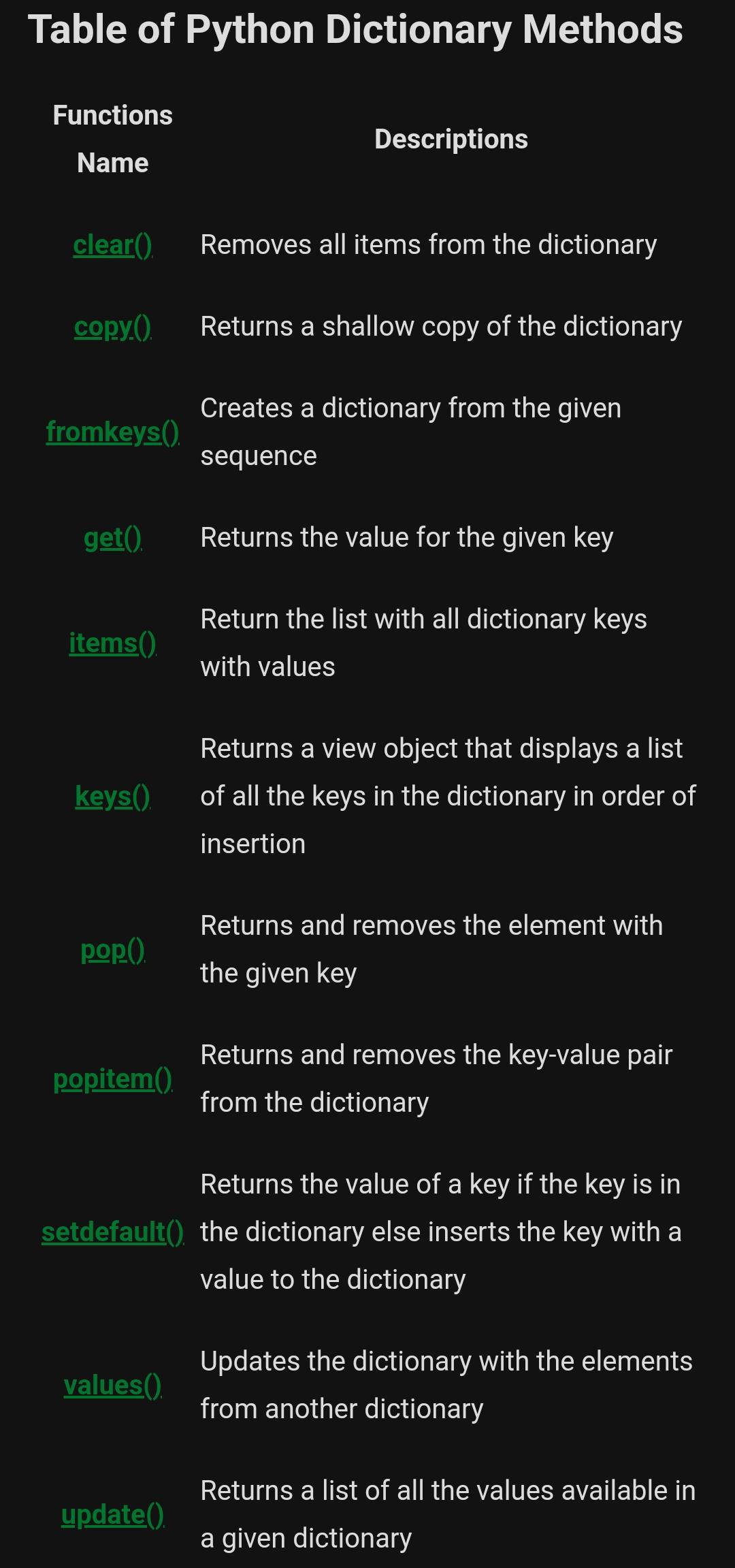
**DV QP CAT 2 Answers**

**1.b) What is dictionary? Explain the methods available in dictionary.**

**Ans.1.b)**

* Dictionaries are used to store data values in key:value pairs.
* A dictionary is a collection which is ordered\*, changeable and do not allow duplicates.
* Dictionaries are written with curly brackets and have keys and values.

