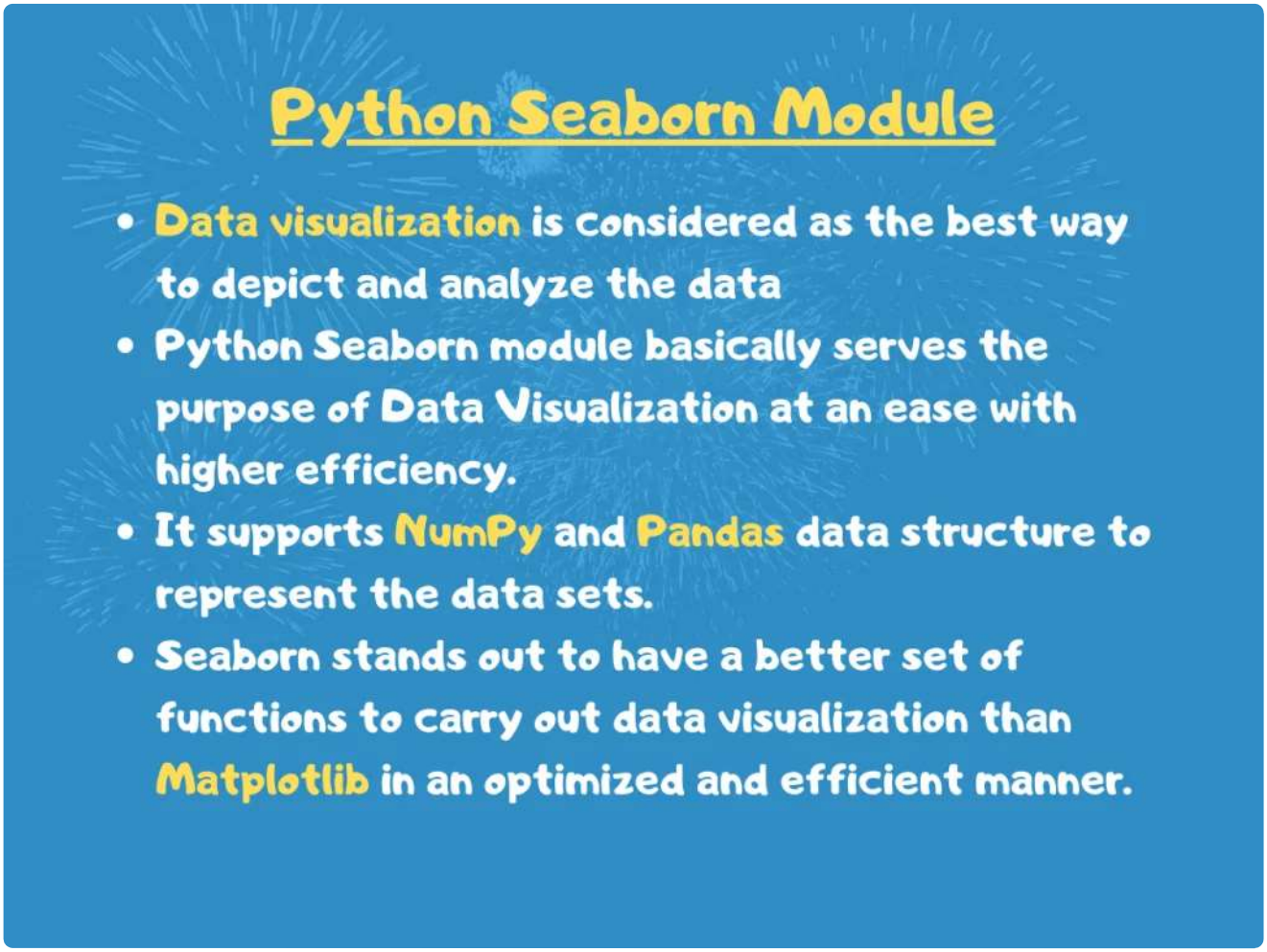
AskPython

Python Seaborn Tutorial



Python Seaborn module serves the purpose of Data Visualization at an ease with higher efficiency. In order to represent the variations in a huge data set, `data visualization` is considered as the best way to depict and analyze the data.

Seaborn stands out to have a better set of functions to carry out data visualization than Matplotlib in an optimized and efficient manner. It supports NumPy and Pandas data structure to represent the data sets.

But, in order to get started with the Seaborn module, I would strongly recommend the readers to understand the Python Matplotlib module.

Getting started with Python Seaborn

In order to get started with the functionalities of Seaborn module, we need to install the module in our environment using the below command:

```
pip install Seaborn
```

Seaborn module requires the following modules installed to work in a smooth manner:

Matplotlib

NumPy

Pandas

SciPy

I've linked the bullet points with the relevant articles for reference.

Data Files Used Throughout the Tutorial

We'll be working with CSV files throughout the tutorial, so this section highlights the files that we'll be using throughout.

Wherever you see a reference to the following file names, you can look back at this section to understand the data that's being passed.

Book1.csv:

	A	B	C	D
1	Name	Age		
2	Jim	21		
3	Jenny	22		
4	Bran	24		
5	Shawn	12		
6	Ritik	26		
7	Rosy	24		
8	Danny	25		
9	Daisy	15		
10	Tom	27		

Book1.csv

tips.csv:

total_bill	tip	sex	smoker	day	time	size
16.99	1.01	Female	No	Sun	Dinner	2
10.34	1.66	Male	No	Sun	Dinner	3
21.01	3.5	Male	No	Sun	Dinner	3
23.68	3.31	Male	No	Sun	Dinner	2
24.59	3.61	Female	No	Sun	Dinner	4
25.29	4.71	Male	No	Sun	Dinner	4
8.77	2	Male	No	Sun	Dinner	2
26.88	3.12	Male	No	Sun	Dinner	4
15.04	1.96	Male	No	Sun	Dinner	2
14.78	3.23	Male	No	Sun	Dinner	2
10.27	1.71	Male	No	Sun	Dinner	2
35.26	5	Female	No	Sun	Dinner	4
15.42	1.57	Male	No	Sun	Dinner	2
18.43	3	Male	No	Sun	Dinner	4
14.83	3.02	Female	No	Sun	Dinner	2
21.58	3.92	Male	No	Sun	Dinner	2
10.33	1.67	Female	No	Sun	Dinner	3
16.29	3.71	Male	No	Sun	Dinner	3
16.97	3.5	Female	No	Sun	Dinner	3
20.65	3.35	Male	No	Sat	Dinner	3

Input csv tips-data set

Python Seaborn For Statistical Analysis

Statistical Analysis is the basic `estimation` out of some parameters of the data-set to a large extent. Data Visualization can be considered as the best way to perform statistical analysis i.e. predict the outcome or the cause based on diagrammatic values.

Either of the following ways can be taken into consideration during the statistical analysis:

seaborn.scatterplot()

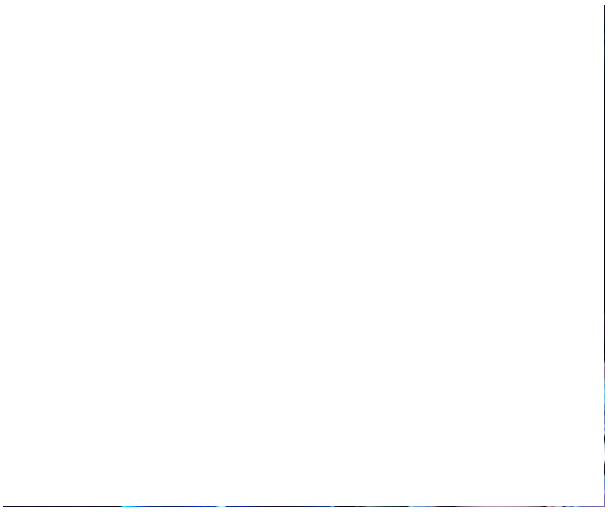
seaborn.lineplot()

---

1. seaborn.scatterplot()

The `seaborn.scatterplot()` function is basically used to depict the relationship between the parameters on the given axes respectively. Every point on the graph depicts a value corresponding to it.

