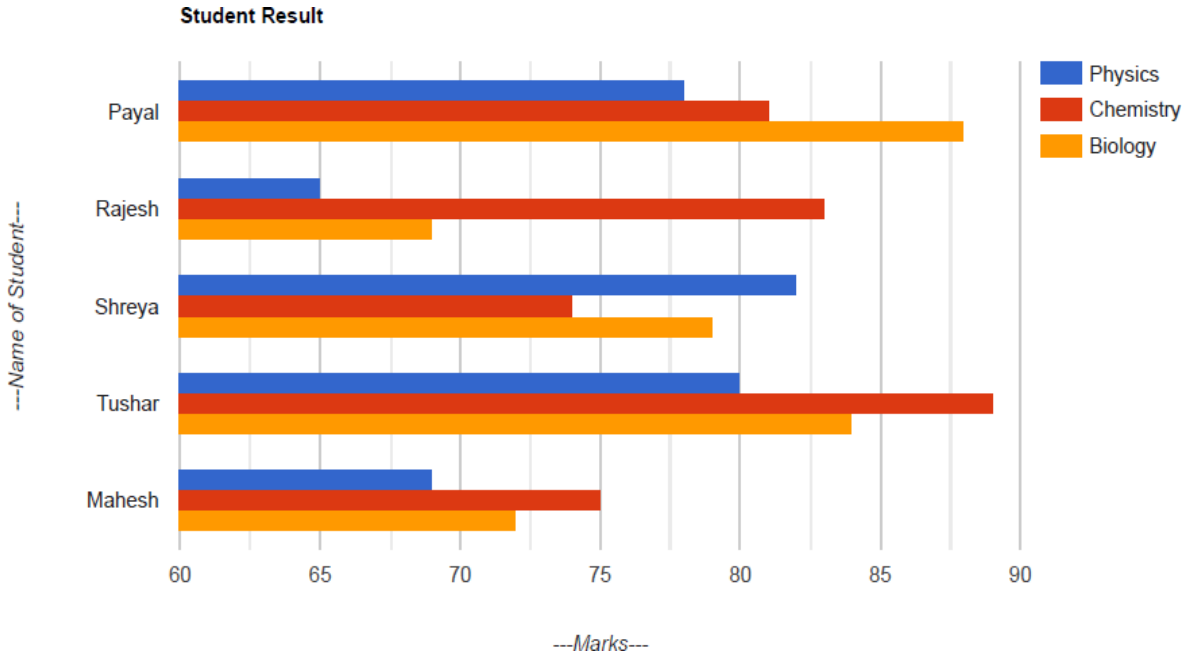
| **Name of Student** | Payal | Rajesh | Shreya | Tushar | Mahesh |
|---|---|---|---|---|---|
| **Physics** | 78 | 65 | 82 | 80 | 69 |
| **Chemistry** | 81 | 83 | 74 | 89 | 75 |
| **Biology** | 88 | 69 | 79 | 84 | 72 |

The above data contains results of three subjects such as physics, chemistry and biology score of 5 students.

The following code will create a grouped bar chart using JavaScript.

```
<html>
  <head>
    <script type="text/javascript"
     src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
     google.charts.load('current', {packages:['corechart']});
     google.charts.setOnLoadCallback(drawChart);
     function drawChart()
     {
        var data = google.visualization.arrayToDataTable
        ([
          ['Name', 'Physics', 'Chemistry', 'Biology'],
          ['Payal', 78, 81, 88],
          ['Rajesh', 65, 83, 69],
          ['Shreya', 82, 74, 79],
          ['Tushar', 80, 89, 84],
          ['Mahesh', 69, 75, 72]
        ]);
        var options =
        {
            title: 'Student Result'
            hAxis: {title: '---Marks---'},
            vAxis: {title: '---Name of Student---'}
        };
        var chart = new
        google.visualization.BarChart(document.getElementById
        ('barchart'));
        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
    <div id="barchart" style="width: 900px; height: 500px;"></div>
  </body>
</html>
```

The above code will create grouped bar chart as follow:




### 3.3 Pie Chart

A pie chart is a type of graph which display the data in a circular graph which is also divided into slices. The slices of pie chart represent the relative size or quantity of data. It requires a list of categorical values and corresponding numerical values. The term pie represents a whole and a slice represent the portions or parts of a whole.

