Line chart in Matplotlib – Python

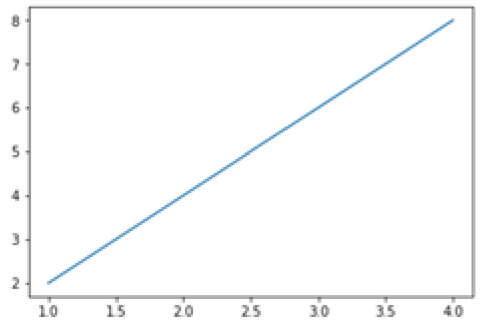
Matplotlib is a data visualization library in Python. The **pyplot**, a sublibrary of matplotlib, is a collection of functions that helps in creating a variety of charts. ***Line charts*** are used to represent the relation between two data X and Y on a different axis. Here we will see some of the examples of a line chart in Python :

Simple line plots

First import Matplotlib.pyplot library for plotting functions. Also, import the Numpy library as per requirement. Then define data values x and y.

```
# importing the required libraries

import matplotlib.pyplot as plt

import numpy as np



# define data values

x = np.array([1, 2, 3, 4])  # X-axis points

y = x*2 # Y-axis points



plt.plot(x, y)  # Plot the chart

plt.show()  # display
```

Output:

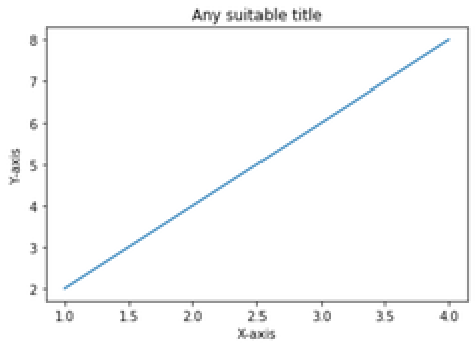


Simple line plot between X and Y data

we can see in the above output image that there is no label on the x-axis and y-axis. Since labeling is necessary for understanding the chart dimensions. In the following example, we will see how to add labels, Ident in the charts

```
import matplotlib.pyplot as plt

import numpy as np




# Define X and Y variable data

x = np.array([1, 2, 3, 4])

y = x*2


plt.plot(x, y)

plt.xlabel("X-axis")  # add X-axis label

plt.ylabel("Y-axis")  # add Y-axis label

plt.title("Any suitable title")  # add title
plt.show()
```

Output:



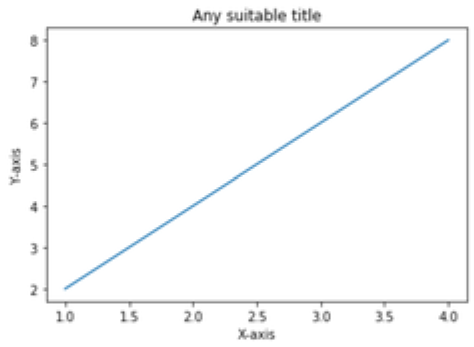
Simple line plot with labels and title

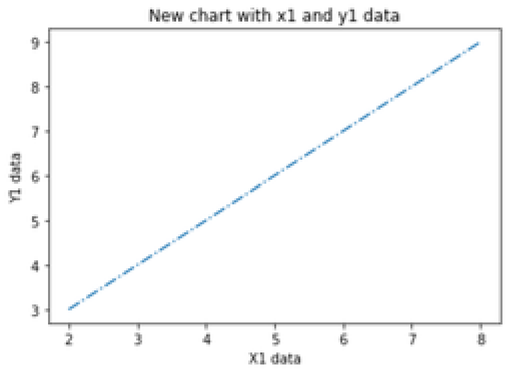
Multiple charts

We can display more than one chart in the same container by using pyplot.figure() function. This will help us in comparing the different charts and also control the look and feel of charts .

```
import matplotlib.pyplot as plt

import numpy as np




x = np.array([1, 2, 3, 4])

y = x*2



plt.plot(x, y)

plt.xlabel("X-axis")

plt.ylabel("Y-axis")
plt.title("Any suitable title")

plt.show()  # show first chart



# The figure() function helps in creating a

# new figure that can hold a new chart in it.

plt.figure()

x1 = [2, 4, 6, 8]

y1 = [3, 5, 7, 9]

plt.plot(x1, y1, '-.')



# Show another chart with '-' dotted line

plt.show()
```