Syntax:



```
seaborn.scatterplot(x=value, y=value, data=data)
```

Example:

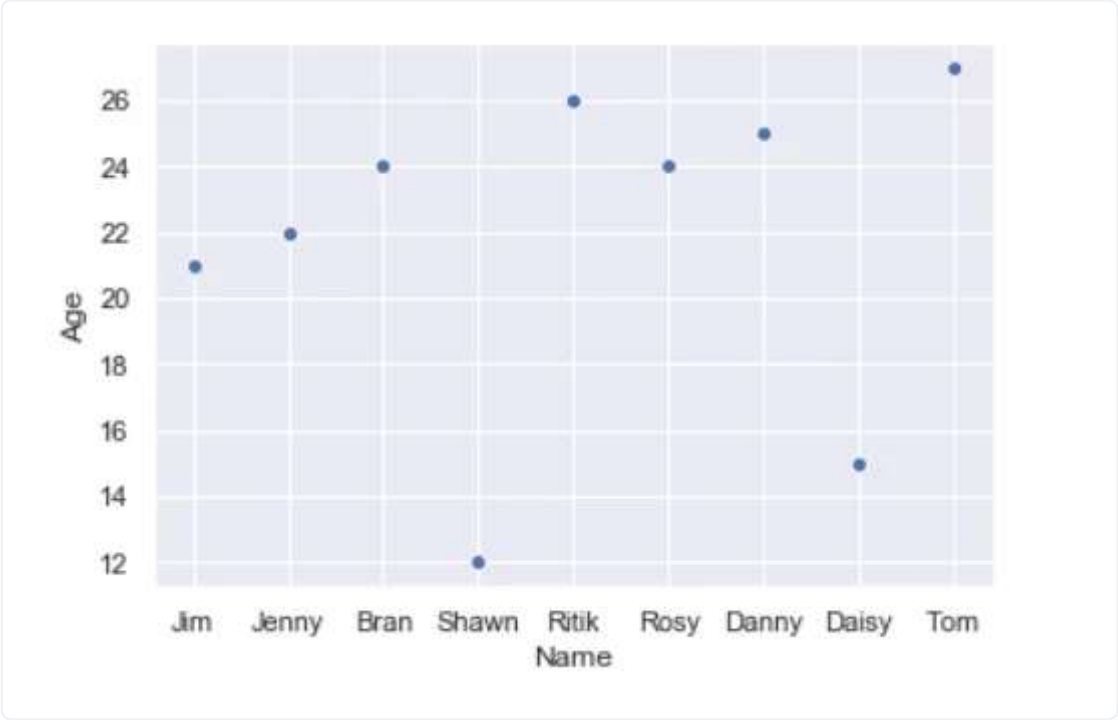
```
import seaborn
import pandas
import matplotlib.pyplot as plt

csv = pandas.read_csv(r'C:\Book1.csv')
res = seaborn.scatterplot(x="Name", y="Age", data=csv)
plt.show()
```

In the above example, we have imported **Python Pandas module** in order to use the `read_csv()` function to read the contents of the data set.

The column-'Name' is represented by the x-axis and the column-'Age' by the y-axis.

Output:



Seaborn ScatterPlot

2. seaborn.lineplot()

The `seaborn.lineplot()` function can be extensively used in situations wherein we feel the need to check the dependency of a parameter on the other in a continuous manner relative to time.

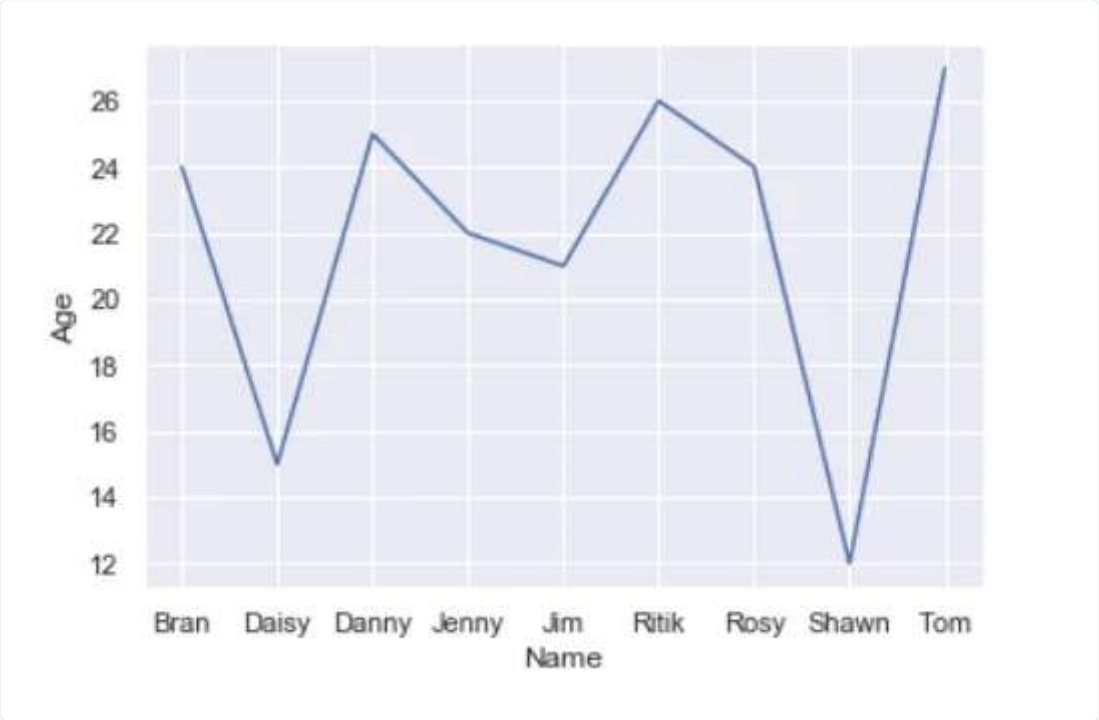
Syntax:

```
seabron.lineplot(x=value, y=value, data=data)
```

Example:

```
import seaborn
import pandas
import matplotlib.pyplot as plt
csv = pandas.read_csv(r'C:\Book1.csv')
res = seaborn.lineplot(x="Name", y="Age", data=csv)
plt.show()
```

Output:



Seaborn LinePlot

Categorical Scatter Plot

Categorical data divides and represents itself in the form of discrete groups i.e. a subset of the original data.

Python Seaborn module contains the following methods to represent and visualize categorical data:

seaborn.catplot()

seaborn.stripplot()

seaborn.swarmplot()

---

1. seaborn.catplot()

The `seaborn.catplot()` function, as mentioned above, is one of the techniques to analyze the relationship between a numeric value and a categorical group of values together.

Syntax:

```
seaborn.catplot(x=value, y=value, data=data)
```

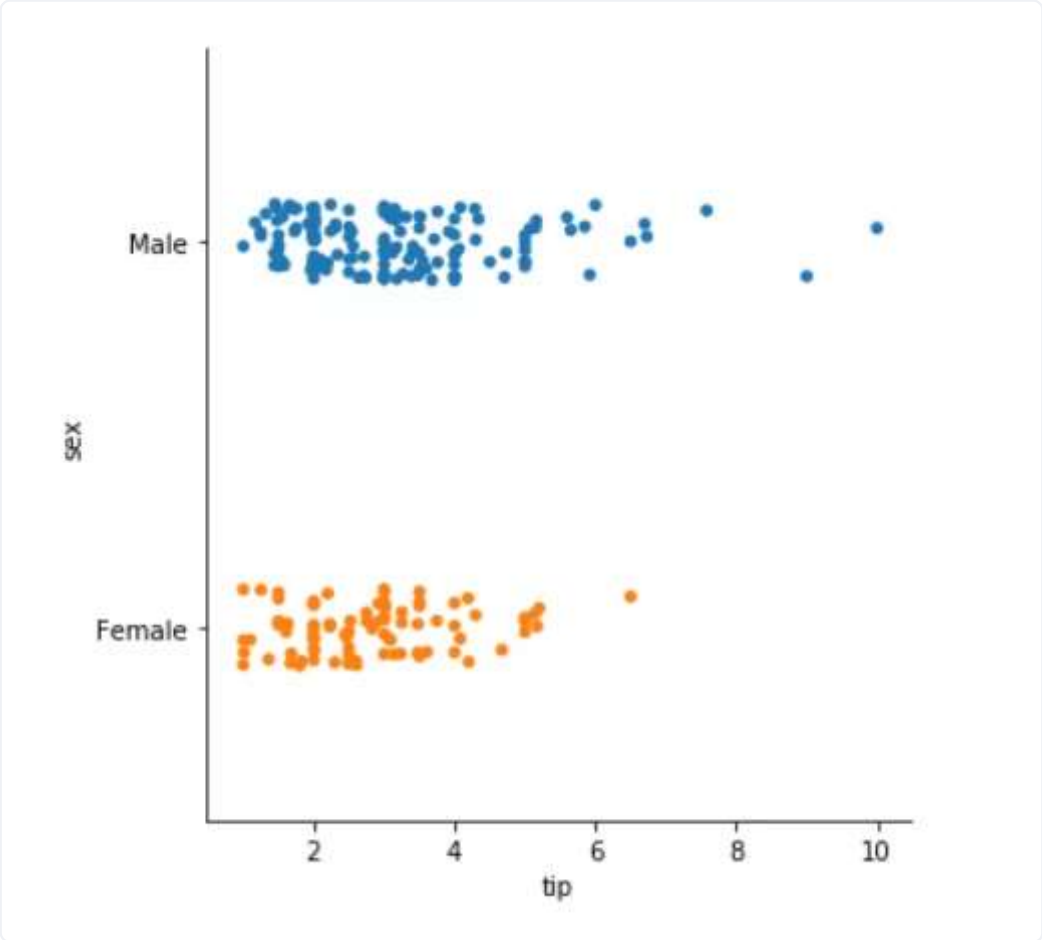
Example:

```
import seaborn
import pandas
import matplotlib.pyplot as plt


csv = seaborn.load_dataset("tips")
res = seaborn.catplot(x="tip", y="sex", data=csv)
```

```
plt.show()
```