**Example:** To create a pie chart using google chart of the following data.

| Name of Browser | Market Share (in %) |
|---|---|
| Chrome | 64.67 |
| Safari | 19.06 |
| Edge | 3.99 |
| Firefox | 3.66 |
| Samsung Internet | 2.81 |
| Opera | 2.36 |
| UC Browser | 0.97 |
| Android | 0.57 |
| IE | 0.5 |
| Other | 1.41 |

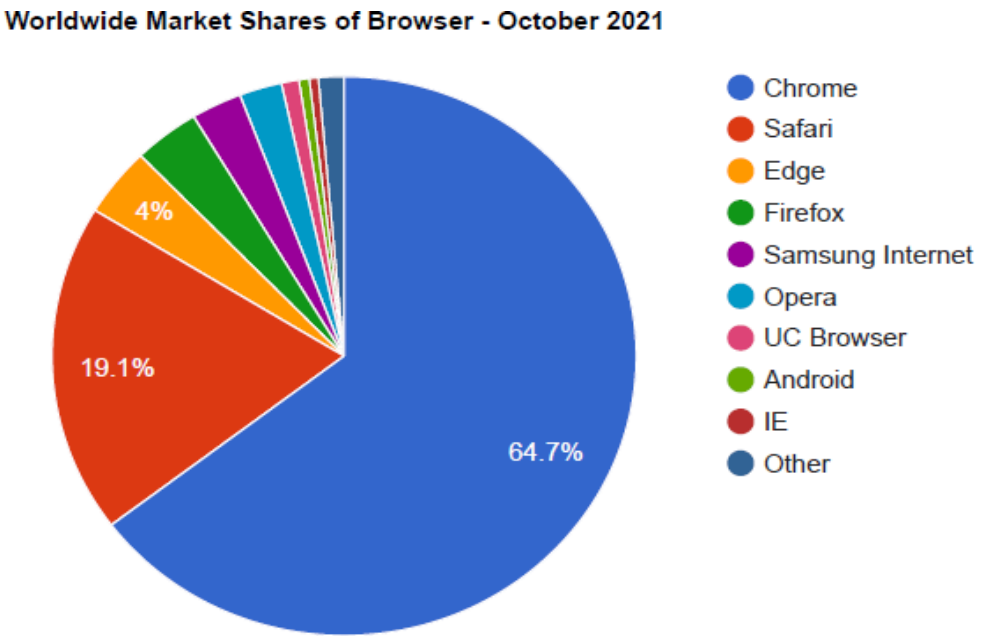
The above data contains worldwide market shares of browsers in October 2021

(Source: https://gs.statcounter.com/browser-market-share)

The following code will create a pie chart using JavaScript.

```
<html>
  <head>
    <script type="text/javascript"
     src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
     google.charts.load('current', {packages:['corechart']});
     google.charts.setOnLoadCallback(drawChart);
     function drawChart()
     {
        var data = google.visualization.arrayToDataTable
        ([
          ['Browser', 'Market Share'],
          ['Chrome', 64.67],
          ['Safari', 19.06],
          ['Edge', 3.99],
          ['Firefox', 3.66],
          ['Samsung Internet', 2.81],
          ['Opera', 2.36],
          ['UC Browser', 0.97],
          ['Android', 0.57],
          ['IE', 0.5],
          ['Other', 1.41]
        ]);
        var options =
        {
          title: 'Worldwide Market Shares of Browser - October 2021'
        };
        var chart = new google.visualization.PieChart
          (document.getElementById('piechart'));
        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
    <div id="piechart" style="width: 900px; height: 500px;"></div>
  </body>
</html>
```