The methods available in dictionary are

**2.b) Compare the difference between Numpy and Pandas**

**Ans.2.b)**

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Pandas** | **Numpy** |
| 1 | When we have to work on Tabular data, we prefer the pandas module. | When we have to work on Numerical data, we prefer the numpy module. |
| 2 | The powerful tools of pandas are Data frame and Series. | Whereas the powerful tool of numpy is Arrays. |
| 3 | Pandas consume more memory. | Numpy is memory efficient. |
| 4 | Pandas have a better performance when the number of rows is 500K or more. | Numpy has a better performance when number of rows is 50K or less. |
| 5 | Indexing of the pandas series is very slow as compared to numpy arrays. | Indexing of numpy arrays is very fast. |
| 6 | Pandas have 2d table object called DataFrame. | Numpy is capable of providing multi-dimensional arrays. |
| 7 | It was developed by Wes McKinney and was released in 2008. | It was developed by Travis Oliphant and was released in 2005. |
| 8 | It is used in a lot of organizations like Kaidee, Trivago, Abeja Inc. , and a lot more. | It is being used in organizations like Walmart Tokopedia, Instacart, and many more. |
| 9 | It has a higher industry application. | It has a lower industry application. |

**3.b) How to manually add a legend with a color box on a Matplotlib figure?.**

**Ans. 3.b)**

Matplotlib is a popular data visualization library in Python known for its flexibility and high-quality visualizations. By following this tutorial, you will learn how to create a legend with a color box on your Matplotlib figure, making your visualizations more informative and visually appealing.

Before diving into the code, it is important to understand the different elements of a legend. A legend is a key that labels the elements in our plot with different colors, markers, or lines. By adding a legend, we can understand the data being presented and make it easier for the audience to interpret our visualizations. In the next section, we will look at the syntax for creating a legend with a color box on a Matplotlib figure.

Syntax

To manually add a legend with a color box on a Matplotlib figure in Python, we can use the following syntax −

# Import libraries

import matplotlib.patches as mpatches

# Creating legend with color box

color\_patch = mpatches.Patch(color='red', label='legend')

plt.legend(handles=[color\_patch])

**3.c) How to Generate a Waffle chart? Write a script to plot a waffle chart.**

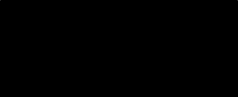
**\*Assume data frame as follow-** **data ={‘printer': ['Nokia', 'Samsung'. 'canon', 'Epson', 'hp'],**

**'stock': [24, 12, 8, 15, 3] }**

**Ans.3.c)**

A Waffle Chart is a gripping visualization technique that is normally created to display progress towards goals. Where each cell in the Waffle Chart constitutes of 10 X 10 cell grid in which each cell represents one percentage point summing up to total 100%. It is commonly an effective option when you are trying to add interesting visualization features to a visual. Waffle Charts are widely used as an Excel dashboard.

For generating Waffle Chart in Python, modules needed are – [matplotlib](https://www.geeksforgeeks.org/python-matplotlib-an-overview/), [pandas](http://geeksforgeeks.org/pandas-tutorial/) and pyWaffle.

To install these packages, run the following commands :

pip install matplotlib

pip install pandas

pip install pywaffle

![A black background with a black square

Description automatically generated with medium confidence](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAO4AAABhCAMAAADm1m0dAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAMUExURSQkJAAAAAwMDAAAAKbJGiwAAAAEdFJOU////wBAKqn0AAAACXBIWXMAAA7EAAAOxAGVKw4bAAAAh0lEQVR4Xu3PsQEAIAyAMPv/0zp4BU0WZs6ZRXbdvt/fJeyW2S2zW2a3zG6Z3TK7ZXbL7JbZLbNbZrfMbpndMrtldsvsltkts1tmt8xumd0yu2V2y+yW2S2zW2a3zG6Z3TK7ZXbL7JbZLbNbZrfMbpndMrtldsvsltkts1tmt8xumd0yu10zF9t2Wi2ZeGLFAAAAAElFTkSuQmCC)The Script to plot waffle chart of printer:

# python program to generate Waffle Chart

# importing all necessary requirements

import pandas as pd

import matplotlib.pyplot as plt

from pywaffle import Waffle

# creation of a dataframe

data ={‘printer': ['Nokia', 'Samsung'. 'canon', 'Epson', 'hp'],

'stock': [24, 12, 8, 15, 3]

}

df = pd.DataFrame(data)

# To plot the waffle Chart

fig = plt.figure(

    FigureClass = Waffle,

    rows = 5,

    values = df.stock,

    labels = list(df.printer)

)

**4.b) Create a list of nested dictionaries into pandas dataframe.**

**Ans.4.b)**

A list of the nested dictionary, a Python program to create a Pandas dataframe using it.