Output:



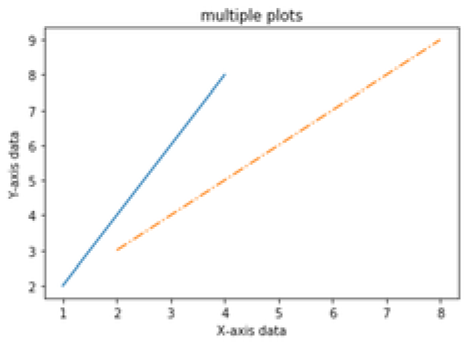


Multiple plots on the same axis

Here we will see how to add 2 plots within the same axis.

```
import matplotlib.pyplot as plt

import numpy as np



x = np.array([1, 2, 3, 4])

y = x*2



# first plot with X and Y data

plt.plot(x, y)



x1 = [2, 4, 6, 8]

y1 = [3, 5, 7, 9]



# second plot with x1 and y1 data

plt.plot(x1, y1, '-.')



plt.xlabel("X-axis data")

plt.ylabel("Y-axis data")

plt.title('multiple plots')

plt.show()
```

Output:

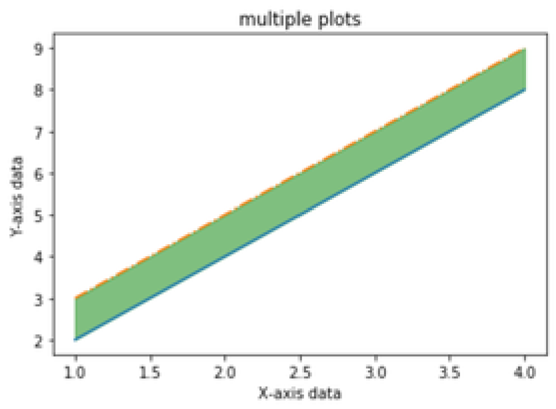


Fill the area between two plots

Using the pyplot.fill_between() function we can fill in the region between two line plots in the same graph. This will help us in understanding the margin of data between two line plots based on certain conditions.

```
import matplotlib.pyplot as plt

import numpy as np



x = np.array([1, 2, 3, 4])

y = x*2



plt.plot(x, y)



x1 = [2, 4, 6, 8]

y1 = [3, 5, 7, 9]



plt.plot(x, y1, '-.')

plt.xlabel("X-axis data")
plt.ylabel("Y-axis data")

plt.title('multiple plots')



plt.fill_between(x, y, y1, color='green', alpha=0.5)

plt.show()
```

Output:



Fill the area between Y and Y1 data corresponding to X-axis data

---

Bar Plot in Matplotlib

- Last Updated : 04 Mar, 2021

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. The bar plots can be plotted horizontally or vertically. A bar chart describes the comparisons between the discrete categories. One of the axis of the plot represents the specific categories being compared, while the other axis represents the measured values corresponding to those categories.

