# UNIT I

## Introduction to Data Analytics:

Data and its importance, data analytic and its types, importance of data analytics

## Python Fundamentals:

Python Language Basics, Jupyter Notebook, Introduction to pandas, Data Structures, Essential Functionality

## Central Tendency and Dispersion:

Visual Representation of the Data, Measures of Central Tendency, Dispersion

## Reference Books:


1. McKinney, W. (2012). Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. " O'Reilly Media, Inc.".
2. Swaroop, C. H. (2003). A Byte of Python. Python Tutorial.
3. Ken Black, sixth Editing. Business Statistics for Contemporary Decision Making. “John Wiley & Sons, Inc”.
4. Anderson Sweeney Williams (2011). Statistics for Business and Economics. “Cengage Learning”.

## Course Outcomes

- Use of  statistical tools and techniques in analyzing the different dimensions of data
- Knowing different functions and packages in Python for data interpretation
- Getting hands on experience in model building using data tools
- Calculating the estimate of variation using ANOVA methods
- Getting out different classifiers with Precision and recall methods

## Data Analytics

- Data is getting generated at a massive rate, by the minute.
- Organizations, on the other hand, are trying to explore every opportunity to make sense of this data.
- This is where Data analytics has become crucial in running a business successfully.
- It is commonly used in companies to drive profit and business growth.
- **What is Data analytics** is the process of exploring and analyzing large datasets to make predictions and boost data-driven decision making.
- Data analytics allows us to **collect, clean, and transform data** to **derive meaningful insights**.
- It helps to answer questions, test hypotheses, or challenge theories.

**Data analytics refers to the process of examining, cleaning, transforming, and modeling data to extract useful information, draw conclusions, and support decision-making.**

**It involves the use of various techniques, tools, and technologies to analyze and interpret large sets of data, uncover patterns, trends, and insights, and make informed business or research decisions.**

## Applications of Data Analytics

❖ Data analytics is used in most sectors of businesses. Here are some primary areas where data analytics does its magic:

- Data analytics is used in the banking and e-commerce industries to detect **fraudulent transactions**.
- The healthcare sector uses data analytics to improve patient health by detecting diseases before they happen. It is commonly used for **cancer detection**.
- Data analytics finds its usage in inventory management to keep track of different items.
- Logistics companies use data analytics to ensure faster delivery of products by optimizing vehicle routes.
- Marketing professionals use analytics to reach out to the right customers and perform targeted marketing to increase ROI (Return on Investment).
- Data analytics can be used for city planning, to build smart cities.
- **The applications of data analytics continue to expand as technology advances and organizations recognize the value of leveraging data for informed decision-making.**
- Data analytics finds applications across a wide range of industries and sectors due to its ability to derive meaningful insights from large volumes of data.

- Here are some key applications of data analytics:
- **Business Intelligence (BI):**
- Organizations use data analytics to gain insights into their operations, customer behavior, and market trends. BI tools help in visualizing data and making informed business decisions.
- **Financial Analysis:**
- In finance, data analytics is employed for fraud detection, risk management, portfolio optimization, and predicting market trends. It helps financial institutions make data-driven decisions to maximize returns and minimize risks.
- **Healthcare Analytics:**
- In healthcare, data analytics is used for patient care improvement, resource optimization, fraud detection, and predicting disease outbreaks. It aids in personalized medicine and contributes to medical research.
- **Marketing and Customer Analytics:**
- Businesses use data analytics to analyze customer behavior, preferences, and trends. This information helps in targeted marketing campaigns, customer segmentation, and improving overall customer experience.

- **Supply Chain and Logistics:**
- Data analytics optimizes supply chain processes by predicting demand, managing inventory efficiently, and improving overall logistics and distribution.
- **Human Resources (HR) Analytics:**
- HR departments use analytics for talent acquisition, employee performance analysis, workforce planning, and employee retention strategies.
- **Social Media Analytics:**
- Companies analyze social media data to understand customer sentiment, track brand mentions, and measure the effectiveness of marketing campaigns. This information is valuable for shaping marketing strategies and brand management.
- **Cybersecurity:**
- Data analytics helps in identifying and preventing security threats by analyzing patterns and anomalies in network data. It enhances the ability to detect and respond to cyberattacks in real-time.
- **Education Analytics:**
- Educational institutions use data analytics for student performance analysis, personalized learning, and predicting dropout rates. It aids in optimizing educational programs and resources.

- **Smart Cities:**
- Cities use data analytics to improve urban planning, traffic management, energy consumption, and public services. It contributes to creating more efficient and sustainable urban environments
- **Sports Analytics:**
- Sports teams utilize data analytics for player performance analysis, injury prevention, and strategic decision-making. It enhances coaching strategies and team management.
- **Manufacturing and Quality Control:**
- Data analytics is applied in manufacturing for quality control, predictive maintenance, and process optimization. It helps identify inefficiencies and improve overall production processes.

## Why is Data Analytics important?

- Data Analytics has a key role in improving your business as it is used to gather hidden insights, generate reports, perform market analysis, and improve business requirements.
- Data analytics is important because it helps businesses optimize their performances. Implementing it into the business model means companies can help reduce costs by identifying more efficient ways of doing business and by storing large amounts of data
- Data analysis helps businesses acquire relevant, accurate information, suitable for developing future marketing strategies, business plans, and realigning the company's vision or mission.
- Data analytics is important to understand trends and patterns from the massive amounts of data that are being collected. It helps optimize business performance, forecast future results, understand audiences, and reduce costs.
- Data analytics is the process of examining, transforming, and interpreting data to uncover meaningful patterns, insights, and trends. It is important because it enables organizations to make data-driven decisions, improve operational efficiency, identify opportunities, and gain a competitive edge.

- **Data analytics gives a company a cost-effective solution**. Even it will help them to improvise the decisions making process. The sectors like manufacturing, media, healthcare, real estate, and others require data analytics techniques and tools
- **The scope of data analytics** science involves extracting insights and patterns from data to drive decision-making and gain a competitive advantage. Data Analytics is a powerful discipline that transforms raw data into actionable insights, driving informed decision-making and fostering innovation across industries.
- Data analytics future growth is expected to continue to the increasing amount of data being generated, the growing importance of data-driven decision-making in businesses, and the continued development of big data and AI technologies
- **Data analytics and data analytics bootcamp is a huge industry and is predicted to keep growing. It is expected to touch US$11.87 billion by 2026 as it keeps growing at a steady pace. This industry will disrupt the market, causing a great shift in it and bringing several job opportunities with it.**

## What is the role of Data Analytics?

- **Data analytics help a business optimize its performance, perform more efficiently, maximize profit, or make more strategically-guided decisions. The techniques and processes of data analytics have been automated into mechanical processes and algorithms that work over raw data for human consumption.**
- Data analytics is the science of analyzing raw data to make conclusions about information, helping businesses optimize their performance, make more efficient decisions, and maximize profits. The role of data analytics is to provide insights and support decision-making processes in various industries. Key aspects of data analytics include.
- **Gathering and cleaning data:** Data analysts collect data through various methods, such as conducting surveys, tracking visitor characteristics on a company website, or buying data sets from data collection specialists. They also clean the data to maintain its quality by removing duplicates, errors, or outliers

- **Interpreting data:** Analysts find patterns or trends in data that can help answer specific problems or questions. They use statistical analysis, graphical modeling, and other techniques to extract valuable insights from the data.

- **Data visualization:** Data analysts create visualizations like charts and graphs to represent data in an easy-to-understand format. This helps stakeholders, including company leadership, understand the importance of the insights.

- **Communicating findings:** Analysts present their findings to interested parties, such as executives and managers, using reports and presentations. They also use various tools to make their work more accurate and efficient during data analysis

- **Data analytics is important because it helps businesses optimize their performance, improve efficiency, and make more strategically-guided decisions. It is used in various industries, including banking and finance, where it is used to predict market trends and assess risk. Data analytics also plays a crucial role in detecting and preventing fraud to improve efficiency and reduce risk for organizations**

❖ **Some common tools used in data analytics include:**

❖ **Jupyter Notebook:** A web-based interactive computing environment that allows users to create and share documents that contain live code, equations, visualizations, and narrative text.

❖ **Apache Spark:** A fast and general-purpose cluster-computing system that can process large data sets with high speed and efficiency.

❖ **Google Cloud AutoML:** A suite of machine learning products that allows users to build, train, and deploy custom machine learning models without requiring expertise in machine learning.

❖ **Statistical Analysis System (SAS) :** A software suite used for advanced analytics, multivariate analysis, business intelligence, data management, and predictive analytics.

❖ **Microsoft Power BI:** A business analytics tool that provides interactive visualizations and business intelligence capabilities with an interface simple enough for end-users to create their own reports and dashboards

- **Tableau:** A data visualization and business intelligence software that helps users create interactive, web-based dashboards and reports.

- **KNIME: Konstanz Information Miner:** An open-source data analytics platform that provides a user-friendly interface for data preprocessing, machine learning, and statistical analysis.

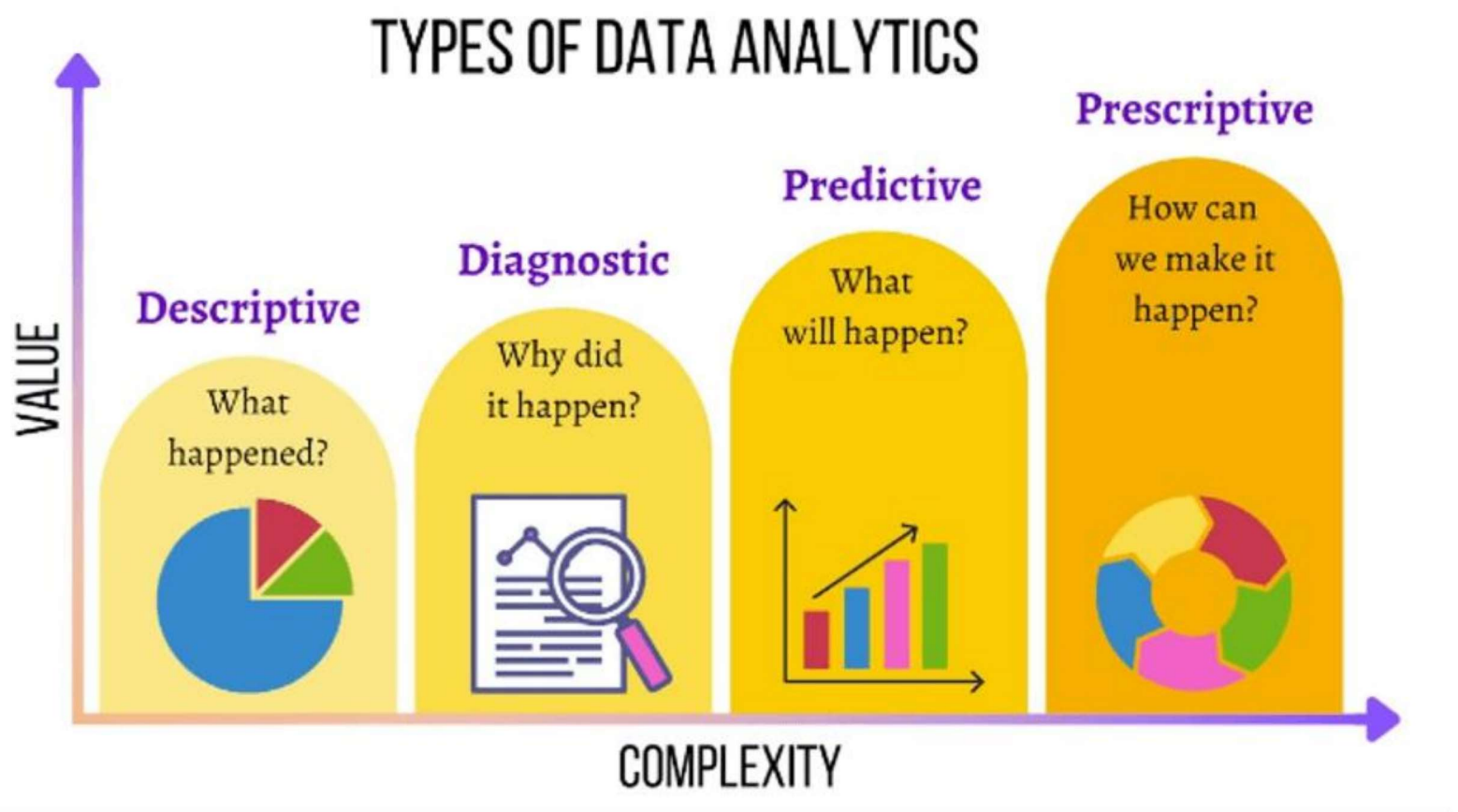
- **Steamlit:** A web-based data analytics platform that allows users to create interactive, web-based dashboards and reports.