The stepwise procedure to create a Pandas Dataframe using the list of nested dictionary.

**Step #1:** Creating a list of nested dictionary.

**Step #2:** Adding dict values to rows.

**Step #3:** Pivoting DataFrame and assigning column names.

**![A black background with a black square

Description automatically generated with medium confidence](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAMkAAABSCAMAAADJuQC3AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAMUExURRsbGwAAAAkJCQAAAE3tO8cAAAAEdFJOU////wBAKqn0AAAACXBIWXMAAA7EAAAOxAGVKw4bAAAAd0lEQVR4Xu3PMREAIBDAMB7/olkQUbhk6tq15w973Xifkx4nPU56nPQ46XHS46THSY+THic9Tnqc9DjpcdLjpMdJj5MeJz1Oepz0OOlx0uOkx0mPkx4nPU56nPQ46XHS46THSY+THic9Tnqc9DjpcdLjpMdJzcwBHeJAZRfWA6YAAAAASUVORK5CYII=)Example:**

import pandas as pd

# List of nested dictionaries

data = [

{'Name': 'Alice', 'Age': 25, 'City': 'New York'},

{'Name': 'Bob', 'Age': 30, 'City': 'San Francisco'},

{'Name': 'Charlie', 'Age': 28, 'City': 'Los Angeles'},

]

# Creating a Pandas DataFrame

df = pd.DataFrame(data)

# Displaying the DataFrame

print(df)

This will create a DataFrame with columns 'Name', 'Age', and 'City', and each row represents a person with their respective details.

**![A black background with a black square

Description automatically generated with medium confidence](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAMkAAABSCAMAAADJuQC3AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAMUExURRsbGwAAAAkJCQAAAE3tO8cAAAAEdFJOU////wBAKqn0AAAACXBIWXMAAA7EAAAOxAGVKw4bAAAAd0lEQVR4Xu3PMREAIBDAMB7/olkQUbhk6tq15w973Xifkx4nPU56nPQ46XHS46THSY+THic9Tnqc9DjpcdLjpMdJj5MeJz1Oepz0OOlx0uOkx0mPkx4nPU56nPQ46XHS46THSY+THic9Tnqc9DjpcdLjpMdJzcwBHeJAZRfWA6YAAAAASUVORK5CYII=)Output:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Age | City |
| 0 | Alice | 25 | New York |
| 1 | Bob | 30 | San Francisco |
| 2 | Charlie | 28 | Las Angeles |

You can customize the data in the nested dictionaries to fit your specific requirements. Each key in the dictionaries becomes a column in the DataFrame, and each dictionary becomes a row.

**4.c) Explain the steps of Creating a Word Cloud in Python.**

**Ans.4.c)**

Creating a word cloud in Python involves visualizing the frequency of words in a text dataset in a visually appealing manner.

The steps to create a Word Cloud using Python:

**1. Install Required Libraries:**

**![A black background with a black square

Description automatically generated with medium confidence](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAMkAAABSCAMAAADJuQC3AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAMUExURRsbGwAAAAkJCQAAAE3tO8cAAAAEdFJOU////wBAKqn0AAAACXBIWXMAAA7EAAAOxAGVKw4bAAAAd0lEQVR4Xu3PMREAIBDAMB7/olkQUbhk6tq15w973Xifkx4nPU56nPQ46XHS46THSY+THic9Tnqc9DjpcdLjpMdJj5MeJz1Oepz0OOlx0uOkx0mPkx4nPU56nPQ46XHS46THSY+THic9Tnqc9DjpcdLjpMdJzcwBHeJAZRfWA6YAAAAASUVORK5CYII=)** - First, make sure you have the necessary libraries installed. You can install them using:

pip install wordcloud matplotlib

**2. Import Libraries:**

**![A black background with a black square

Description automatically generated with medium confidence](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAMkAAABSCAMAAADJuQC3AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAMUExURRsbGwAAAAkJCQAAAE3tO8cAAAAEdFJOU////wBAKqn0AAAACXBIWXMAAA7EAAAOxAGVKw4bAAAAd0lEQVR4Xu3PMREAIBDAMB7/olkQUbhk6tq15w973Xifkx4nPU56nPQ46XHS46THSY+THic9Tnqc9DjpcdLjpMdJj5MeJz1Oepz0OOlx0uOkx0mPkx4nPU56nPQ46XHS46THSY+THic9Tnqc9DjpcdLjpMdJzcwBHeJAZRfWA6YAAAAASUVORK5CYII=)** - Import the required libraries in your Python script or Jupyter Notebook.