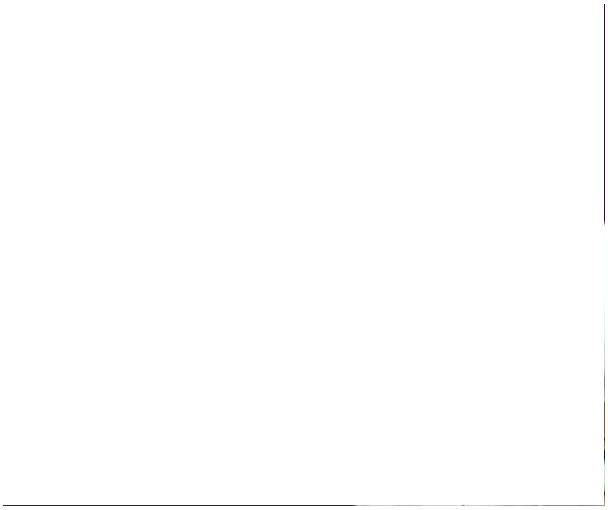
Output:



catplot

2. seaborn.stripplot()

The `seaborn.stripplot()` function considers one of the input columns as categorical data input and then it plots the points accordingly in an ordinal fashion despite the different data type of the input.



Syntax:

```
seaborn.stripplot(x=value, y=value, data=data)
```

Example:

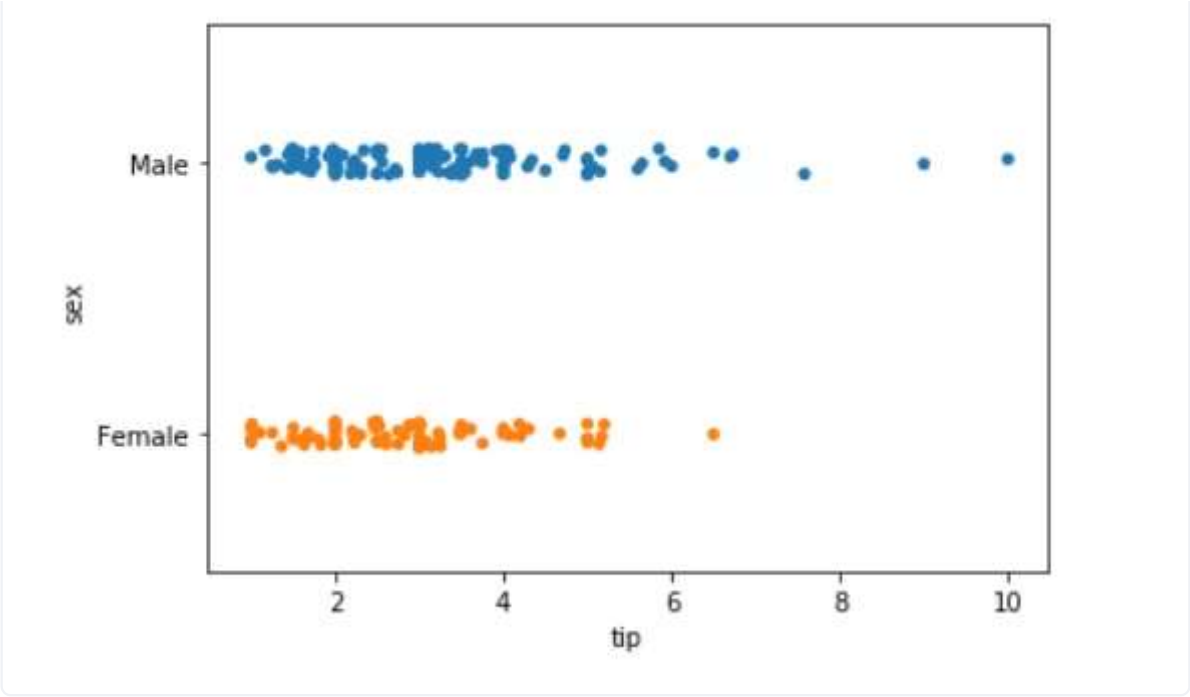
```
import seaborn
import pandas
import matplotlib.pyplot as plt


csv = seaborn.load_dataset("tips")
res = seaborn.stripplot(x="tip", y="sex", data=csv,jitter=0.05)

plt.show()
```

The parameter `jitter` is useful when the data set consists of data points that overlap. In such cases, setting a jitter value can help them get **uniformly distributed**.

Output:



stripplot

3. seaborn.swarmplot()

The seaborn.swarmplot() function resembles the **seaborn.stripplot()** function with a slight difference. The `seaborn.swarmplot()` function plots the data values along the categorical axis chosen. Thus, it completely avoids **overlapping**.

Syntax:

```
seaborn.swarmplot(x=value, y=value, data=data)
```

Example:

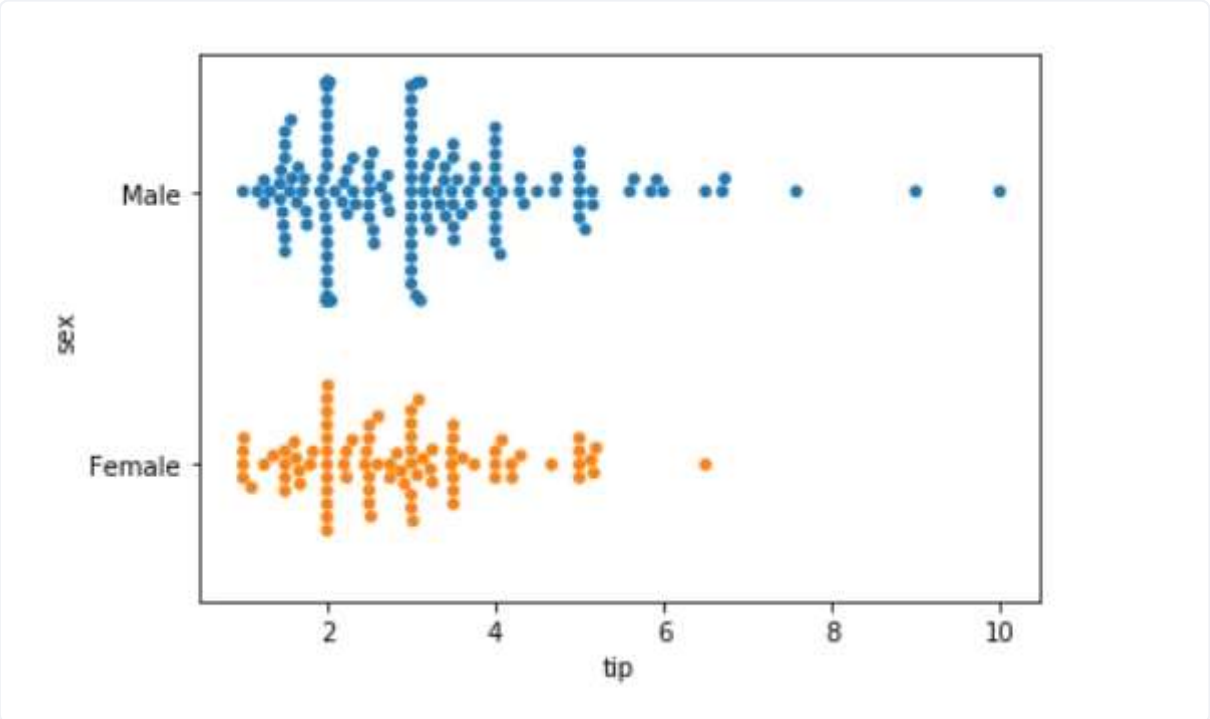
```
import seaborn
import pandas
import matplotlib.pyplot as plt


csv = seaborn.load_dataset("tips")
res = seaborn.swarmplot(x="tip", y="sex", data=csv)

plt.show()
```

In the above example, I have passed the column 'sex' as the only categorical data and have plotted against the same along the x-axis, respectively.

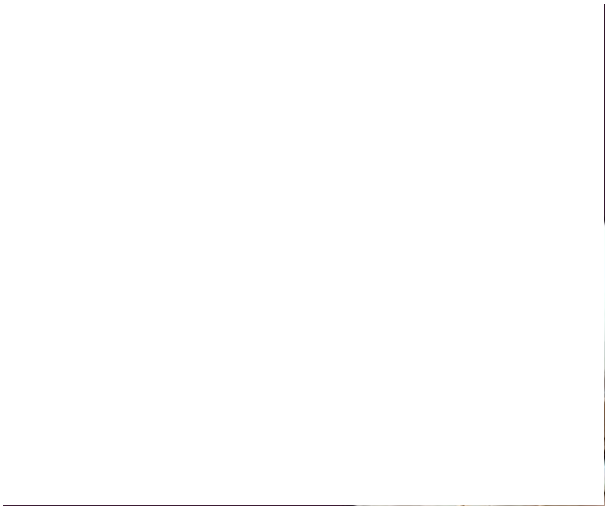
Output:



swarmplot

Categorical Distribution Plots

Categorical Distribution data basically refers to the type of data wherein the result describes the certain possibility of the random/chosen variable to belong to one of the given **possible categories**.