The above code will create pie chart as follow:




**Example:** To create a donut chart using google chart of the following data.

| **Activity** | Sleep | School | Study | Eat | Sports | Entertainment | Refresh |
|---|---|---|---|---|---|---|---|
| **No. of Hour** | 8 | 6 | 2 | 2 | 2.5 | 2 | 1.5 |

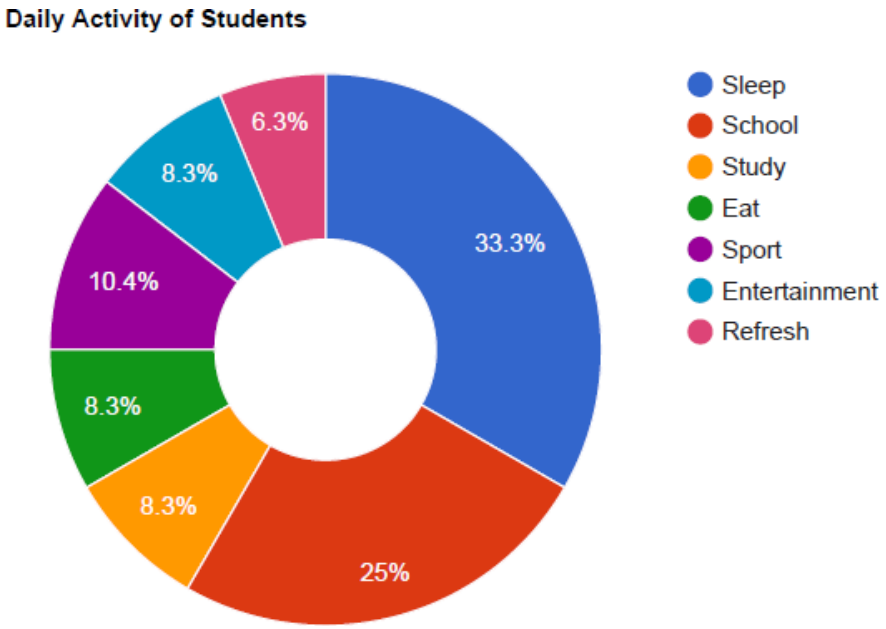
The above data contains numbers of hour spent in daily activities by the students.

The following code will create a donut chart using JavaScript.

```
<html>
  <head>
    <script type="text/javascript"
     src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
     google.charts.load('current', {packages:['corechart']});
     google.charts.setOnLoadCallback(drawChart);
     function drawChart()
     {
        var data = google.visualization.arrayToDataTable
        ([
          ['Activity', 'No. of Hours'],
          ['Sleep', 8],
          ['School', 6],
          ['Study', 2],
          ['Eat', 2],
          ['Sport', 2.5],
          ['Entertainment', 2],
          ['Refresh', 1.5],
        ]);
        var options =
        {
          title: 'Daily Activity of Students', pieHole: 0.4
        };

        var chart = new google.visualization.PieChart
```

```
          (document.getElementById('piechart'));
        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
    <div id="piechart" style="width: 900px; height: 500px;"></div>
  </body>
</html>
```

The above code will create donut chart as follow:




**Example:** To create a 3D pie chart using google chart of the following data.

| **Activity** | Sleep | School | Study | Eat | Sports | Entertainment | Refresh |
|---|---|---|---|---|---|---|---|
| **No. of Hour** | 8 | 6 | 2 | 2 | 2.5 | 2 | 1.5 |