import matplotlib.pyplot as plt

from wordcloud import WordCloud

**3. Prepare Text Data:**

**![A black background with a black square

Description automatically generated with medium confidence](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAMkAAABSCAMAAADJuQC3AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAMUExURRsbGwAAAAkJCQAAAE3tO8cAAAAEdFJOU////wBAKqn0AAAACXBIWXMAAA7EAAAOxAGVKw4bAAAAd0lEQVR4Xu3PMREAIBDAMB7/olkQUbhk6tq15w973Xifkx4nPU56nPQ46XHS46THSY+THic9Tnqc9DjpcdLjpMdJj5MeJz1Oepz0OOlx0uOkx0mPkx4nPU56nPQ46XHS46THSY+THic9Tnqc9DjpcdLjpMdJzcwBHeJAZRfWA6YAAAAASUVORK5CYII=)** - Load or prepare the text data that you want to create a word cloud for. This could be a string of text or a document.

text\_data = "Your text data goes here. This is a sample text for creating a word cloud."

**4. Generate Word Cloud:**

**![A black background with a black square

Description automatically generated with medium confidence](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAMkAAABSCAMAAADJuQC3AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAMUExURRsbGwAAAAkJCQAAAE3tO8cAAAAEdFJOU////wBAKqn0AAAACXBIWXMAAA7EAAAOxAGVKw4bAAAAd0lEQVR4Xu3PMREAIBDAMB7/olkQUbhk6tq15w973Xifkx4nPU56nPQ46XHS46THSY+THic9Tnqc9DjpcdLjpMdJj5MeJz1Oepz0OOlx0uOkx0mPkx4nPU56nPQ46XHS46THSY+THic9Tnqc9DjpcdLjpMdJzcwBHeJAZRfWA6YAAAAASUVORK5CYII=)** - Create a WordCloud object and generate the word cloud from the text data. You can customize parameters like the background color, maximum number of words, and font size.

wordcloud = WordCloud(width=800, height=400, background\_color='white').generate(text\_data)

**5. Display the Word Cloud:**

**![A black background with a black square

Description automatically generated with medium confidence](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAMkAAABSCAMAAADJuQC3AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAMUExURRsbGwAAAAkJCQAAAE3tO8cAAAAEdFJOU////wBAKqn0AAAACXBIWXMAAA7EAAAOxAGVKw4bAAAAd0lEQVR4Xu3PMREAIBDAMB7/olkQUbhk6tq15w973Xifkx4nPU56nPQ46XHS46THSY+THic9Tnqc9DjpcdLjpMdJj5MeJz1Oepz0OOlx0uOkx0mPkx4nPU56nPQ46XHS46THSY+THic9Tnqc9DjpcdLjpMdJzcwBHeJAZRfWA6YAAAAASUVORK5CYII=)** - Use Matplotlib to display the generated word cloud.

plt.figure(figsize=(10, 5))

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis('off') # Turn off axis labels

plt.show()

**6. Optional: Save Word Cloud to File:**

**![A black background with a black square

Description automatically generated with medium confidence](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAMkAAABSCAMAAADJuQC3AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAMUExURRsbGwAAAAkJCQAAAE3tO8cAAAAEdFJOU////wBAKqn0AAAACXBIWXMAAA7EAAAOxAGVKw4bAAAAd0lEQVR4Xu3PMREAIBDAMB7/olkQUbhk6tq15w973Xifkx4nPU56nPQ46XHS46THSY+THic9Tnqc9DjpcdLjpMdJj5MeJz1Oepz0OOlx0uOkx0mPkx4nPU56nPQ46XHS46THSY+THic9Tnqc9DjpcdLjpMdJzcwBHeJAZRfWA6YAAAAASUVORK5CYII=)** - If you want to save the word cloud as an image file, you can use the `to\_file` method.