Python Seaborn has the following functions to represent the categorical distributed data efficiently:

seaborn.violinplot()

seaborn.boxplot()

seaborn.boxenplot()

---

1. seaborn.violinplot()

The `seaborn.violinplot()` function represents the underlying distribution of the data. It depicts and represents the distribution of data against different categorical data input.

Syntax:

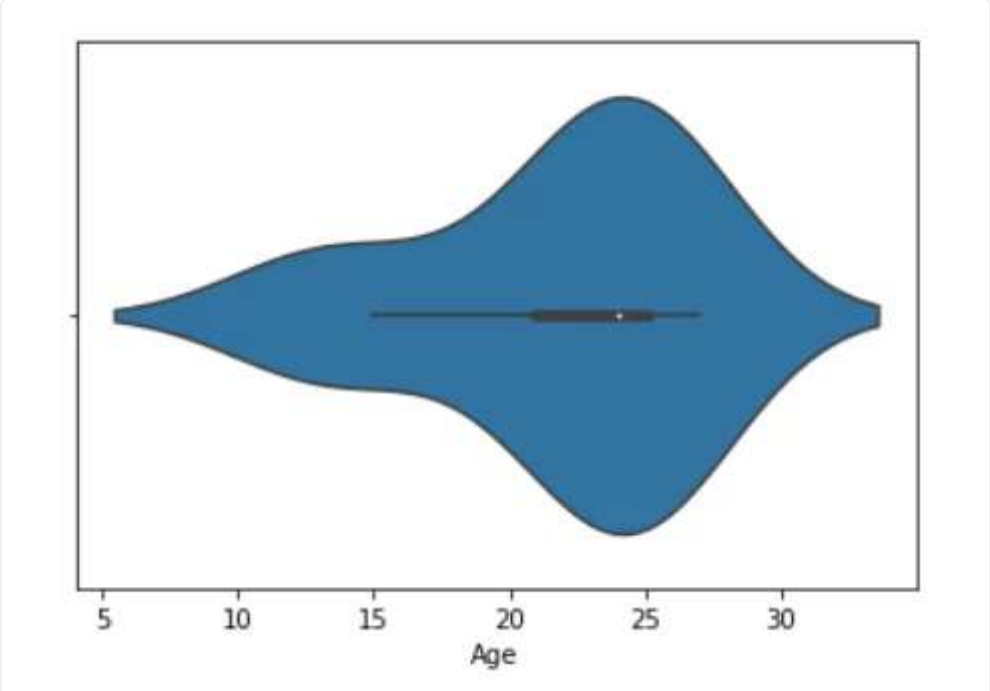
```
seaborn.violinplot(x=value, y=value, data=data)
```

Example:

```
import seaborn
import pandas
import matplotlib.pyplot as plt
csv = pandas.read_csv("C:\\Book1.csv")
res = seaborn.violinplot(x=csv['Age'])
plt.show()
```

In the above example, we have considered the distribution of data along the column-'Age', respectively.

Output:



Seaborn-violinplot

2. seaborn.boxplot()

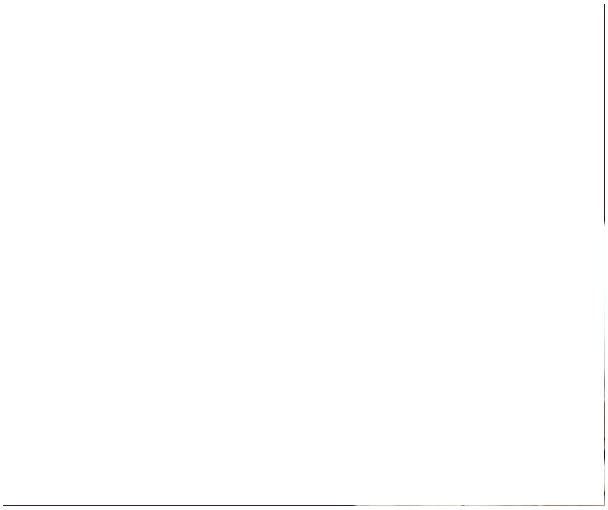
The `seaborn.boxplot()` function represents the **categorical distribution** of data and sets comparison among the different categorical data inputs.

The **'box' structure** represents the **main quartile of the data input** while the **'line' structure** represents the rest of the **distribution** of data. The **outliers** are represented by points using an **inter-quartile function**.

Syntax:

```
seaborn.boxplot(x=value, y=value, data=data)
```

Example:

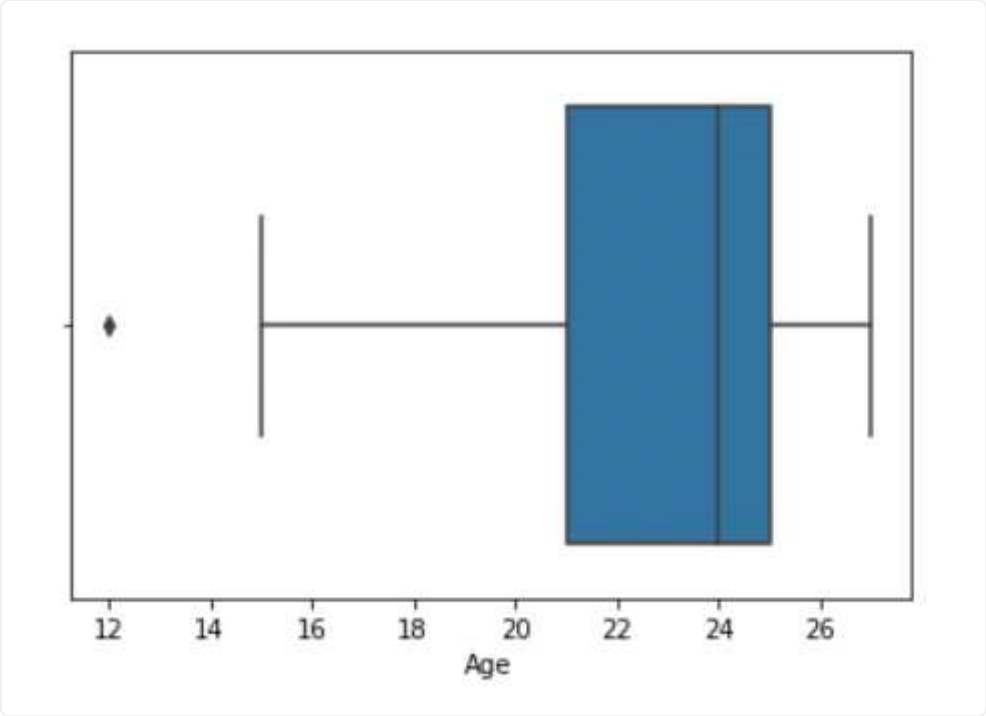


```
import seaborn
import pandas
import matplotlib.pyplot as plt
csv = pandas.read_csv("C:\\Book1.csv")
res = seaborn.boxplot(x=csv['Age'])
plt.show()
```

In the above example, we have used Book1.csv file as the input data set.

If you try to analyze the data-set, you will find the Age-12 to be an outlier type of data and the rest of the data ranging between 15-27. This is represented well by the **seaborn.boxplot()** function.

Output:

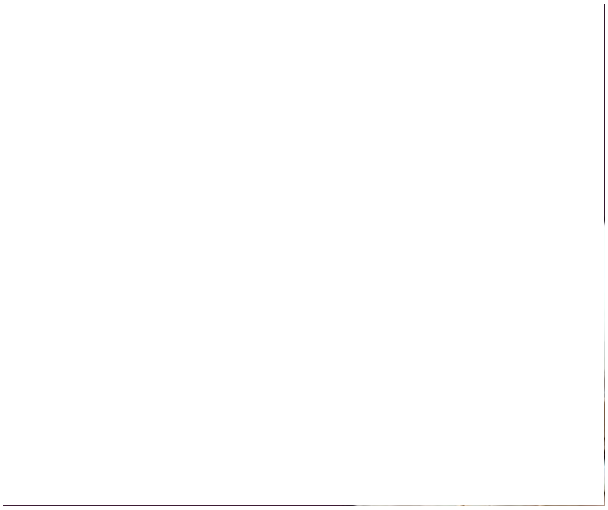


Seaborn boxplot

3. seaborn.boxenplot()

The `seaborn.boxenplot()` function is quite similar to **seaborn.boxplot()** function with a slight difference in the representation.