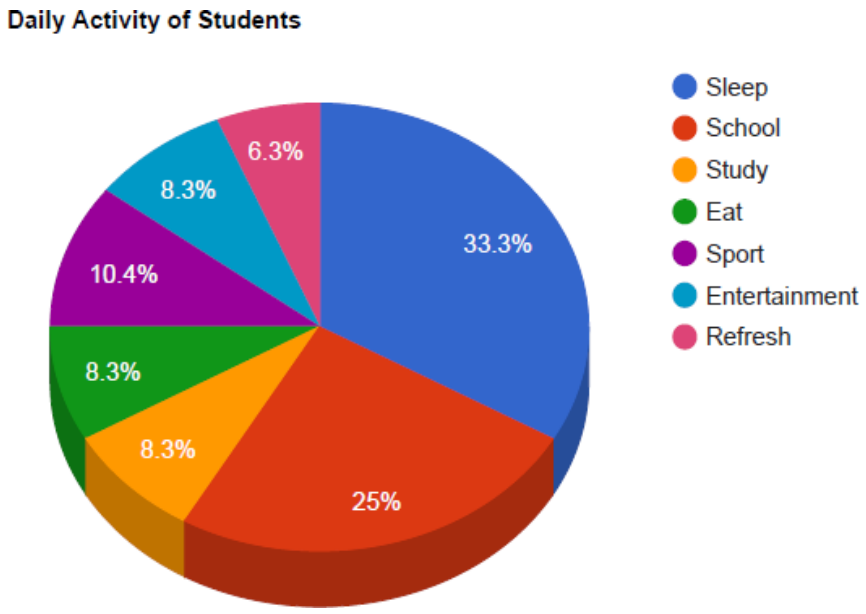
The above data contains numbers of hour spent in daily activities by the students.

The following code will create a 3D pie chart using JavaScript.

```
<html>
  <head>
    <script type="text/javascript"
     src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
     google.charts.load('current', {packages:['corechart']});
     google.charts.setOnLoadCallback(drawChart);
     function drawChart()
     {
        var data = google.visualization.arrayToDataTable
        ([
          ['Activity', 'No. of Hours'],
          ['Sleep', 8],
```

```
          ['School', 6],
          ['Study', 2],
          ['Eat', 2],
          ['Sport', 2.5],
          ['Entertainment', 2],
          ['Refresh', 1.5],
        ]);
        var options =
        {
          title: 'Daily Activity of Students', is3D:true
        };
        var chart = new google.visualization.PieChart
          (document.getElementById('piechart'));
        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
    <div id="piechart" style="width: 900px; height: 500px;"></div>
  </body>
</html>
```

The above code will create 3D pie chart as follow:




## 8. WORKING WITH CHART ANIMATIONS

The charts which represents data in a dynamic and animated way for better understanding is called animation chart. The google charts can animated smoothly in two ways, either on startup when you draw the chart at first time or when you redraw the chart after making changes on data or operations. The animated chart displays several chart states one after the other. The animation chart required a little additional efforts and knowledge to create dynamic changes and movements in a chart.

**Example:** To create an animated bar chart using google chart of the following data.

| **Product** | Computer | Printer | Tablet | Mobile | Web Camera | Head Phone |
|---|---|---|---|---|---|---|
| **No. of Item Sales** | 12 | 8 | 22 | 40 | 25 | 48 |

The above data contains sales of different computer related items.