wordcloud.to\_file('wordcloud.png')

**![A black background with a black square

Description automatically generated with medium confidence](data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAMkAAABSCAMAAADJuQC3AAAAAXNSR0IArs4c6QAAAARnQU1BAACxjwv8YQUAAAAMUExURRsbGwAAAAkJCQAAAE3tO8cAAAAEdFJOU////wBAKqn0AAAACXBIWXMAAA7EAAAOxAGVKw4bAAAAd0lEQVR4Xu3PMREAIBDAMB7/olkQUbhk6tq15w973Xifkx4nPU56nPQ46XHS46THSY+THic9Tnqc9DjpcdLjpMdJj5MeJz1Oepz0OOlx0uOkx0mPkx4nPU56nPQ46XHS46THSY+THic9Tnqc9DjpcdLjpMdJzcwBHeJAZRfWA6YAAAAASUVORK5CYII=)Complete Example:**

import matplotlib.pyplot as plt

from wordcloud import WordCloud

# Prepare Text Data

text\_data = "Your text data goes here. This is a sample text for creating a word cloud."

# Generate Word Cloud

wordcloud = WordCloud(width=800, height=400, background\_color='white').generate(text\_data)

# Display the Word Cloud

plt.figure(figsize=(10, 5))

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis('off')

plt.show()

# Save Word Cloud to File (Optional)

# wordcloud.to\_file('wordcloud.png')

**5.b) Define following terms: i) Barplot, ii) Boxplot, iii) Stripplot**

**Ans.5.b)**

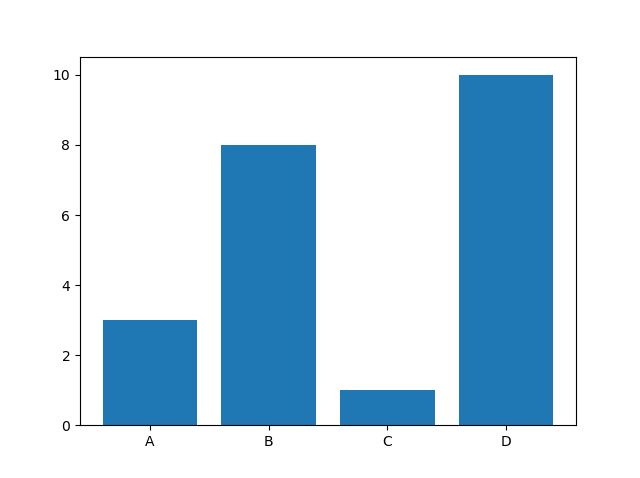
**i) Barplot**

A barplot (or barchart) is one of the most common type of graphic. It shows the

relationship between a numeric variable and a categoric variable.

Bar Plot are classified into four types of graphs - bar graph or bar chart, line graph, pie

chart, and diagram.



**Limitations of Bar Plot:**

• When we try to display changes in speeds such as acceleration, Bar graphs won’t help us.

**Advantages of Bar plot:**

• Bar charts are easy to understand and interpret.

• Relationship between size and value helps for in easy comparison.

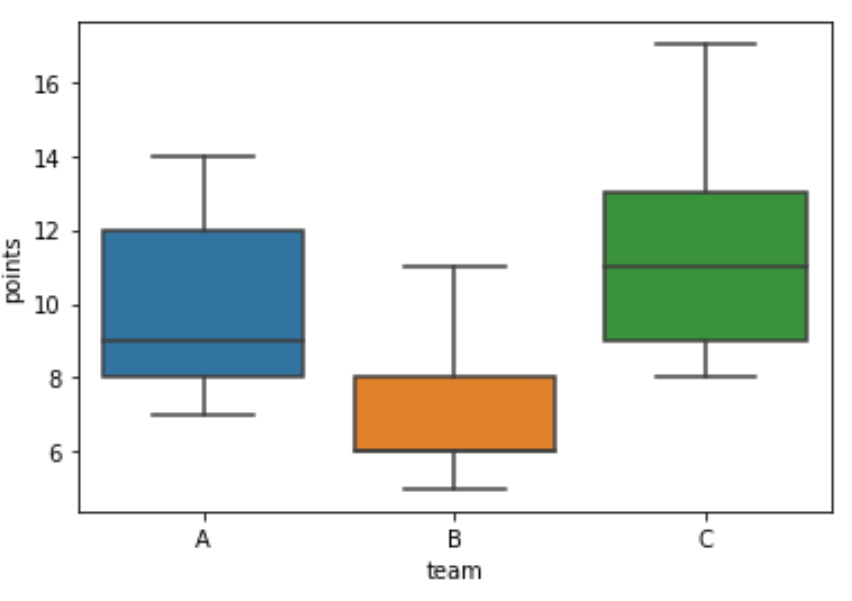
• They're simple to create.