The **seaborn.boxenplot()** function represents the distribution of the categorical data in a way where the **large quartiles** represent the features corresponding to the actual data observations. It presents the data in a format that gives us a **detailed information in a visualized form** about the entire distribution of data.



Syntax:

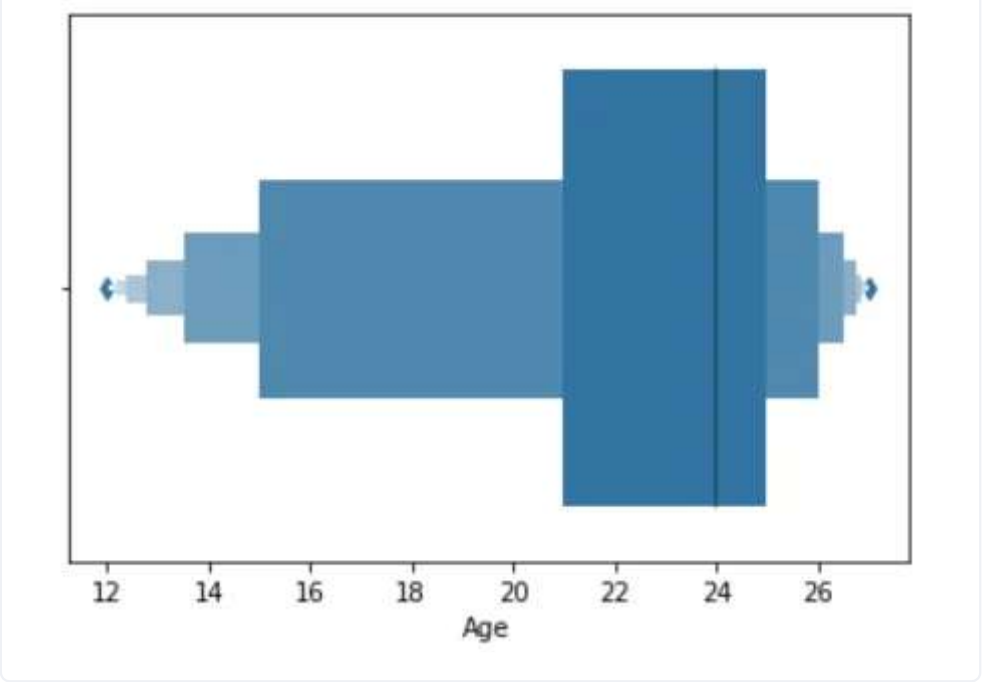
```
seaborn.boxenplot(x=value, y=value, data=data)
```

Example:

```
import seaborn
import pandas
import matplotlib.pyplot as plt
csv = pandas.read_csv("C:\\Book1.csv")
res = seaborn.boxenplot(x=csv['Age'])
plt.show()
```

If you analyze and compare the below output with the input data set, it is clearly understood that **boxenplot** represents the entire distribution of the data points ranging between 12-27, along with the distribution of the categorical data with a large quartile-box structure.

Output:



Seaborn boxenplot

Categorical estimate plots

The estimation of categorical data basically refers to the representation of certain estimation or prediction of the categorical data values to the corresponding data variable.

Python Seaborn has the following functions to be used for the estimation of categorical data:

seaborn.countplot()

seaborn.barplot()

seaborn.pointplot()

---

1. seaborn.countplot()

The `seaborn.counplot()` function is used to estimate and represent the categorical variable in terms of the frequency or count of it.

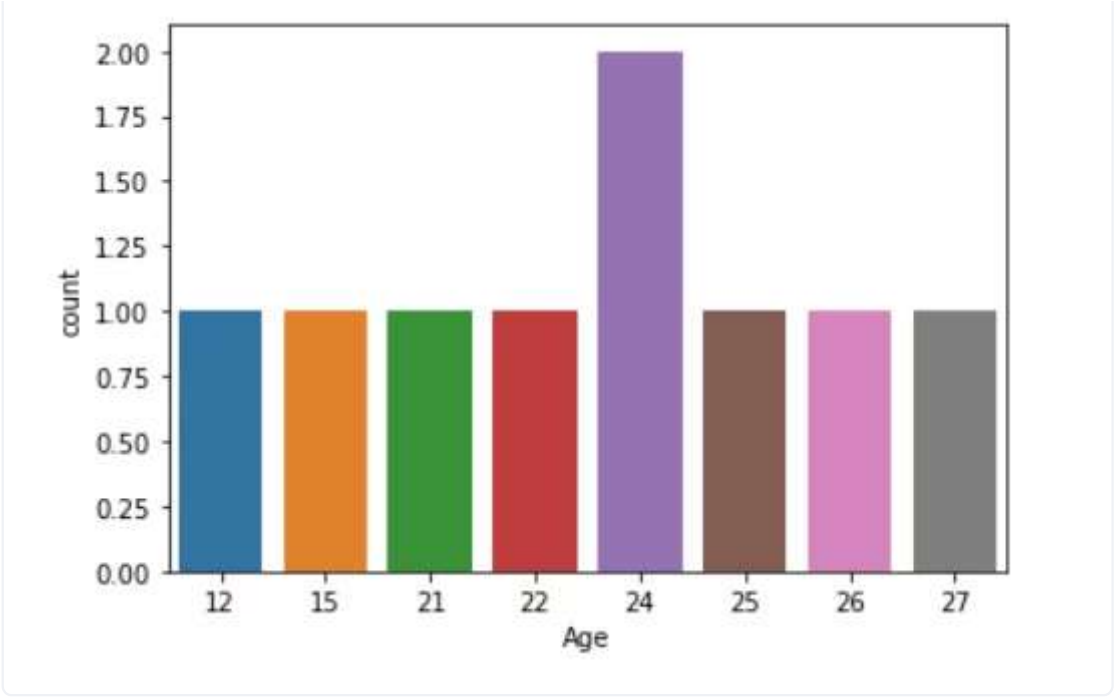
Syntax:

```
seaborn.countplot(x=value, y=value, data=data)
```

Example:

```
import seaborn
import pandas
import matplotlib.pyplot as plt
csv = pandas.read_csv("C:\\Book1.csv")
res = seaborn.countplot(x=csv['Age'])
plt.show()
```

Output:



Seaborn countplot

As seen clearly in the above image, the **countplot() function** has basically counted the frequency of the input data field and represented it along the y-axis while the data field – 'Age' being represented along the x-axis.

---

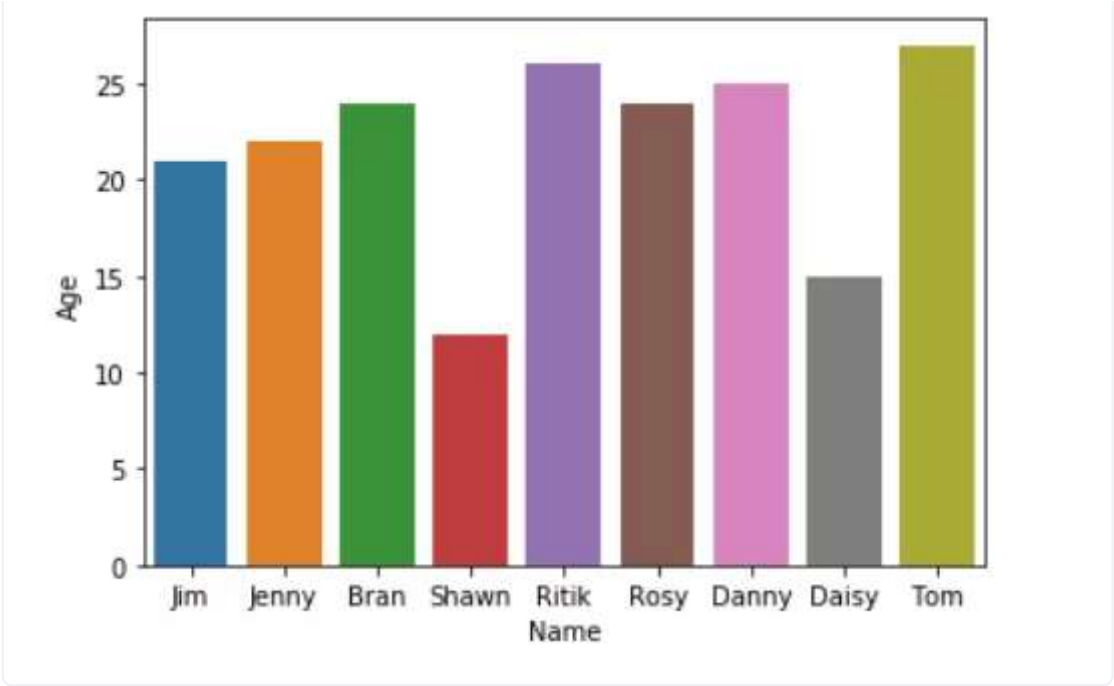
2. seaborn.barplot()

The `seaborn.barplot()` function basically represents the estimated data in the form of the central tendency of the data representation.