- **R Programming:** A leading analytics tool in the industry, widely used for statistics and data modeling. It can easily manipulate data and present it in different ways.

- **Python:** A versatile programming language that can handle various data analyses and integrate with third-party packages for machine learning and data visualization.

## ❖ DIFFERENT TYPES OF DATA ANALYSIS

❖ The four main types of data analysis: Descriptive, Diagnostic, Predictive, and Prescriptive



- **Descriptive Analytics**
- **Descriptive analytics is a simple, surface-level type of analysis that looks at what has happened in the past.**
- The two main techniques used in descriptive analytics are **data aggregation and data mining—so**, the data analyst **first gathers the data and presents** it in a summarized format (that's the aggregation part) and then "**mines" the data to discover patterns**.
- The data is then presented in a way that can be easily understood by a wide audience (not just data experts).
- It's important to note that descriptive analytics doesn't try to explain the historical data or establish cause-and-effect relationships; at this stage, **it's simply a case of determining and describing the "what".**
- Descriptive analytics draws on descriptive statistics, which you can learn about here.
- **Data aggregation is the process of summarizing a large pool of data for high level analysis. At its most basic level, it involves compiling information from a range of prescribed databases and organizing it into a simpler, easy-to-use medium, usually utilizing sum, average, mean, or median references**

## ❖ Diagnostic Analytics

❖ **While descriptive analytics looks at the "what", diagnostic analytics explores the "why".**

❖ When running diagnostic analytics, data analysts will first seek to identify anomalies within the data—that is, anything that cannot be explained by the data in front of them.

❖ **For example:**

❖ If the data shows that there was a sudden drop in sales for the month of March, the data analyst will need to investigate the cause.

❖ To do this, they'll get on what's known as the discovery phase, identifying any additional data sources that might tell them more about why such irregularities happened.

❖ Finally, the data analyst will try to uncover causal relationships—for example, looking at any events that may correlate or correspond with the decrease in sales. At this stage, data analysts may use probability theory, regression analysis, filtering, and time-series data analytics.

## ❖ Predictive Analytics

❖ **Predictive analytics tries to predict what is likely to happen in the future**.

❖ This is where data analysts start to come up with actionable, data-driven insights that the company can use to inform their next steps.

❖ Predictive analytics estimates the likelihood of a future outcome based on historical data and probability theory, and while it can never be completely accurate, it does eliminate much of the guesswork from key business decisions.

❖ Predictive analytics can be used to forecast all sorts of outcomes—from what products will be most popular at a certain time, to how much the company revenue is likely to increase or decrease in a given period.

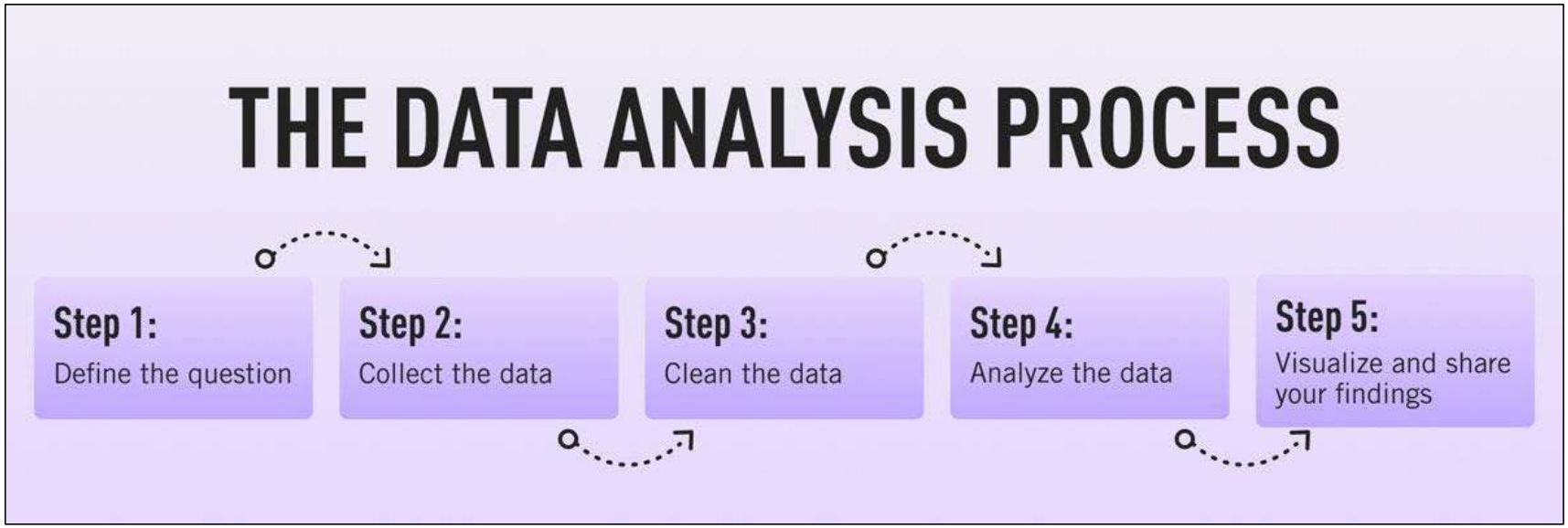
❖ Ultimately, predictive analytics is used to increase the business's chances of "hitting the mark" and taking the most appropriate action.

## ❖ Prescriptive Analytics

- ❖ Prescriptive analytics advises on the actions and decisions that should be taken.
- ❖ In other words, prescriptive analytics shows you how you can take advantage of the outcomes that have been predicted.
- ❖ When conducting prescriptive analysis, data analysts will consider a range of possible scenarios and assess the different actions the company might take.
- ❖ Prescriptive analytics is one of the more complex types of analysis, and may involve working with algorithms, machine learning, and computational modeling procedures.
- ❖ However, the effective use of prescriptive analytics can have a huge impact on the company's decision-making process and, ultimately, on the bottom line.

**Explain the step-by-step process that a data analyst typically follows in their workflow.**

- Like any scientific discipline, data analysis follows a rigorous step-by-step process. Each stage requires different skills and know-how.
- To get meaningful insights, though, it's important to understand the process as a whole. An underlying framework is invaluable for producing results that stand up to scrutiny.



**Step 1: Define the question(s) you want to answer**

- In the first step of process the data analyst is given a problem/business task. The analyst has to understand the task and the stakeholder's expectations for the solution.
- A stakeholder is a person that has invested their money and resources to a project. The analyst must be able to ask **different questions in order to find the right solution to their problem**.
- The analyst has to find the root cause of the problem in order to fully understand the problem.
- The analyst must make sure that he/she doesn't have any distractions while analyzing the problem.
- Communicate effectively with the stakeholders and other colleagues to completely understand what the underlying problem is.
- At this stage, you'll take a clearly defined problem and come up with a relevant question or hypothesis you can test.
- **Questions to ask yourself for the Ask phase are:**
  - What are the problems that are being mentioned by my stakeholders?
  - What are their expectations for the solutions?

**Step 2: Collect the data**

- The second step is to Prepare or Collect the Data. This step includes collecting data and storing it for further analysis.
- The analyst has to collect the data based on the task given from multiple sources. The data has to be collected from various sources, internal or external sources.
- Internal data is the data available in the organization that you work for while external data is the data available in sources other than your organization.
- The data that is collected by an individual from their own resources is called first-party data.
- The data that is collected and sold is called second-party data.
- Data that is collected from outside sources is called third-party data. The common sources from where the data is collected are Interviews, Surveys, Feedback, Questionnaires. The collected data can be stored in a spreadsheet or SQL database.
- Data analysts will usually gather structured data from primary or internal sources, such as CRM software or email marketing tools.

**Step 3: Clean the data**

- The third step is Clean and Process Data. After the data is collected from multiple sources, it is time to clean the data.
- Clean data means data that is free from **misspellings, redundancies**, and **irrelevance**.
- Clean data largely depends on data integrity. There might be duplicate data or the data might not be in a format, therefore the unnecessary data is removed and cleaned.
- There are different functions provided by SQL and Excel to clean the data. **This is one of the most important steps in Data Analysis as clean and formatted data helps in finding trends and solutions**.
- The most important part of the Process phase is to check whether your data is biased or not. Bias is an act of favoring a particular group/community while ignoring the rest. Biasing is a big no-no as it might affect the overall data analysis. The data analyst must make sure to include every group while the data is being collected.
- Your original dataset may contain duplicates, anomalies, or missing data which could distort how the data is interpreted, so these all need to be removed.
- Data cleaning can be a time-consuming task, but it's crucial for obtaining accurate results.
- Ex. Field extraction algorithm, Clustering .

**Step 4: Analyze the data**

- The fourth step is to Analyze. The cleaned data is used for analyzing and identifying trends.
- It also performs calculations and combines data for better results. The tools used for performing calculations are Excel or SQL.
- These tools provide in-built functions to perform calculations or sample code is written in SQL to perform calculations.
- Using Excel, we can create pivot tables and perform calculations while SQL creates temporary tables to perform calculations.
- Programming languages are another way of solving problems. They make it much easier to solve problems by providing packages. The most widely used programming languages for data analysis are R and Python.
- Some common techniques include **regression analysis, cluster analysis, and time-series analysis.**
- **This step in the process also ties in with the four different types of analysis we looked at in (Descriptive, Diagnostic, Predictive, and Prescriptive).**

**Step 5: Interpret and share the results**

- ❖ This final step in the process is where data is transformed into valuable business insights.
- ❖ Depending on the type of analysis conducted, you'll present your findings in a way that others can understand—in the form of a chart or graph.
- ❖ The data now transformed has to be made into a visual (chart, graph).
- ❖ The reason for making data visualizations is that there might be people, mostly stakeholders that are non-technical.
- ❖ Visualizations are made for a simple understanding of complex data. **Tableau** and **Looker** are the two popular tools used for compelling data visualizations.
- ❖ **Tableau** is a simple drag and drop tool that helps in creating compelling visualizations.
- ❖ **Looker is a data viz tool** that directly connects to the database and creates visualizations.
- ❖ Tableau and Looker are both equally used by data analysts for creating a visualization.
- ❖ R and Python have some packages that provide beautiful data visualizations.
- ❖ R has a package named **ggplot** which has a variety of data visualizations.
- ❖ A presentation is given based on the data findings.

- Sharing the insights with the team members and stakeholders will help in making better decisions.
- It helps in making more informed decisions and it leads to better outcomes.
- Presenting the data involves transforming raw information into a format that is easily comprehensible and meaningful for various stakeholders.
- This process includes the creation of visual representations, such as charts, graphs, and tables, to effectively communicate patterns, trends, and insights collected from the data analysis.
- The goal is to facilitate a clear understanding of complex information, making it accessible to both technical and non-technical audiences.
- Effective data presentation involves thoughtful selection of visualization techniques based on the nature of the data and the specific message planned.
- It goes beyond mere display to storytelling, where the presenter interprets the findings, highlights key points, and guides the audience through the narrative that the data unfolds.
- Whether through reports, presentations, or interactive dashboards, the art of presenting data involves balancing simplicity with depth, ensuring that the audience can easily grip the significance of the information presented and use it for informed decision-making.