• They can help in presenting very large or very small values easily.

**ii) Boxplot**

A box plot is a way of statistically representing the distribution of the data through five main dimensions :

* Minimum: Smallest number in the dataset.
* First quartile: Middle number between the minimum and the median.
* Second quartile (Median): Middle number of the (sorted) dataset.
* Third quartile: Middle number between median and maximum.
* Maximum: Highest number in the dataset.



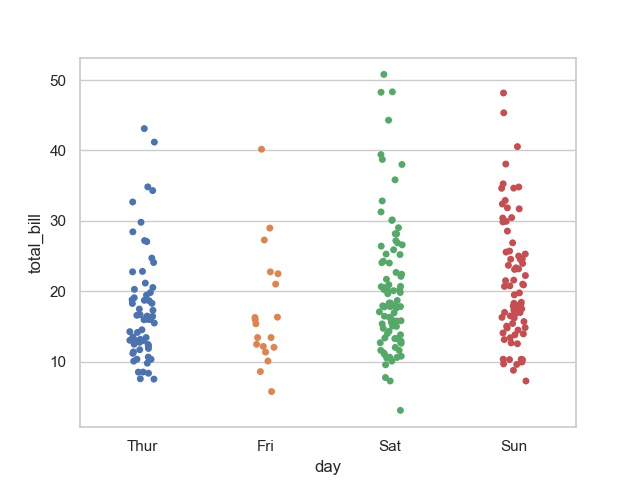
**Advantages:**

* The box plot organizes large amounts of data and visualizes outlier values.

**Disadvantages:**

* The box plot is not relevant for detailed analysis of the data as it deals with a summary of the data distribution.

**iii) Stripplot**

A strip plot is a type of categorical scatter plot that displays individual data points along a one dimensional axis. It is particularly useful for visualizing the distribution of data points within different categories. In a strip plot, each data point is represented as a dot, and the dots are aligned in a single line for each category. 

**Advantages**

• Permits efficient application of factors that would be difficult to apply to small plots

**Disadvantages**

• Differential precision in the estimation of interaction and the main effects

• Complicated statistical analysis

**5.c) How to create a Map with a Dataset? Explain with their syntax and parameter use.**

**Ans.5.c)**

Creating a map with a dataset in Python can be achieved using various libraries, and one popular choice is `folium` for interactive maps. Below are the steps along with syntax and parameter explanations:

**1. Install Required Library:**

- Install the `folium` library if you haven't already:

pip install folium

**2. Import Libraries:**

- Import the necessary libraries in your Python script or Jupyter Notebook.

import folium

**3. Load or Prepare the Dataset:**

- Load or prepare the dataset containing geographical information (latitude, longitude, and any other relevant data).

**4. Create a Map:**

- Use the `folium.Map()` constructor to create a base map. Specify the initial center coordinates and zoom level.

my\_map = folium.Map(location=[latitude, longitude], zoom\_start=12)

- Parameters:

- `location`: A list or tuple containing the latitude and longitude of the map's center.

- `zoom\_start`: Initial zoom level of the map.

**5. Add Markers or GeoJSON Data:**

- Add markers or custom data to the map using `folium.Marker()` for individual markers or `folium.GeoJson()` for GeoJSON data.

# Example with Marker

folium.Marker(location=[marker\_latitude, marker\_longitude], popup='Marker Popup').add\_to(my\_map)

# Example with GeoJSON data

folium.GeoJson(geojson\_data).add\_to(my\_map)

- Parameters:

- `location`: Latitude and longitude of the marker or GeoJSON data.

- `popup`: Text or HTML content that appears when clicking on the marker.