Example:

```
import seaborn
import pandas
import matplotlib.pyplot as plt
csv = pandas.read_csv("C:\\Book1.csv")
res = seaborn.barplot(x=csv['Name'], y=csv['Age'])
plt.show()
```

Output:



Seaborn barplot

3. seaborn.pointplot()

The `seaborn.pointplot()` function represents the estimation of the central tendency of the distribution with the help of scatter points and lines joining them.

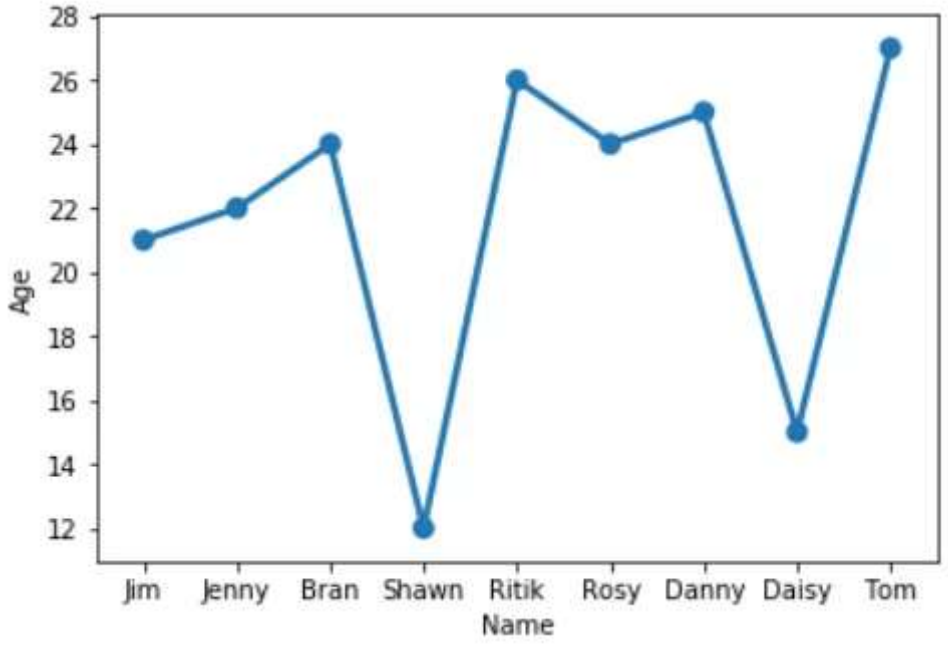
Syntax:

```
seaborn.pointplot(x=value, y=value, data=data)
```

Example:

```
import seaborn
import pandas
import matplotlib.pyplot as plt
csv = pandas.read_csv("C:\\Book1.csv")
res = seaborn.pointplot(x=csv['Name'], y=csv['Age'])
plt.show()
```

Output:



Seaborn pointplot

Customized Styles and Themes in Seaborn

Python Seaborn has in-built functions and themes to visualize the data in a better and attractive manner.

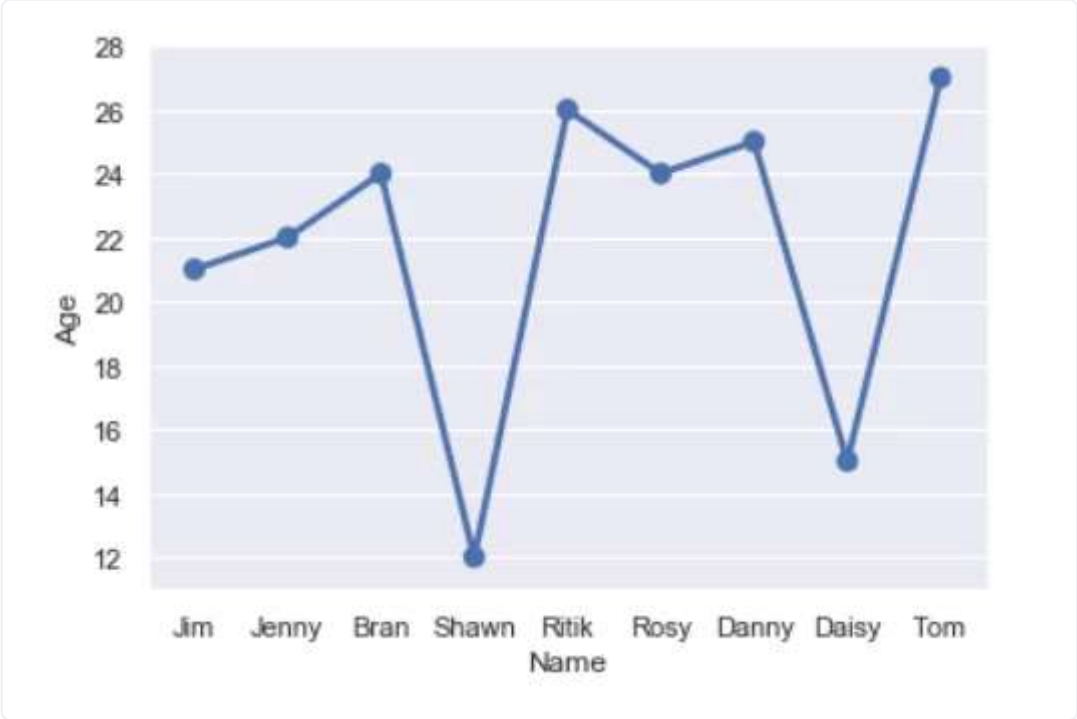
The `seaborn.set()` function is used for the **default** theme acquisition of the output visualization.

Syntax:

```
seaborn.set()
```

```
import seaborn
import pandas
import matplotlib.pyplot as plt
seaborn.set()
csv = pandas.read_csv("C:\\Book1.csv")
res = seaborn.pointplot(x=csv['Name'], y=csv['Age'])
plt.show()
```

Output:



Seaborn Style Using set()

Python Seaborn provides us with the following themes to work with and represent, visualize the data:

- **Ticks**
- **Whitegrid theme**
- **Darkgrid theme**
- **Dark**
- **White**

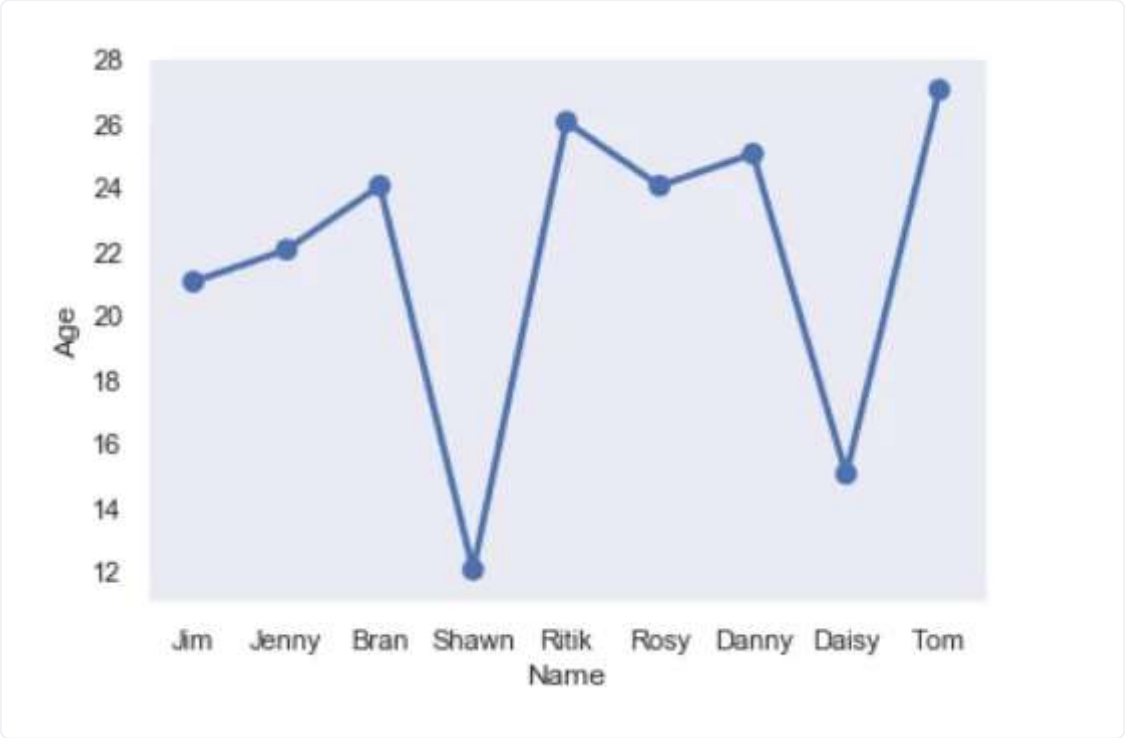
Syntax:

```
seaborn.set_style("theme-name")
```