## Why Data Analytics Using Python?

- There are many programming languages available, but Python is popularly used by statisticians, engineers, and scientists to perform data analytics.

Here are some of the reasons why Data Analytics using Python has become popular:

- Python is easy to learn and understand and has a simple syntax.
- The programming language is scalable and flexible.
- It has a vast collection of libraries for numerical computation and data manipulation.
- Python provides libraries for graphics and data visualization to build plots.
- It has broad community support to help solve many kinds of queries.

**Python Libraries for Data Analytics**

- One of the main reasons why Data Analytics using Python has become the most preferred and popular mode of data analysis is that it provides a range of libraries.
- **NumPy:** NumPy supports n-dimensional arrays and provides numerical computing tools. It is useful for Linear algebra and Fourier transform.
- **Pandas:** Pandas provides functions to handle missing data, perform mathematical operations, and manipulate the data.
- **Matplotlib:** Matplotlib library is commonly used for plotting data points and creating interactive visualizations of the data.
- **SciPy:** SciPy library is used for scientific computing. It contains modules for optimization, linear algebra, integration, interpolation, special functions, signal and image processing.
- **Scikit-Learn:** Scikit-Learn library has features that allow you to build regression, classification, and clustering models.

# Visual Representation of the Data, Measures of Central Tendency, Dispersion

## Visual Representation of the Data

### What is Data Visualization?

- Data visualization is a field in data analysis that deals with visual representation of data. It graphically plots data and is an effective way to communicate inferences from data.

- Using data visualization, we can get a visual summary of our data. With pictures, maps and graphs, the human mind has an easier time processing and understanding any given data.
- Data visualization plays a significant role in the representation of both small and large data sets, but it is especially useful when we have large data sets, in which it is impossible to see all of our data, let alone process and understand it manually.

- 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.
  - **Matplotlib**
  - **Seaborn**
  - **Bokeh**
  - **Plotly**
- We will discuss these libraries one by one and will plot some most commonly used graphs

**Data Set Used**

```
import pandas as pd
# reading the database
data = pd.read_csv("tips.csv")
# printing the top 10 rows
display(data.head(10))
```

**Matplotlib**

❖ 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.

- To install this type the below command in the terminal.
- pip install matplotlib
- After installing Matplotlib, let's see the most commonly used plots using this library.

**1. 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.

Example:

- import pandas as pd
- import matplotlib.pyplot as plt

**reading the database**

- data = pd.read_csv("tips.csv")

**Scatter plot with day against tip**

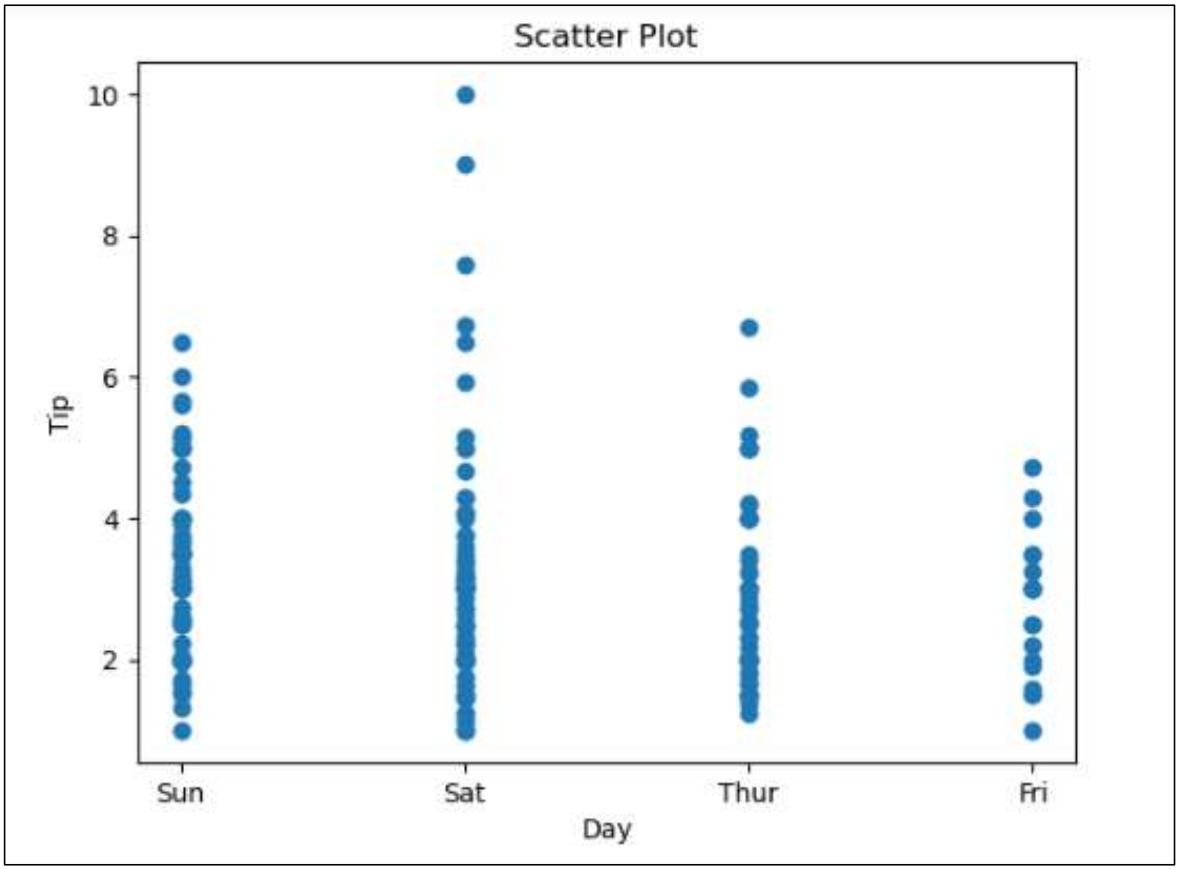
- plt.scatter(data['day'], data['tip'])

**Adding Title to the Plot**

- plt.title("Scatter Plot")

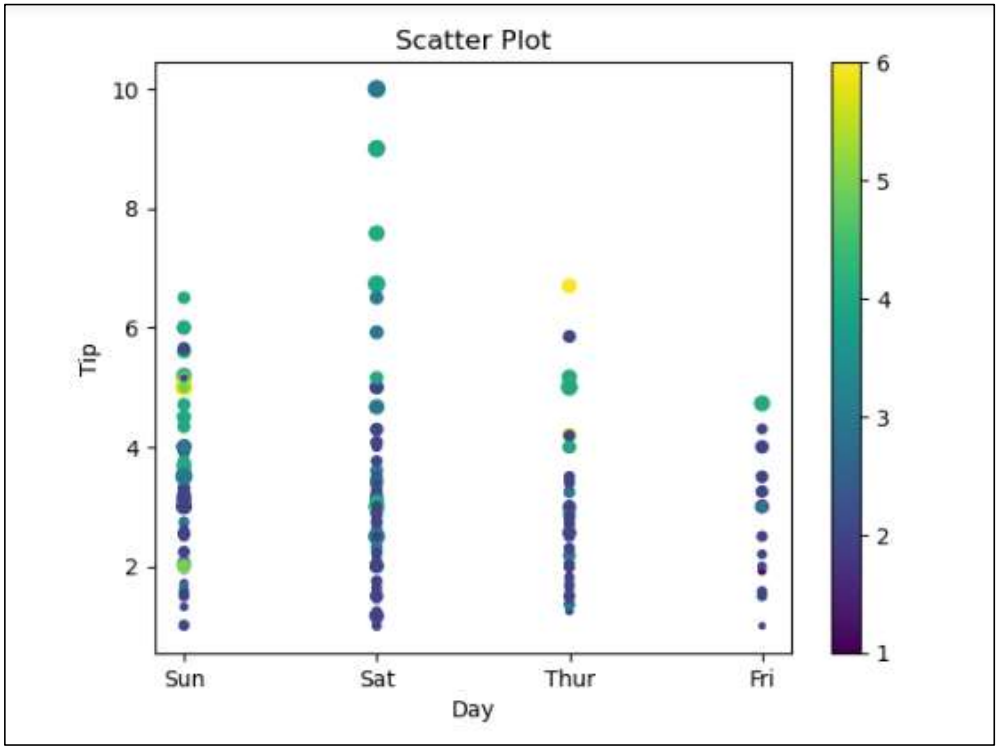
**Setting the X and Y labels**

- plt.xlabel('Day')
- plt.ylabel('Tip')
- 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.

```
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()
```



## 2. 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.

**Example:**

- import pandas as pd
- import matplotlib.pyplot as plt

**# reading the database**

- data = pd.read_csv("tips.csv")

**# Scatter plot with day against tip**

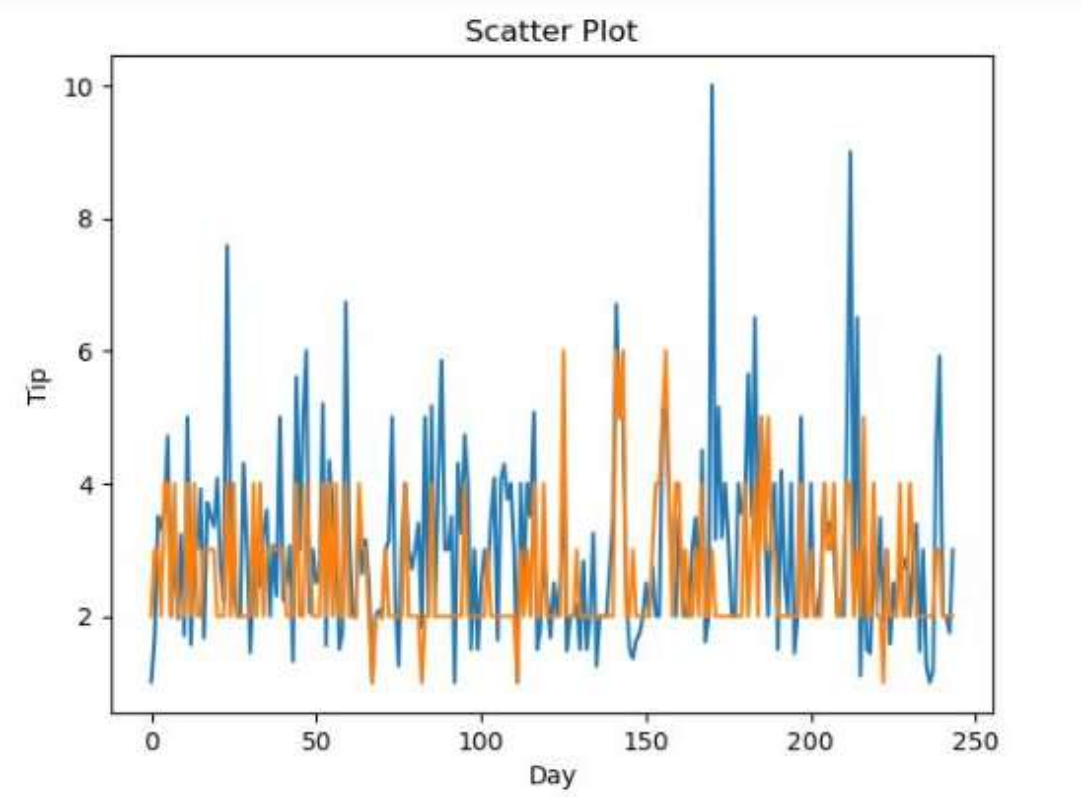
- plt.plot(data['tip'])
- plt.plot(data['size'])

**# Adding Title to the Plot**

- plt.title("Scatter Plot")

**# Setting the X and Y labels**

- plt.xlabel('Day')
- plt.ylabel('Tip')
- plt.show()



### 3. Bar Chart

❖ 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.