**6. Display or Save the Map:**

- Use the `display()` function if using Jupyter Notebook, or save the map as an HTML file.

# Display the map (if using Jupyter Notebook)

my\_map

# Save the map as an HTML file

my\_map.save('my\_map.html')

**Complete Example:**

import folium

# Create a Map

my\_map = folium.Map(location=[37.7749, -122.4194], zoom\_start=12)

# Add Marker

folium.Marker(location=[37.7749, -122.4194], popup='San Francisco').add\_to(my\_map)

# Display the Map (if using Jupyter Notebook)

my\_map

**6.b) i) What is Seaborn? ii) Different categories of plot in Seaborn.**

**Ans.6.b)**

**i) Seaborn:**

Seaborn is a statistical data visualization library in Python that is based on Matplotlib. It provides a high-level interface for creating informative and attractive statistical graphics. Seaborn comes with several built-in themes and color palettes to make it easy to create aesthetically pleasing visualizations.

**ii) Different Categories of Plots in Seaborn:**

Seaborn provides a variety of plot types that are categorized into different functions. Here are some of the main categories of plots available in Seaborn:

**1. Relational Plots:**

- These plots are used to visualize the relationship between two variables.

**2. Categorical Plots:**

- Categorical plots are used to visualize the distribution of categorical data.

**3. Distribution Plots:**

- These plots are used to visualize the univariate distribution of a variable.

**4. Regression Plots:**

- Regression plots are used for visualizing the relationship between two variables and fitting a regression line.

**5. Matrix Plots:**

- Matrix plots are used to visualize data in matrix form.

**6. Multi-plot Grids:**

- Multi-plot grids allow the creation of complex visualizations with multiple plots.

**7. Axis Grids:**

- Axis grids provide a way to map the structure of a dataset onto the grid of subplots.

**8. Color Palettes:**

- Seaborn provides a variety of color palettes for enhancing the visual appeal of plots.

These categories cover a wide range of visualizations, making Seaborn a powerful tool for statistical data visualization in Python. The library is built on top of Matplotlib and integrates well with Pandas DataFrames.

**6.c) Difference between Matplotlib and Seaborn with examples.**

**Ans.6.c)**

|  |  |  |
| --- | --- | --- |
| **Features** | **Matplotlib** | **Seaborn** |
| **Functionality** | It is utilized for making basic graphs. Datasets are visualised with the help of bargraphs, histograms, piecharts, scatter plots, lines and so on. | Seaborn contains a number of patterns and plots for data visualization. It uses fascinating themes. It helps in compiling whole data into a single plot. It also provides distribution of data. |
| **Syntax** | It uses comparatively complex and lengthy syntax. Example: Syntax for bargraph- matplotlib.pyplot.bar(x\_axis, y\_axis). | It uses comparatively simple syntax which is easier to learn and understand. Example: Syntax for bargraph- seaborn.barplot(x\_axis, y\_axis). |
| **Dealing Multiple Figures** | We can open and use multiple figures simultaneously. However they are closed distinctly. Syntax to close one figure at a time: matplotlib.pyplot.close(). Syntax to close all the figures: matplotlib.pyplot.close(“all”) | Seaborn sets time for the creation of each figure. However, it may lead to (OOM) out of memory issues |
| **Visualization** | Matplotlib is well connected with Numpy and Pandas and acts as a graphics package for data visualization in python. Pyplot provides similar features and syntax as in MATLAB. Therefore, MATLAB users can easily study it. | Seaborn is more comfortable in handling Pandas data frames. It uses basic sets of methods to provide beautiful graphics in python. |
| **Pliability** | Matplotlib is a highly customized and robust | Seaborn avoids overlapping of plots with the help of its default themes |
| **Data Frames and Arrays** | Matplotlib works efficiently with data frames and arrays.It treats figures and axes as objects. It contains various stateful APIs for plotting. Therefore plot() like methods can work without parameters. | Seaborn is much more functional and organized than Matplotlib and treats the whole dataset as a single unit. Seaborn is not so stateful and therefore, parameters are required while calling methods like plot() |
| **Use Cases** | Matplotlib plots various graphs using Pandas and Numpy | Seaborn is the extended version of Matplotlib which uses Matplotlib along with Numpy and Pandas for plotting graphs |