Example: 1- `The dark theme`

```
import seaborn
import pandas
import matplotlib.pyplot as plt
seaborn.set_style("dark")
csv = pandas.read_csv("C:\\Book1.csv")
res = seaborn.pointplot(x=csv['Name'], y=csv['Age'])
plt.show()
```

Output:

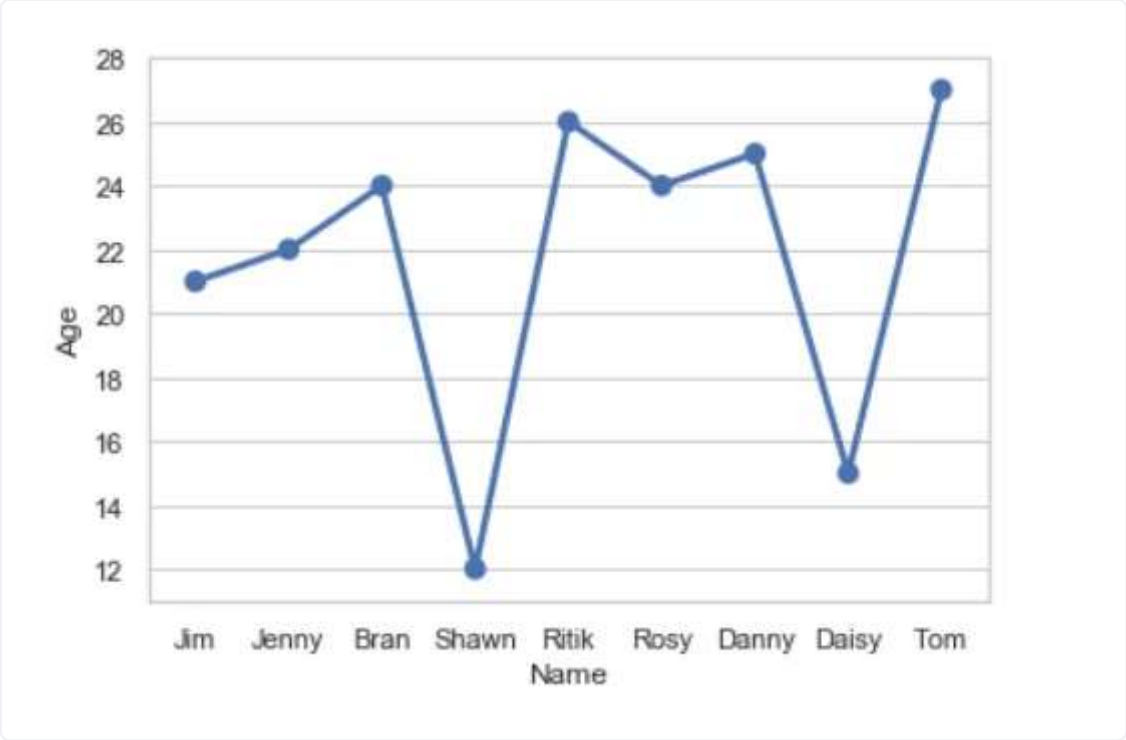


Seaborn Dark Theme

Example: 2- The whitegrid theme

```
import seaborn
import pandas
import matplotlib.pyplot as plt
seaborn.set_style("whitegrid")
csv = pandas.read_csv("C:\\Book1.csv")
res = seaborn.pointplot(x=csv['Name'], y=csv['Age'])
plt.show()
```

Output:



Seaborn White grid Theme

Multi-Plot grids in Seaborn

In order to represent the large data set with categorical values in a precise manner, we can draw **multiple plots of the sub-sets of data** to visualize it.

Syntax:

```
seaborn.FacetGird(data, col=value, col_wrap=value)
```

Example:

```
import seaborn
import pandas
import matplotlib.pyplot as plt
seaborn.set_style("whitegrid")
csv = pandas.read_csv("C:\\Book1.csv")
res = seaborn.FacetGrid(csv, col="Age", col_wrap=3)
```

```
res.map(seaborn.barplot, "Name", "Age")
plt.show()
```

The `FacetGrid class` is used to extensively represent the data with multiple plots against the sub-sets of data. It can be represented along the following dimensions:

row

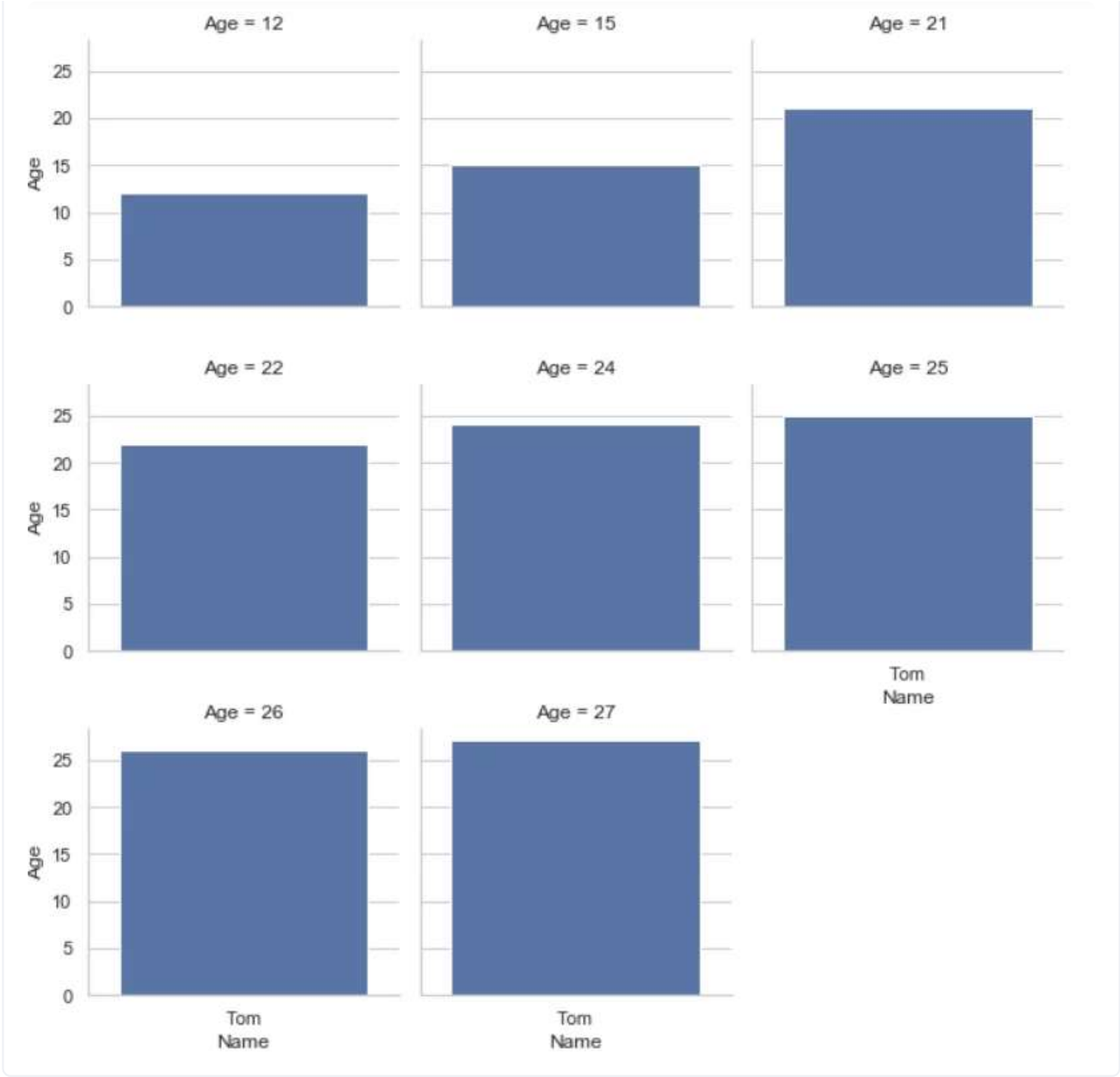
col

hue

The parameter `col_wrap` basically represents the number of rows along which the graphs need to be represented.

The `FacetGrid.map()` function is used to apply a plotting technique to every subset of the data.

Output:



Seaborn Multigrid

Plotting univariate distributions with Seaborn

Univariate distribution basically refers to the **distribution** of the data with respect to a **single random variable/data item**.

Python Seaborn module's `seaborn.distplot()` function can be used to represent the univariate distribution of data set.