**Example:**

- import pandas as pd
- import matplotlib.pyplot as plt

**# reading the database**

- data = pd.read_csv("tips.csv")

**# Bar chart with day against tip**

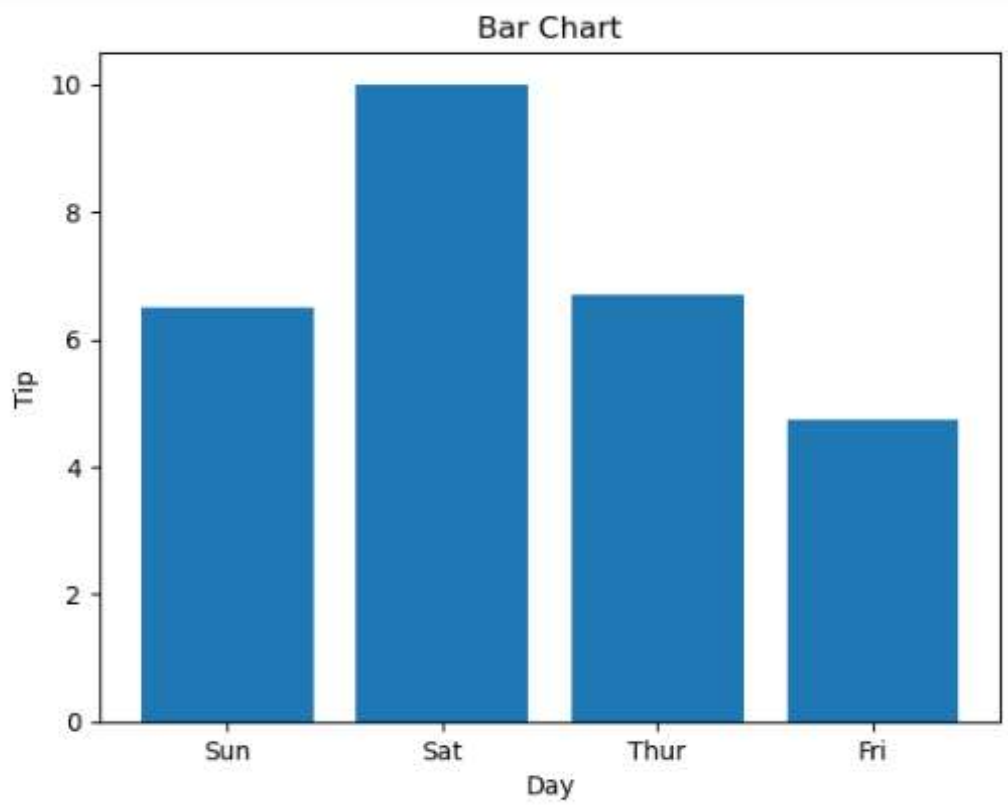
- plt.bar(data['day'], data['tip'])
- plt.title("Bar Chart")

**# Setting the X and Y labels**

- plt.xlabel('Day')
- plt.ylabel('Tip')

**# Adding the legends**

- plt.show()



## 4. 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.

**Example:**

- import pandas as pd
- import matplotlib.pyplot as plt

**# reading the database**

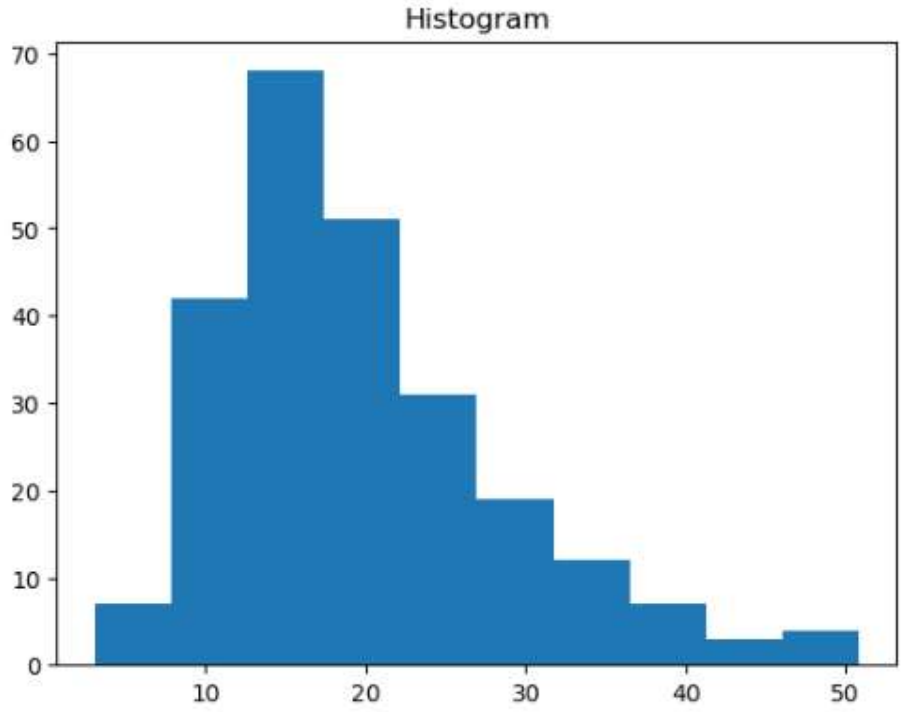
- data = pd.read_csv("tips.csv")

**# 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.
- To install seaborn type the below command in the terminal.

  pip install seaborn
- 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.
- **Note:** Seaborn comes loaded with dataset such as tips, iris, etc. but for the sake of this tutorial we will use Pandas for loading these datasets.

**# importing packages**

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

**# reading the database**

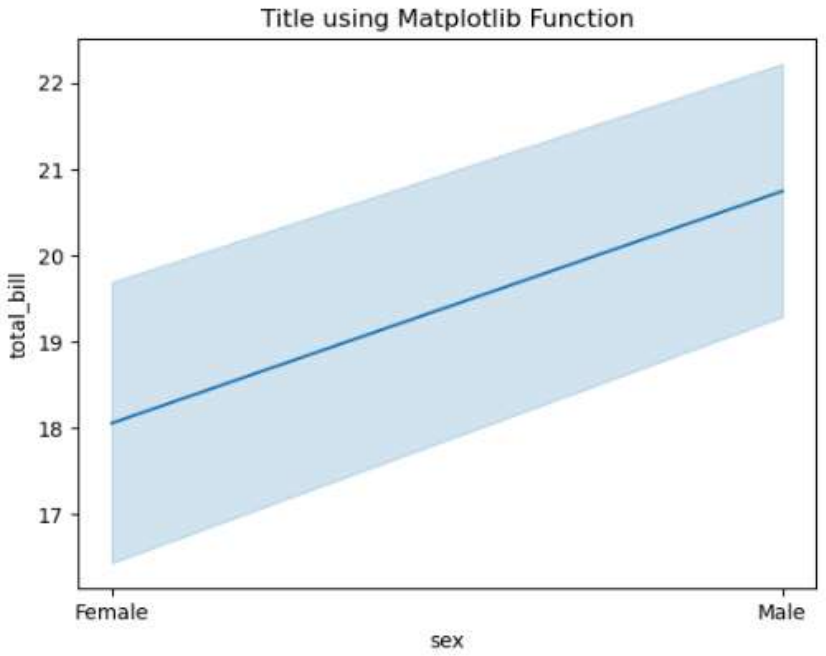
- data = pd.read_csv("tips.csv")

**# draw lineplot**

- sns.lineplot(x="sex", y="total_bill", data=data)

**# setting the title using Matplotlib**

- plt.title('Title using Matplotlib Function')
- plt.show()



## 1. Scatter Plot

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

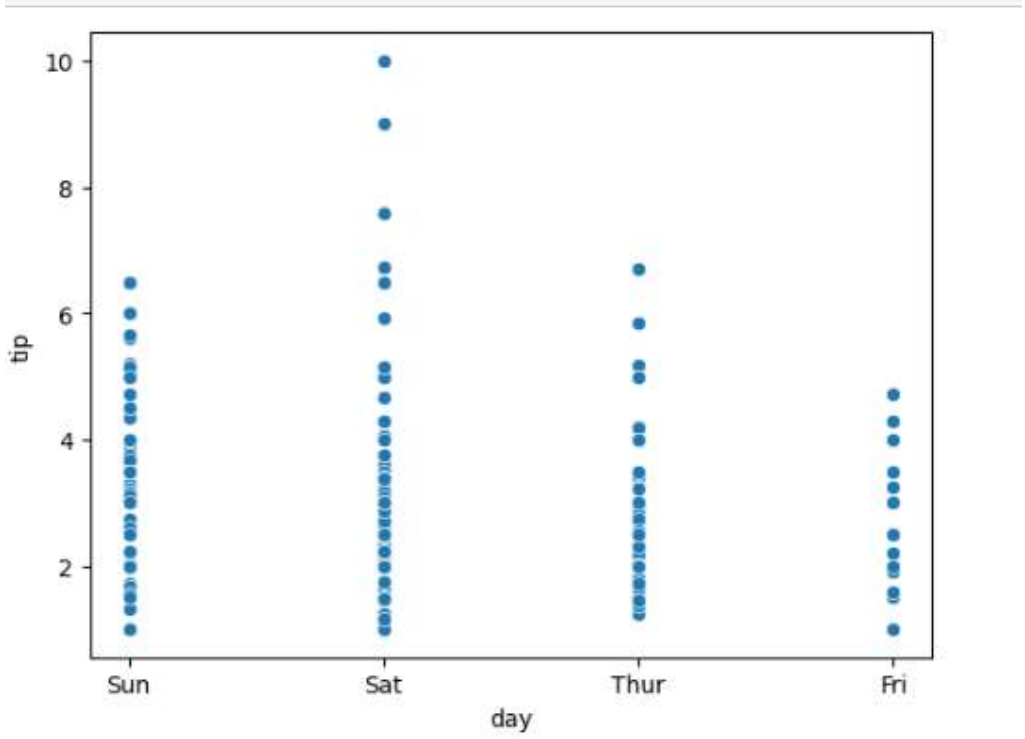
**Example:**

**# 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.scatterplot(x='day', y='tip', data=data,)
- plt.show()



❖ To 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.