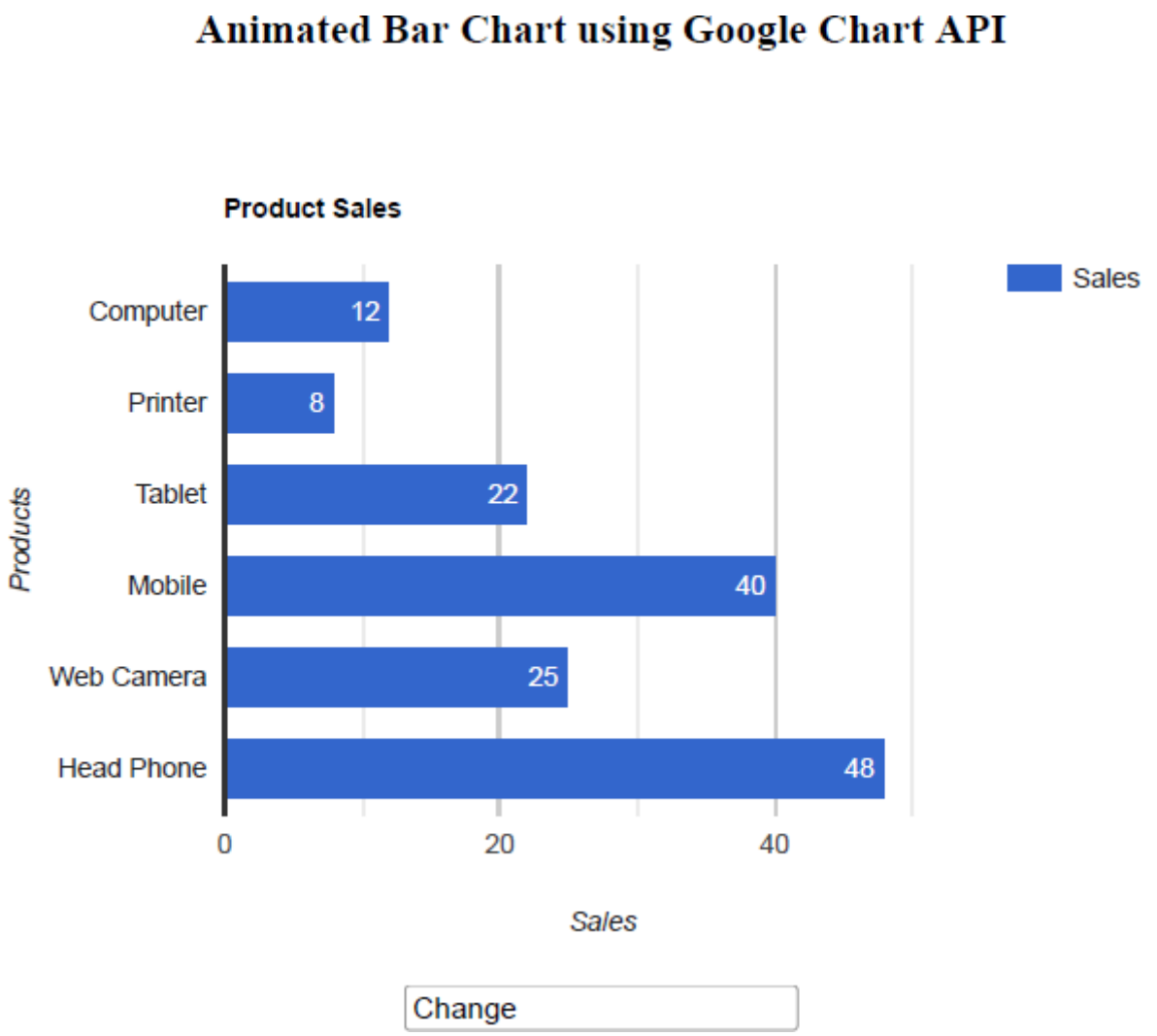
The following code will create an animated bar chart using JavaScript.

```
<!DOCTYPE html>
<html>
<head>
<center><h3> Animated Bar Chart using Google Chart API </h3>
<script type="text/javascript" src="https://www.google.com/jsapi">
</script>
   <script type="text/javascript">
   google.load('visualization', '1', {packages:['corechart']});
   </script>
   <script type="text/javascript">
   function renderChart()
   {
      var data = google.visualization.arrayToDataTable
      ([
           ['Product', 'Sales', { role: 'annotation' } ],
           ['Computer', 0, "12"],
           ['Printer', 0, "8"],
           ['Tablet', 0, "22"],
           ['Mobile', 0, "40"],
           ['Web Camera', 0, "25"],
           ['Head Phone', 0, "48"]
      ]);
      var options =
      {
           title : "Product Sales",
           hAxis: { title: "Sales", viewWindow: { min: 0, max: 55 } },
           vAxis: { title: "Products" },
           animation: {duration: 1000, easing: 'inAndOut',}
      };
      var button = document.getElementById('changeData');
      var initialAnimationPlayed = false;
      var chart = new google.visualization.BarChart(
           document.getElementById("chart"));
      google.visualization.events.addListener(chart, 'ready',
      function()
      {
        if (!initialAnimationPlayed)
        {
           initialAnimationPlayed = true;
           data.setValue(0, 1, 12);
           data.setValue(1, 1, 8);
           data.setValue(2, 1, 22);
           data.setValue(3, 1, 40);
           data.setValue(4, 1, 25);