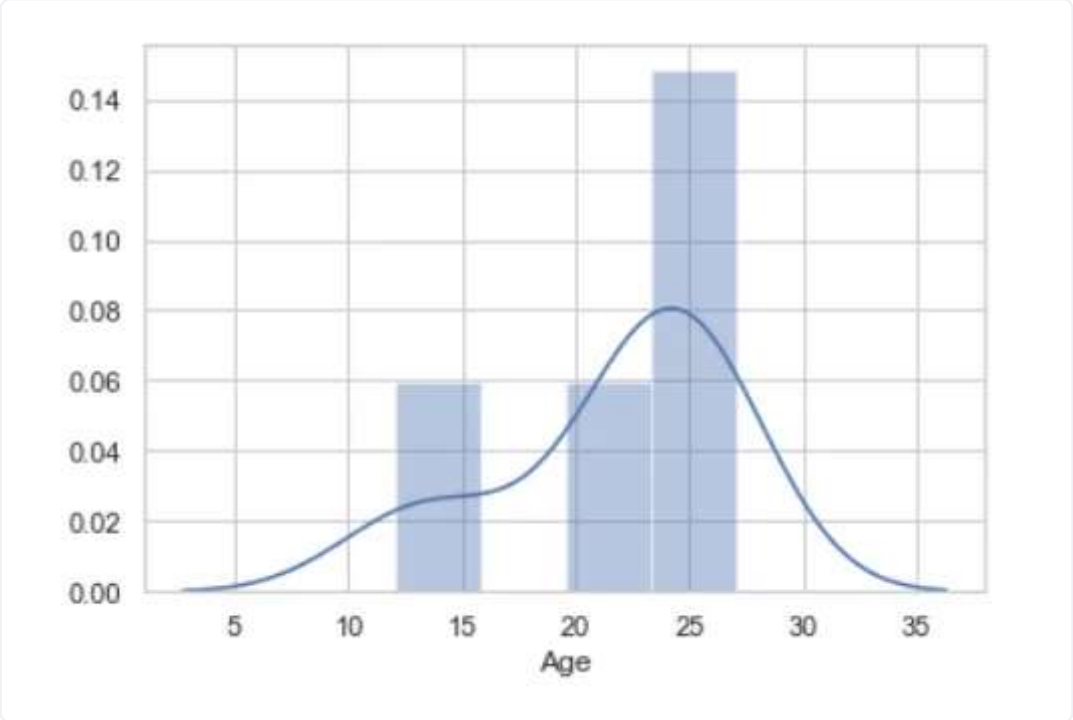
Syntax:

```
seaborn.distplot(data-column)
```

Example:

```
import seaborn
import pandas
import matplotlib.pyplot as plt
seaborn.set_style("whitegrid")
csv = pandas.read_csv("C:\\Book1.csv")
res=seaborn.distplot(csv['Age'])
plt.show()
```

Output:



Seaborn Distplot

Depicting bivariate distributions with Seaborn

Bivariate distribution refers to the visualization of data with respect to **two data columns or items of the data set**.

The `seaborn.jointplot()` can be used to depict the relationship between the two data variables.

Syntax:

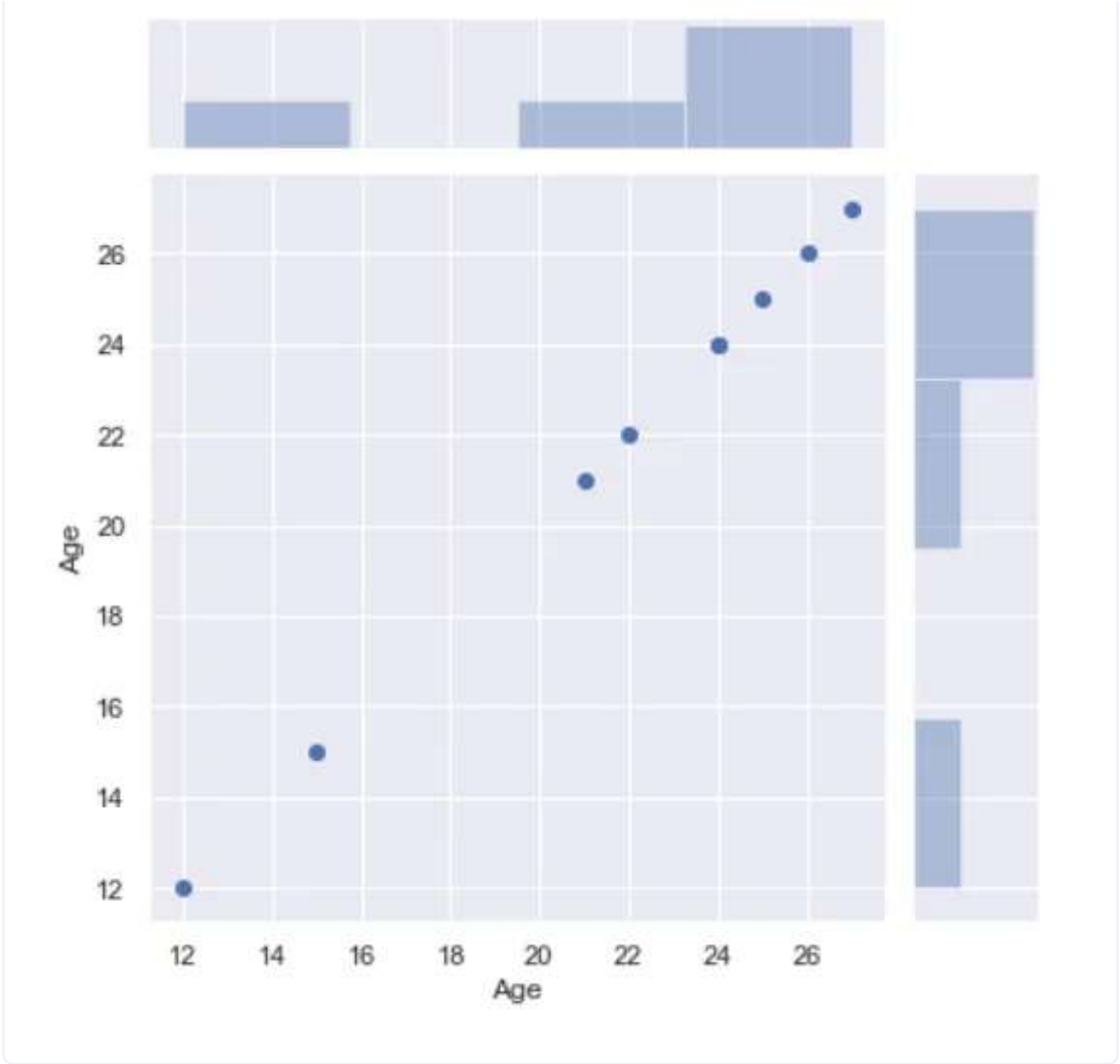
```
seaborn.jointplot(x=variable1, y=variable2)
```

Example:

```
import seaborn
import pandas
import matplotlib.pyplot as plt
seaborn.set_style("darkgrid")
csv = pandas.read_csv("C:\\Book1.csv")
res=seaborn.jointplot(x=csv['Age'], y=csv['Age'])
plt.show()
```

In the above example, We have used both the variables as 'Age' just for the sake of simplicity to depict the visualization of data.

Output:



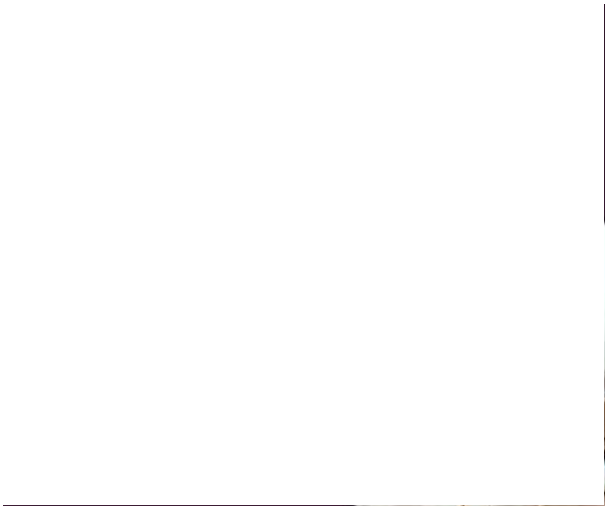
Seaborn jointplot

Conclusion

Thus, in this article, we have understood the basic functionality offered by **Python Seaborn for data visualization**.

References

- Python Seaborn-Official Documentation
- Python Seaborn tutorial-JournalDev



← Previous Post

Next Post →

Search ...

Recent Posts

An Ultimate Guide On Insider Threat Risks And Their Prevention

A beginner's tutorial to train a classifier model with unlabeled data using semi-supervised learning (SSL)

When To Use Colon (:) in Python?

Animating Data in Python – A Simple Guide

GamStop API: Worth the Wait for Open Use

Why do Traders Need to Start Learning Python?

Resize the Plots and Subplots in Matplotlib Using figsize

Diagnose Fever In Python [Easy CLI Method]

Designing State Machines using Python [A Quick Guide]

Tkinter Create Oval – A Quick Guide

Favorite Sites

JournalDev

GoLangDocs

VM-Help

LinuxForDevices

MySQLCode

CodeForGeek