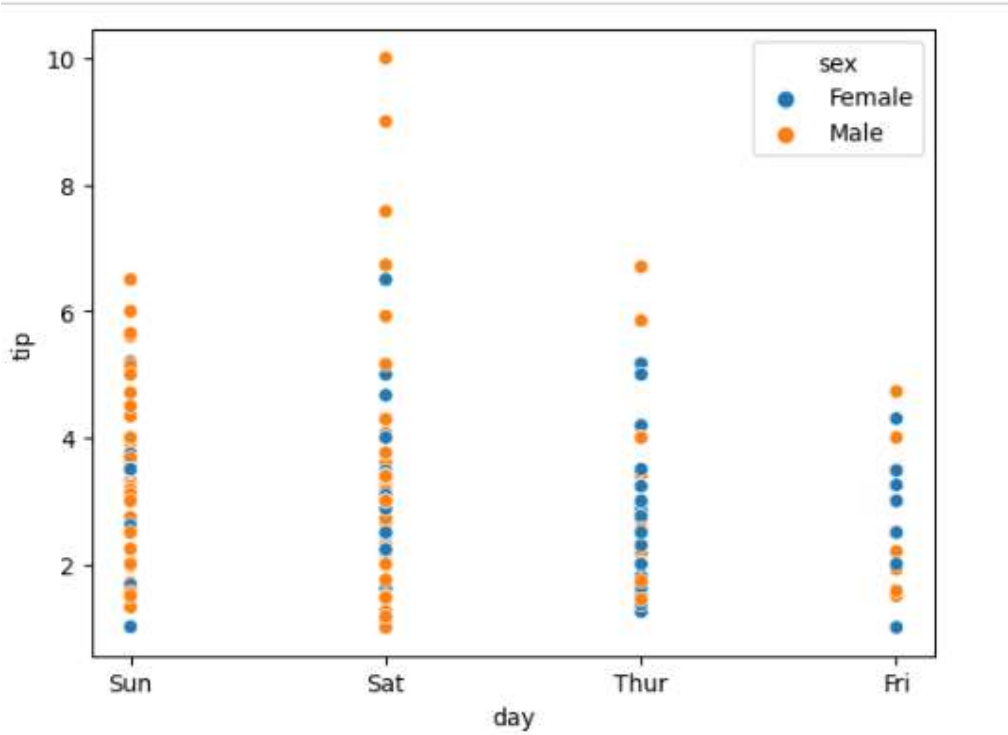
**Example:**

**# 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.scatterplot(x='day', y='tip', data=data,
- hue='sex')
- plt.show()



## 2. Line Plot

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

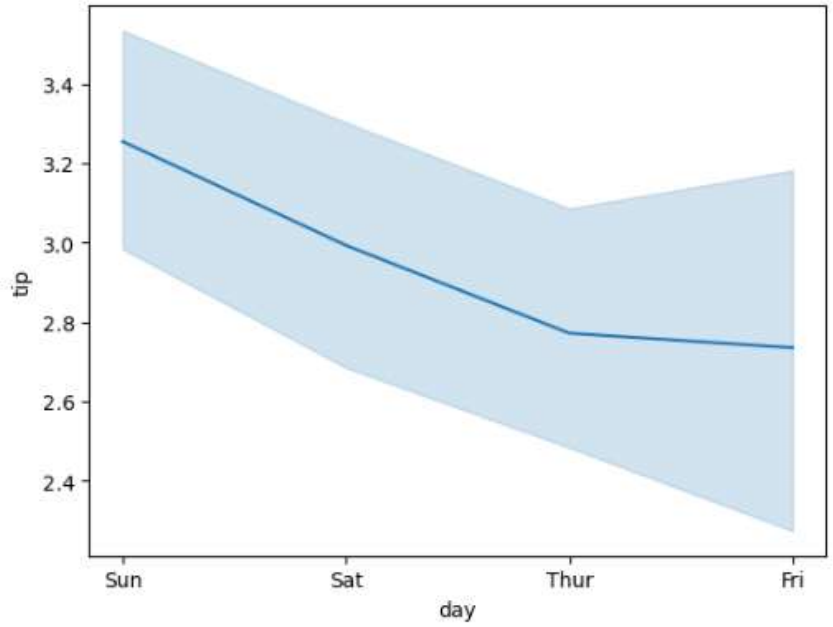
Example:

**# 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.lineplot(x='day', y='tip', data=data)
- plt.show()



**# importing packages**

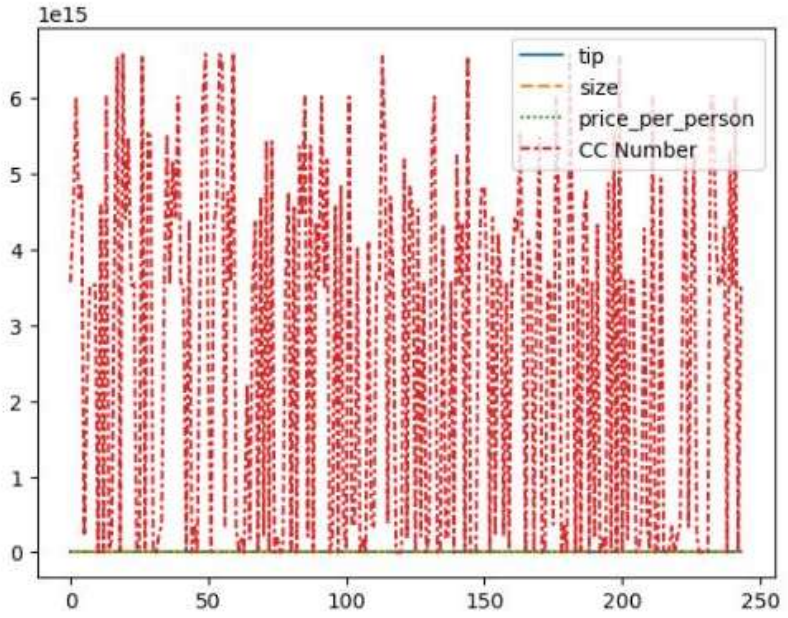
- import seaborn as sns
- import matplotlib.pyplot as plt
- import pandas as pd

**# reading the database**

- data = pd.read_csv("tips.csv")

**# using only data attribute**

- sns.lineplot(data=data.drop(['total_bill'], axis=1))
- plt.show()



### 3. Bar Plot

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

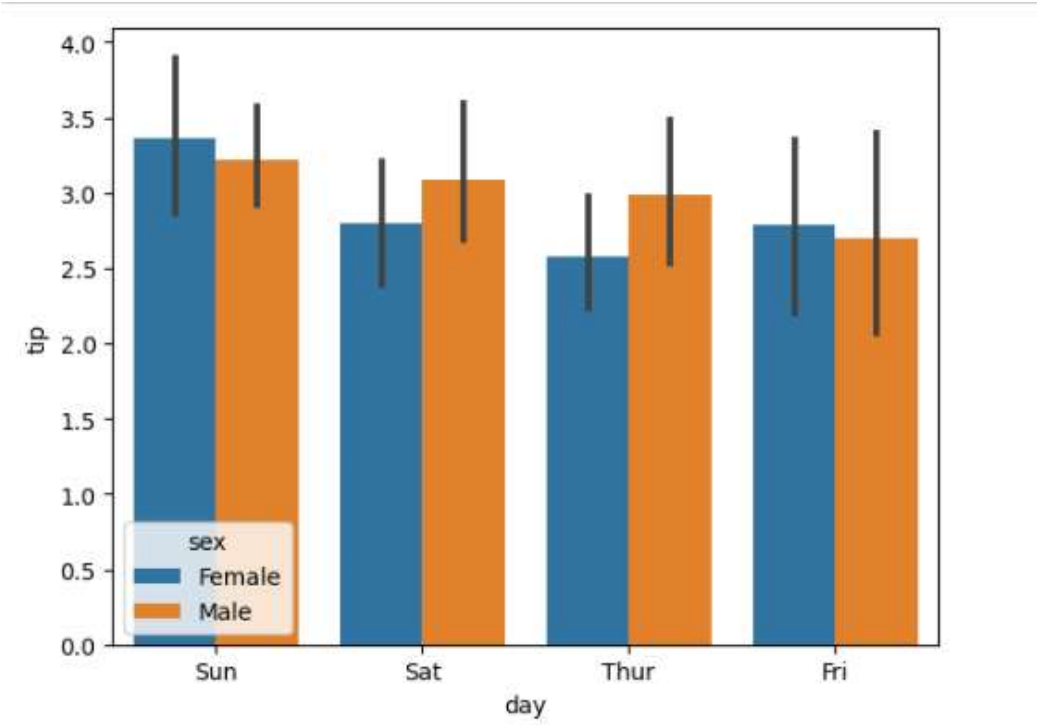
**Example:**

**# 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()



## 4. Histogram

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

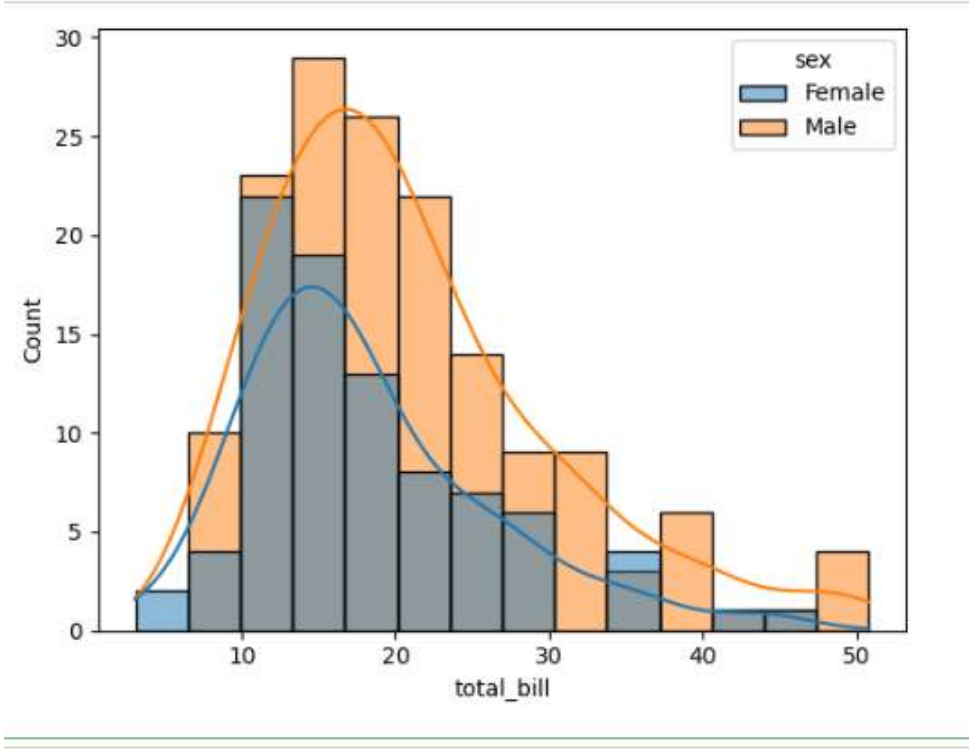
Example:

**# 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()



## 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.
- To install this type the below command in the terminal.

```
pip install bokeh
```

### 1. Scatter Plot

- Scatter Plot in Bokeh can be plotted using the scatter() method of the plotting module. Here pass the x and y coordinates respectively.

Example:

**# 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**

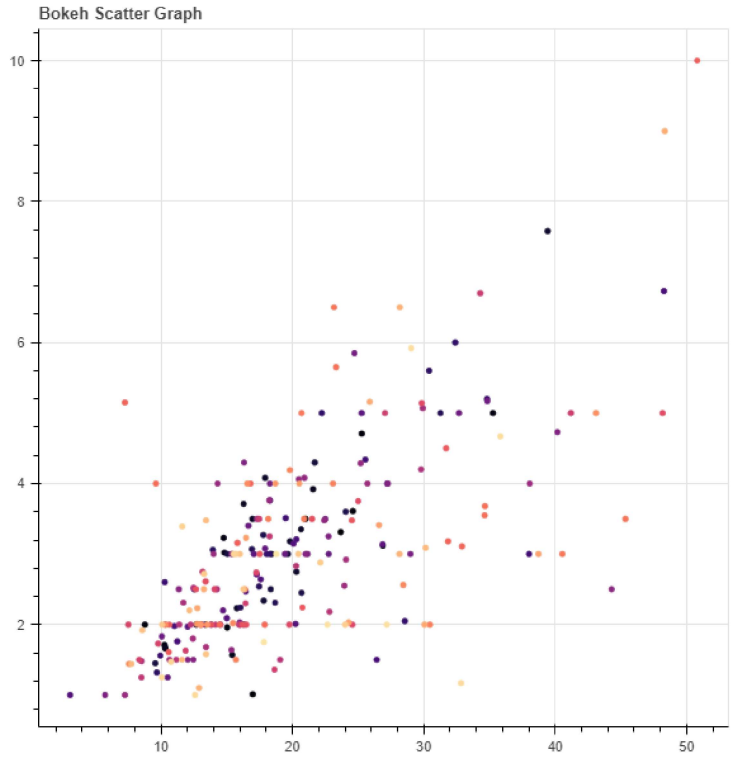
- data = pd.read_csv("tips.csv")
- color = magma(256)

**# plotting the graph**

- graph.scatter(data['total_bill'], data['tip'], color=color)

**# displaying the model**

- show(graph)



## 2. Line Chart

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

Example:

**# importing the modules**

- from bokeh.plotting import figure, output_file, show
- import pandas as pd

**# instantiating the figure object**

- graph = figure(title = "Bokeh Bar Chart")

**# reading the database**

- data = pd.read_csv("tips.csv")

**# Count of each unique value of tip column**

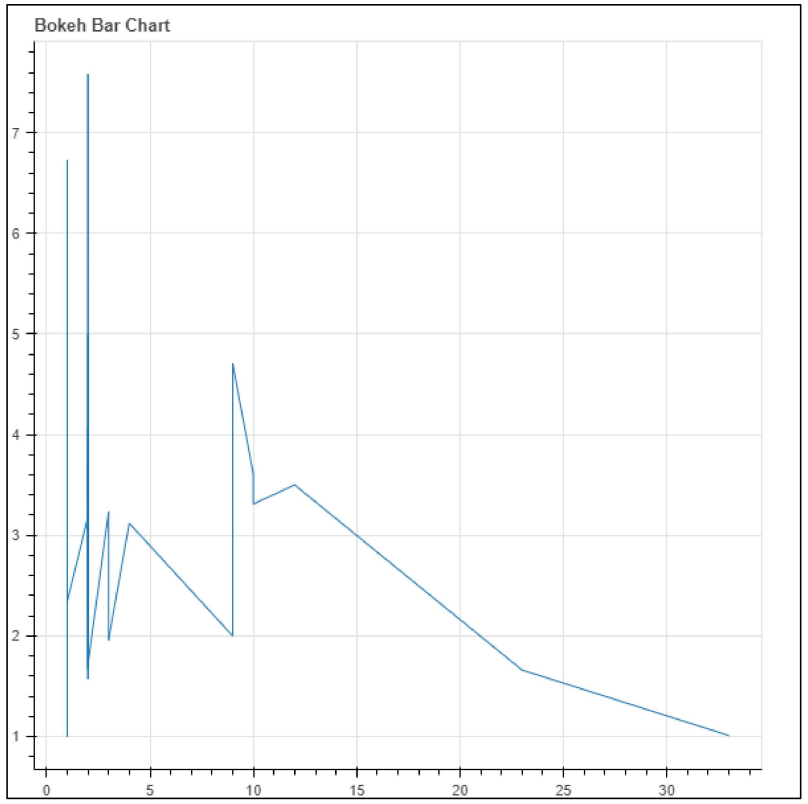
- df = data['tip'].value_counts()

**# plotting the graph**

- graph.line(df, data['tip'])

**# displaying the model**

- show(graph)



## 3. 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.

**Example:**

**# importing the modules**

- from bokeh.plotting import figure, output_file, show
- import pandas as pd

**# instantiating the figure object**

- graph = figure(title = "Bokeh Bar Chart")

**# reading the database**

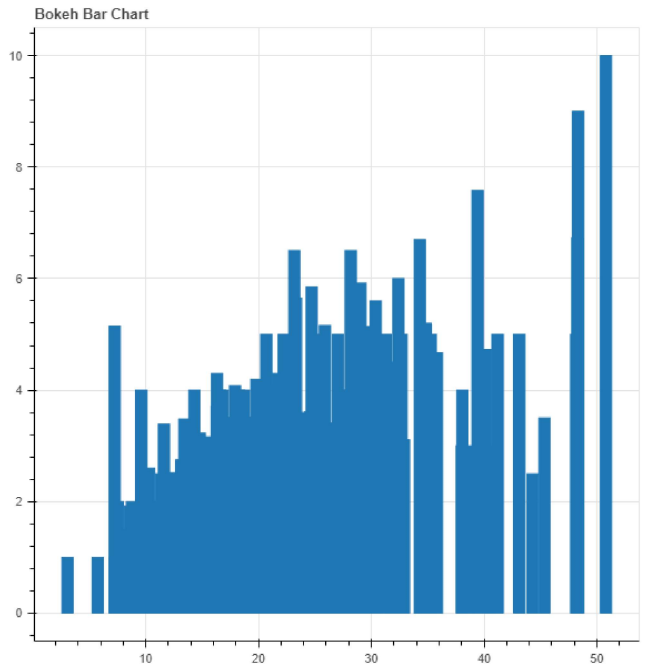
- data = pd.read_csv("C:\\Exp\\tips.csv")

**# plotting the graph**

- graph.vbar(data['total_bill'], top=data['tip'])

**# displaying the model**

- show(graph)



**Interactive Data Visualization**

- One of the key features of Bokeh is to add interaction to the plots. Let's see various interactions that can be added.

**Interactive Legends**

- click_policy property makes the legend interactive. There are two types of interactivity –
- **Hiding:** Hides the Glyphs.
- **Muting:** Hiding the glyph makes it vanish completely, on the other hand, muting the glyph just de-emphasizes the glyph based on the parameters.

Example:

**# importing the modules**

- from bokeh.plotting import figure, output_file, show
- import pandas as pd

**# instantiating the figure object**

- graph = figure(title = "Bokeh Bar Chart")

**# reading the database**

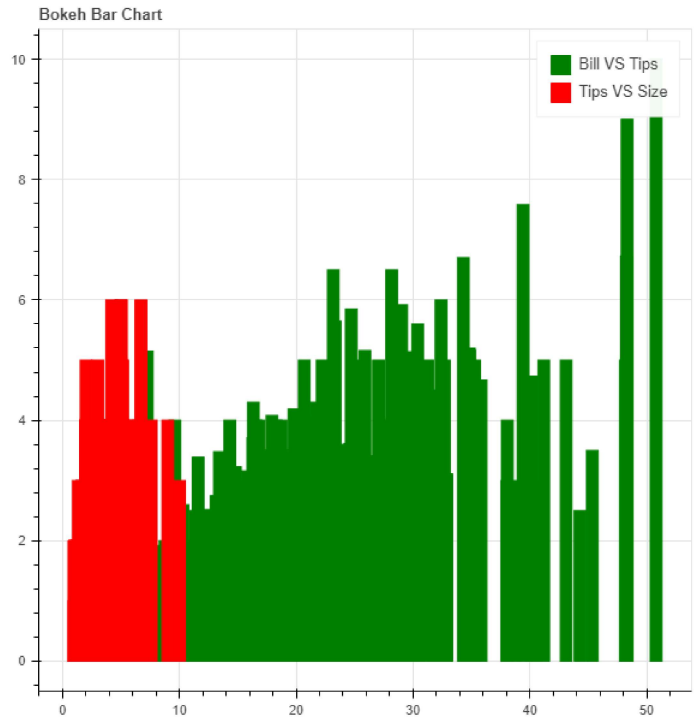
- data = pd.read_csv("tips.csv")

**# plotting the graph**

- 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 Size", color='red')
- graph.legend.click_policy = "hide"

**# displaying the model**

- show(graph)



**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.
- **Note:** All these buttons will be opened on a new tab.

**Example:**

```
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.

**Example:**

- 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 –
- Plotly 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.
- To install it type the below command in the terminal.