```

```
          data.setValue(5, 1, 48);
          chart.draw(data, options);
        }
        });
        chart.draw(data, options);
        var firstData = true;
        button.onclick = function ()
        {
          if (!firstData)
          {
            firstData = !firstData;
            data.setValue(0, 1, 12);
            data.setValue(1, 1, 8);
            data.setValue(2, 1, 22);
            data.setValue(3, 1, 40);
            data.setValue(4, 1, 25);
            data.setValue(5, 1, 48);
            data.setValue(0, 2, "12");
            data.setValue(1, 2, "8");
            data.setValue(2, 2, "22");
            data.setValue(3, 2, "40");
            data.setValue(4, 2, "25");
            data.setValue(5, 2, "48");
          }
          else
          {
            firstData = !firstData;
            data.setValue(0, 1, 40);
            data.setValue(1, 1, 25);
            data.setValue(2, 1, 15);
            data.setValue(3, 1, 30);
            data.setValue(4, 1, 10);
            data.setValue(5, 1, 18);
            data.setValue(0, 2, "40");
            data.setValue(1, 2, "25");
            data.setValue(2, 2, "15");
            data.setValue(3, 2, "30");
            data.setValue(4, 2, "10");
            data.setValue(5, 2, "18");
          }
          chart.draw(data, options);
        };
    }
    google.setOnLoadCallback(renderChart);
</script>
<div id="chart" style="width:550px; height:400px; margin: 0 auto">
</div>
</head>
<body>
<div>
       <input id="changeData" style="Button" value="Change">
</div>
</body>
```