```
pip install plotly
```

**1. 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.

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

**# reading the database**

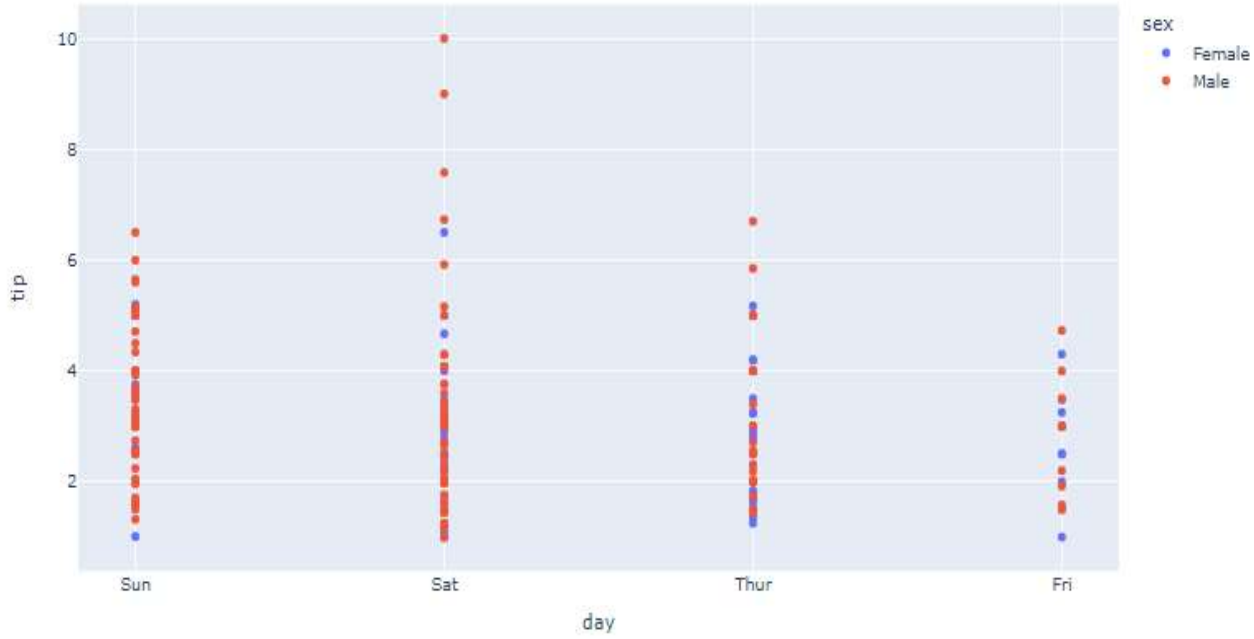
- data = pd.read_csv("tips.csv")

**# plotting the scatter chart**

- fig = px.scatter(data, x="day", y="tip", color='sex')

**# showing the plot**

- fig.show()



## 2. 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

Example:

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

**# reading the database**

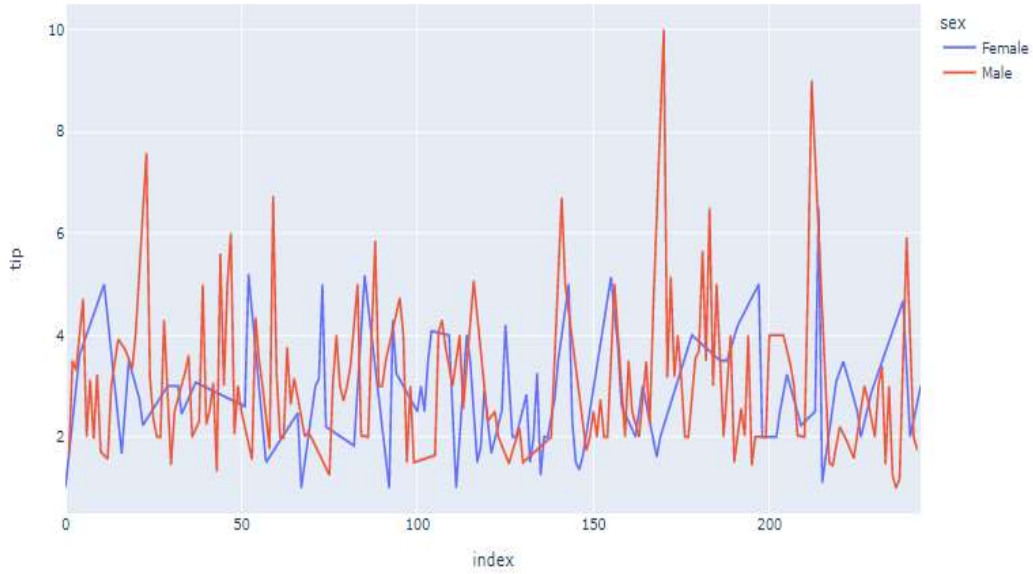
- data = pd.read_csv("C:\\Exp\\tips.csv")

**# plotting the scatter chart**

- fig = px.line(data, y='tip', color='sex')

**# showing the plot**

- fig.show()



### 3. Bar Chart

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

**Example:**

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

**# reading the database**

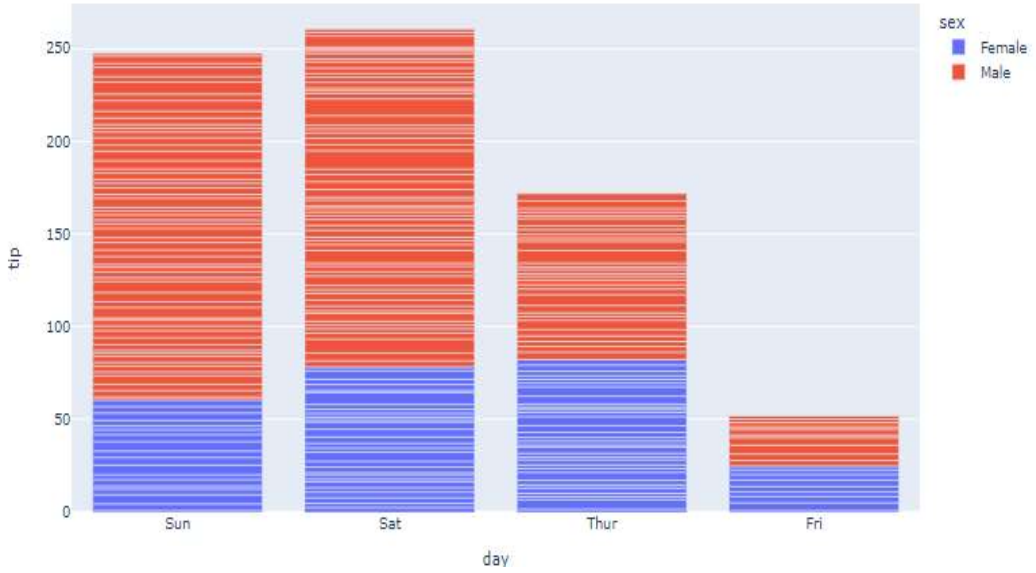
- data = pd.read_csv("tips.csv")

**# plotting the scatter chart**

- fig = px.bar(data, x='day', y='tip', color='sex')

**# showing the plot**

- fig.show()



## 4. Histogram

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

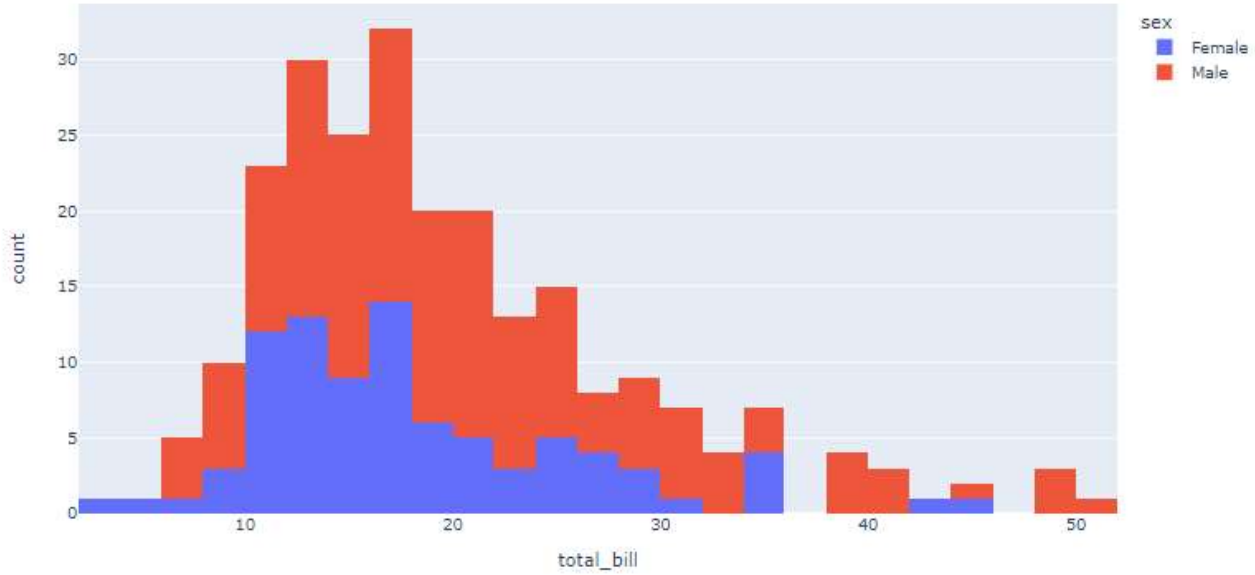
Example:

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

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

# plotting the scatter chart
fig = px.histogram(data, x='total_bill', color='sex')

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



## Measures of Central Tendency

- Measures of central tendency are statistical measures that describe the center or average of a set of data values. They provide a single value that represents the central or typical value in a dataset. The three main measures of central tendency are:

**Mean:**

- The mean, often referred to as the average, is calculated by **adding up all the values** in a dataset and then **dividing by the number of values**.
- Formula:

$$m = \frac{\text{sum of the terms}}{\text{number of terms}}$$

$m$ = mean

**Median:**

- The median is the middle value in a dataset when it is ordered from least to greatest. If there is an even number of values, the median is the average of the two middle values.
- To find the median, the data must be sorted first.
- For an odd number of values: Median = Middle value
- For an even number of values: Median =Sum of two middle values / 2

**Mode:**

- The mode is the value that appears most frequently in a dataset.
- A dataset may have no mode (if all values occur with the same frequency), one mode (unimodal), or multiple modes (multimodal).

## Dispersion

- Dispersion measures in statistics quantify the extent to which data values spread out or deviate from the central tendency. They provide insights into the variability, or spread, within a dataset. Common measures of dispersion include:

**Range:**

- The range is the simplest measure of dispersion and is calculated as the difference between the maximum and minimum values in a dataset.
- **Formula: Range = Maximum value − Minimum value**

**Variance:**

- Variance measures the average squared deviation of each data point from the mean of the dataset.

$$\sigma^2 = \frac{\Sigma (x - \mu)^2}{N}$$ Population Variance

$$s^2 = \frac{\Sigma (x - \bar{x})^2}{n - 1}$$ Sample Variance

**Standard Deviation:**

- The standard deviation is the square root of the variance. It provides a more interpretable measure of dispersion in the same units as the original data.
- Formula for population standard deviation:

$$\mathrm{SD} = \sqrt{\mathrm{Var}}$$

- Formula for sample standard deviation

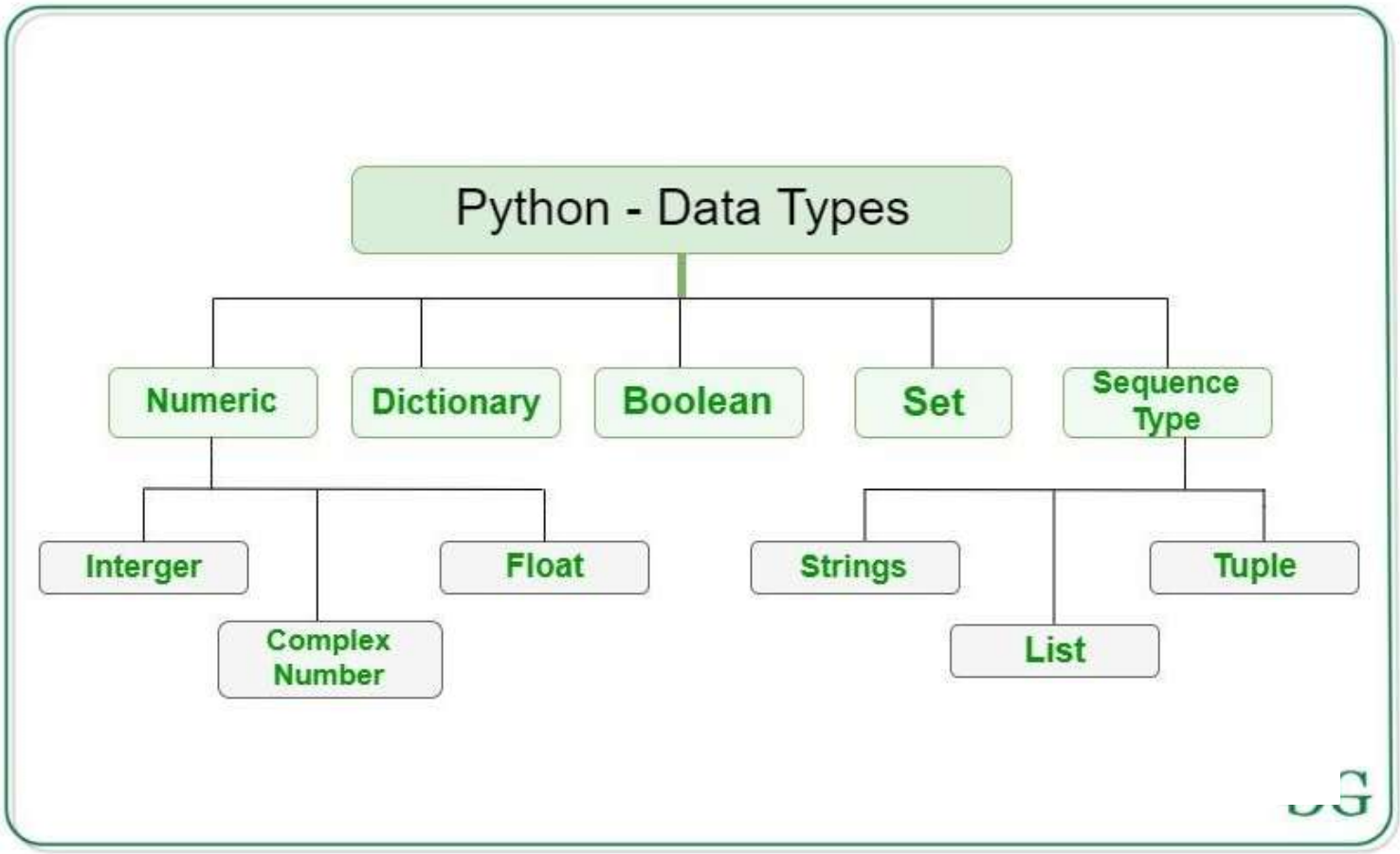
$$\mathrm{SD} = \sqrt{\mathrm{Var}}$$

**Interquartile Range (IQR):**

- The IQR is a measure of statistical dispersion, or in simple terms, it represents the range within which the middle 50% of the data values lie.
- It is the difference between the third quartile (Q3) and the first quartile (Q1).
- Formula: IQR=Q3−Q1

## Different types of data and data structures use cases? Using Python

- Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.
- Following are the standard or built-in data type of Python:
  1. Numeric
  2. Sequence Type
  3. Boolean
  4. Set
  5. Dictionary


Python - Data Types
Numeric
Dictionary
Boolean
Set
Sequence Type
Interger
Complex Number
Float
Strings
List
Tuple

**Numeric**

- In Python, numeric data type represent the data which has numeric value. Numeric value can be integer, floating number or even complex numbers. These values are defined as int, float and complex class in Python.
- Integers – This value is represented by **int** class. It contains positive or negative whole numbers (without fraction or decimal). In Python there is no limit to how long an integer value can be.
- Float – This value is represented by **float** class. It is a real number with floating point representation. It is specified by a decimal point. Optionally, the character e or E followed by a positive or negative integer may be appended to specify scientific notation.
- Complex Numbers – Complex number is represented by complex class. It is specified as (real part) + (imaginary part)j. For example – 2+3j

**Note – type() function is used to determine the type of data type.**

## Sequence Type

- In Python, sequence is the ordered collection of similar or different data types. Sequences allows to store multiple values in an organized and efficient fashion. There are several sequence types in Python –

  String

  List

  Tuple

## 1) String

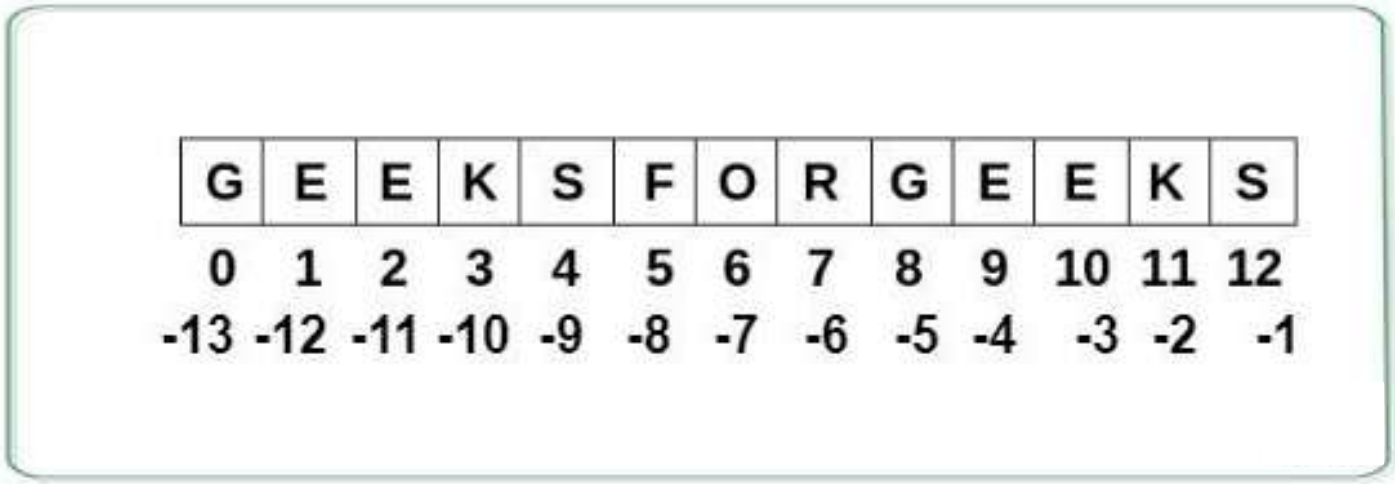
- In Python, Strings are arrays of bytes representing Unicode characters. A string is a collection of one or more characters put in a single quote, double-quote or triple quote.
- In python there is no character data type, a character is a string of length one. It is represented by **str** class.

## Creating String

- Strings in Python can be created using single quotes or double quotes or even triple quotes.

## Accessing elements of String

- In Python, individual characters of a String can be accessed by using the method of Indexing. Indexing allows negative address references to access characters from the back of the String, e.g. -1 refers to the last character, -2 refers to the second last character and so on.



**2) List**

- Lists are just like the arrays, declared in other languages which is a ordered collection of data. It is very flexible as the items in a list do not need to be of the same type.

Creating List

- Lists in Python can be created by just placing the sequence inside the square brackets[].

**Accessing elements of List**

- In order to access the list items refer to the index number. Use the index operator [ ] to access an item in a list. In Python, negative sequence indexes represent positions from the end of the array.
- Instead of having to compute the offset as in List[len(List)-3], it is enough to just write List[-3]. Negative indexing means beginning from the end, -1 refers to the last item, -2 refers to the second-last item, etc.

**3) Tuple**

- Just like list, tuple is also an ordered collection of Python objects. The only difference between tuple and list is that tuples are immutable i.e. tuples cannot be modified after it is created. It is represented by tuple class.

**Creating Tuple**

- In Python, tuples are created by placing a sequence of values separated by 'comma' with or without the use of parentheses for grouping of the data sequence. Tuples can contain any number of elements and of any datatype (like strings, integers, list, etc.).

Note: Tuples can also be created with a single element, but it is a bit tricky. Having one element in the parentheses is not sufficient, there must be a trailing 'comma' to make it a tuple.

**Note – Creation of Python tuple without the use of parentheses is known as Tuple Packing.**

**Accessing elements of Tuple**

- In order to access the tuple items refer to the index number. Use the index operator [ ] to access an item in a tuple. The index must be an integer. Nested tuples are accessed using nested indexing.

**Boolean**

- Data type with one of the two built-in values, True or False. Boolean objects that are equal to True are truthy (true), and those equal to False are falsy (false). But non-Boolean objects can be evaluated in Boolean context as well and determined to be true or false. It is denoted by the class bool.

Note – True and False with capital 'T' and 'F' are valid booleans otherwise python will throw an error.

**Set**

- In Python, Set is an unordered collection of data type that is iterable, mutable and has no duplicate elements. The order of elements in a set is undefined though it may consist of various elements.

**Creating Sets**

- Sets can be created by using the built-in set() function with an iterable object or a sequence by placing the sequence inside curly braces, separated by 'comma'. Type of elements in a set need not be the same, various mixed-up data type values can also be passed to the set.

**Accessing elements of Sets**

- Set items cannot be accessed by referring to an index, since sets are unordered the items has no index. But you can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in keyword.

**Dictionary**

- Dictionary in Python is an unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds key:value pair. Key-value is provided in the dictionary to make it more optimized. Each key-value pair in a Dictionary is separated by a colon :, whereas each key is separated by a 'comma'.

**Creating Dictionary**

- In Python, a Dictionary can be created by placing a sequence of elements within curly {} braces, separated by 'comma'. Values in a dictionary can be of any datatype and can be duplicated, whereas keys can't be repeated and must be immutable. Dictionary can also be created by the built-in function dict(). An empty dictionary can be created by just placing it to curly braces{}.

Note – Dictionary keys are case sensitive, same name but different cases of Key will be treated distinctly.

## Provide an explanation of Jupyter Notebook

- Jupyter Notebook is an open-source, interactive web application that allows you to create and share documents containing live code, equations, visualizations, and narrative text. It is widely used in data science, machine learning, research, and education.
- The name "Jupyter" is a combination of the three core programming languages it supports: Julia, Python, and R.

**Support for Multiple Languages:**

- Jupyter Notebook supports various programming languages, not just the three in its name. This includes languages like Scala, Bash, and others, making it a versatile tool for different tasks.

**Interactive Computing:**

- It allows you to execute code in a step-by-step manner, making it easy to experiment, test, and debug code snippets.

**Rich Output:**

- Besides code execution, Jupyter can display rich output such as HTML, images, videos, and interactive widgets directly in the notebook. This makes it a powerful tool for data visualization and exploration.

**Markdown Support:**

- Jupyter Notebooks support Markdown cells, enabling the inclusion of formatted text, headings, lists, and even LaTeX equations.
- This makes it easy to create well-documented and organized documents.

**Integrated Data Visualization:**

- It seamlessly integrates with popular Python libraries like Matplotlib, Seaborn, and Plotly, allowing you to create interactive and static visualizations within the notebook.

**Ease of Sharing:**

- Notebooks can be easily shared with others by exporting them in various formats, such as HTML, PDF, or slides. This facilitates collaboration and communication of data analyses and research findings.

**Kernel Architecture:**

- Jupyter uses a modular architecture with kernels. Each kernel is responsible for executing code in a specific language. For example, there are kernels for Python, R, Julia, etc.

**Support for Data Science Libraries:**

- Jupyter is commonly used in conjunction with data science libraries like NumPy, Pandas, SciPy, and scikit-learn. This integration makes it a popular environment for data analysis and machine learning.

**Using Jupyter Notebook:**

**Installation:**

- Jupyter can be installed using the Python package manager, pip. You can install it with the command: pip install jupyter.

**Launching Jupyter Notebook:**

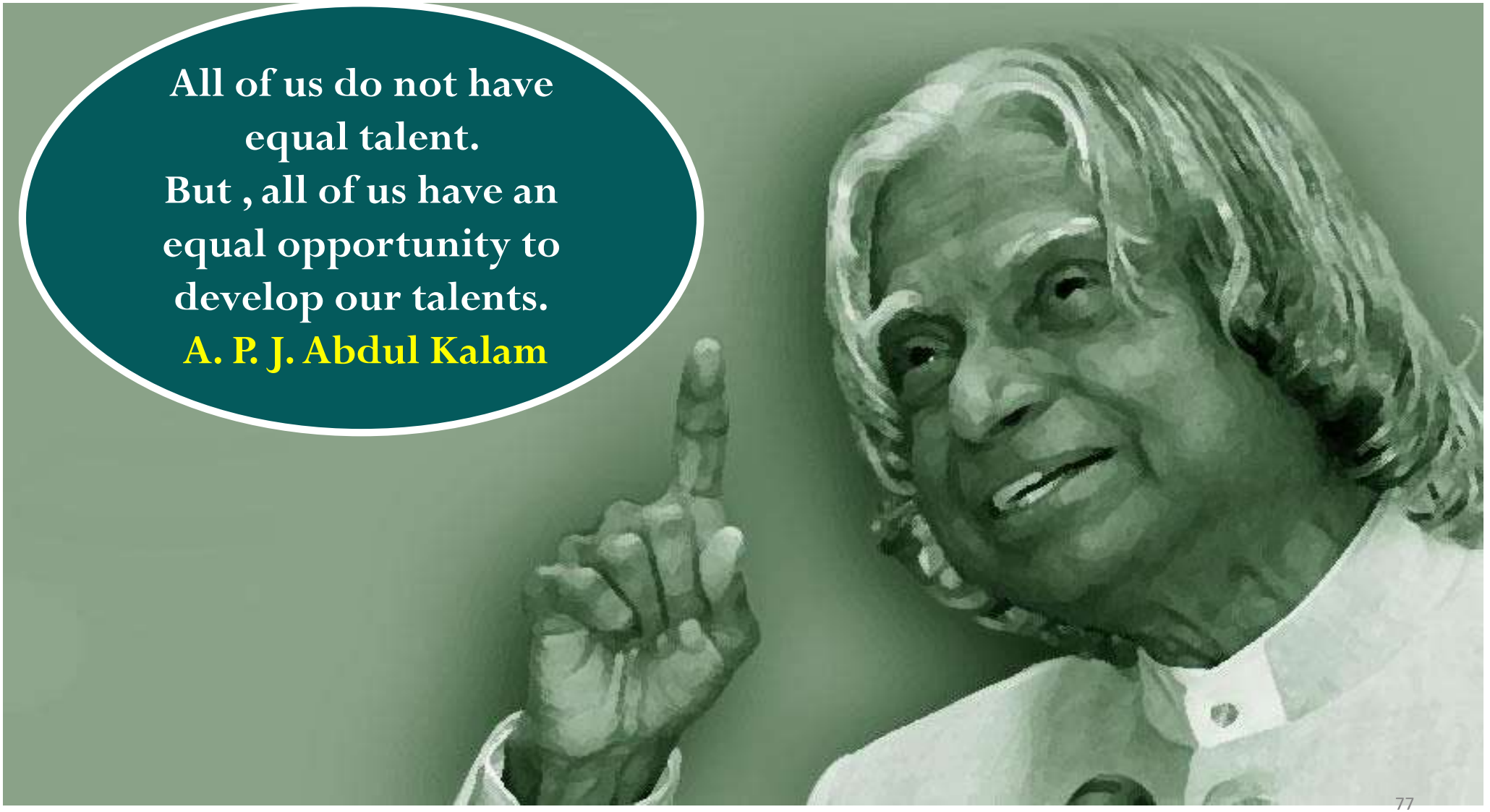
- After installation, you can start Jupyter by running the command jupyter notebook in your terminal. This will open a new tab in your web browser with the Jupyter dashboard.

**Creating and Running Cells:**

- In a Jupyter Notebook, code is written and executed in cells. You can create a new cell by clicking the "+" button on the toolbar and choose the cell type (code or Markdown). To execute a cell, use Shift+Enter.

**Saving and Exporting:**

- Notebooks can be saved using the "Save" option in the toolbar. They can also be exported to different formats through the "File" menu.


All of us do not have
equal talent.
But , all of us have an
equal opportunity to
develop our talents.
A. P. J. Abdul Kalam


Thank
You!