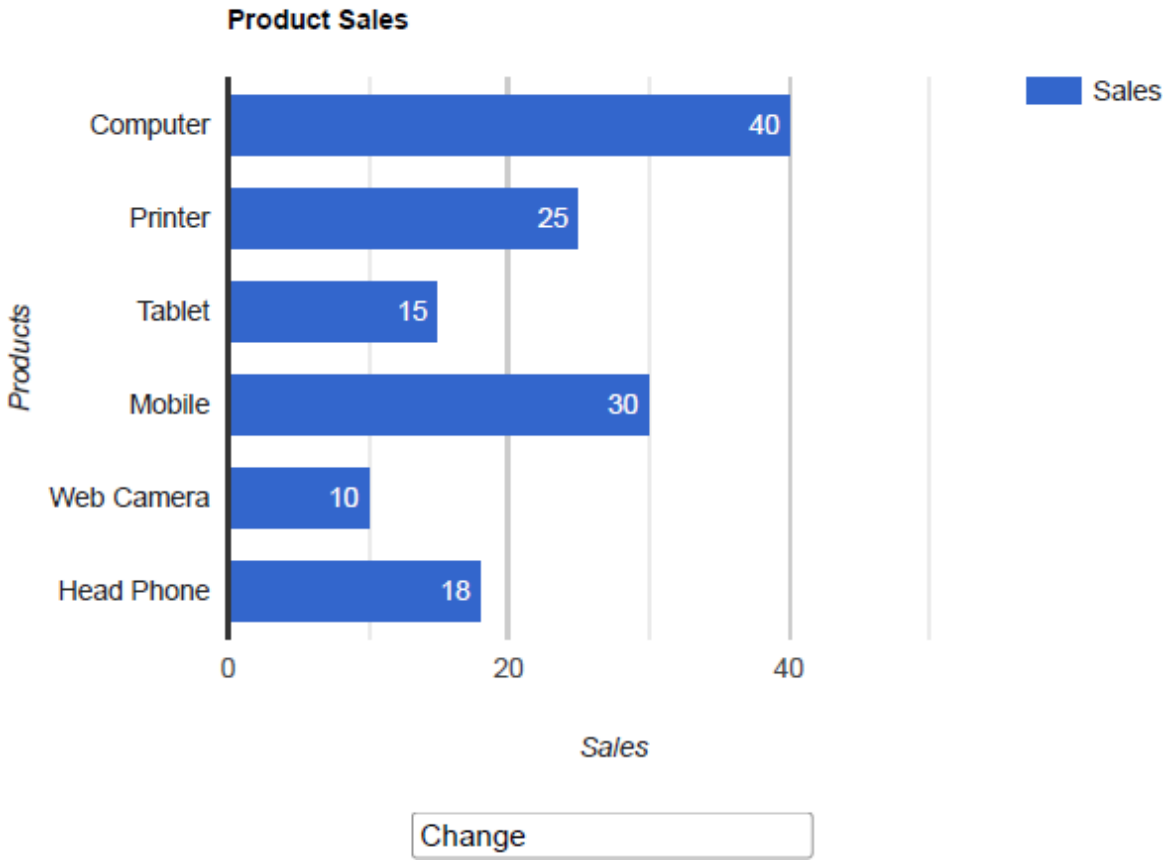
```
</html>
```

The above code will create animated bar chart as follow:




After the click on change button the bar chart is animated as follows:

## Animated Bar Chart using Google Chart API




## 9. SUMMARY

The students will learn data visualization concepts using google chart API in JavaScript. They will be able to create a various type of charts or plots using google chart API.

- Ability to do understand the data visualization concept.
- Ability to plot the various types of charts including bar chart and pie chart with animation using google chart.

# 10. REFERENCES


**Books**

1. Jon J. Raasch, Graham Murray, Vadim Ogievetsky, Joseph Lowery (2015) : JavaScript® and jQuery® for Data Analysis and Visualization, Published by John Wiley & Sons, Inc.
2. Stephen Klosterman (2019) : Data Science Projects with Python, Packt Publishing.
3. Jake VanderPlas (2017) : Python Data Science Handbook: Essential Tools for Working with Data, O'Reilly.

**Web References**

11. https://developers.google.com/chart/
12. https://www.encodedna.com/google-chart/create-interactive-graphs-using-google-chart.htm
13. https://www.tutorialspoint.com/googlecharts/googlecharts_pie_basic.htm
14. https://old.dataone.org/software-tools/google-charts
15. https://codeactually.com/googlecharts.html
16. https://www.w3schools.com/whatis/whatis_google_charts.asp
17. https://en.wikipedia.org/wiki/Usage_share_of_web_browsers
18. https://gs.statcounter.com/browser-market-share
19. https://canvasjs.com/javascript-charts/animated-chart/

## QUESTIONS

### Short Answer:

8. What is chart?
9. List types of charts.
10. What is google chart API?
11. What is content delivery network?
12. What is bar char?
13. What is pie chart?

### Long Answer:

6. Explain google chart in detail.
7. Explain bar chart using google chart with example.
8. Explain pie chart using google chart with example.
9. Explain animated chart using google chart with example.

## PRACTICALS

9. Create bar chart using google chart.
10. Create pie chart using google chart.
11. Create animated chart using google chart.