Creating a bar plot

The **matplotlib** API in Python provides the bar() function which can be used in MATLAB style use or as an object-oriented API. The syntax of the bar() function to be used with the axes is as follows:-

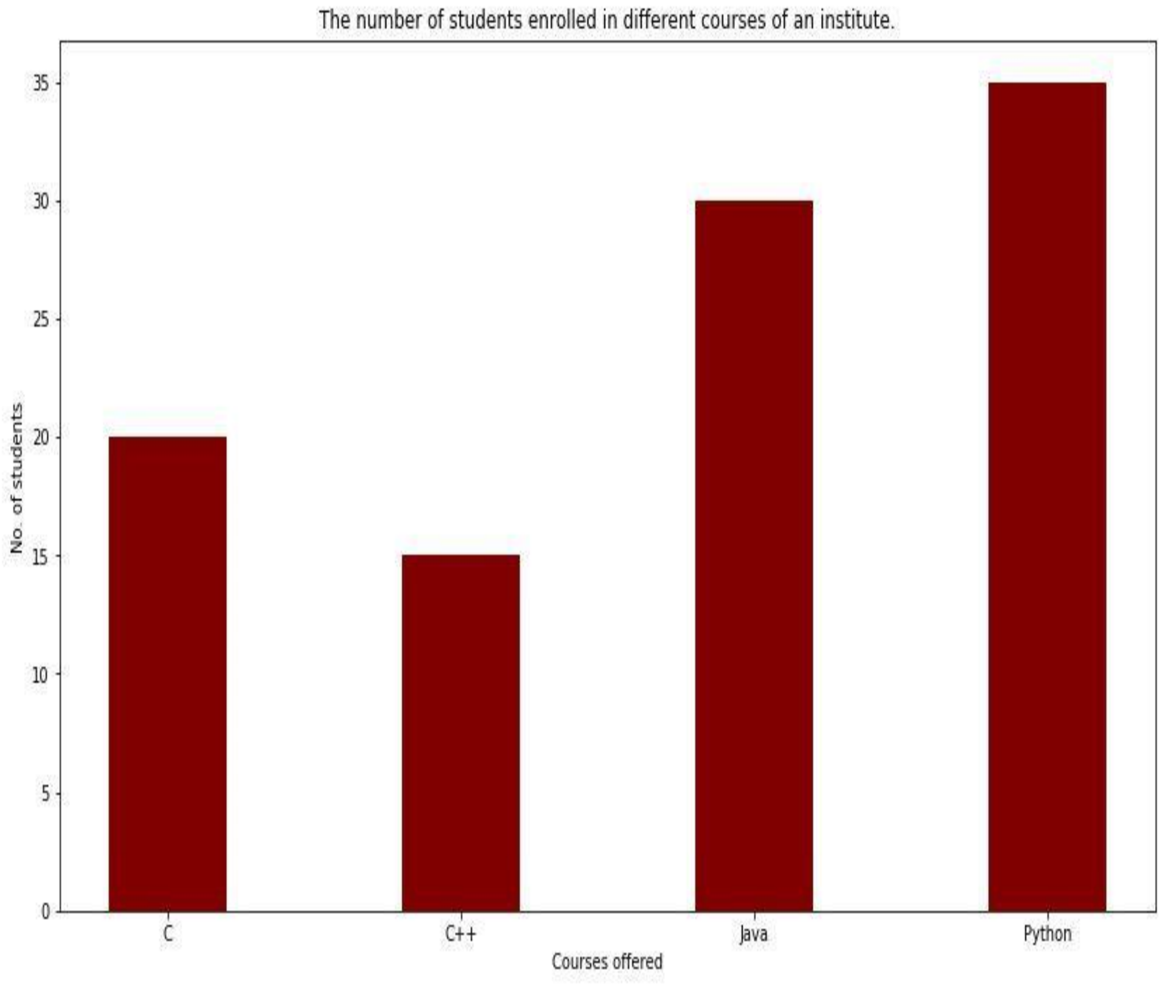
```
plt.bar(x, height, width, bottom, align)
```

The function creates a bar plot bounded with a rectangle depending on the given parameters. Following is a simple example of the bar plot, which represents the number of students enrolled in different courses of an institute.

```
import numpy as np

import matplotlib.pyplot as plt




# creating the dataset

data = {'C':20, 'C++':15, 'Java':30,

        'Python':35}

courses = list(data.keys())

values = list(data.values())



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



# creating the bar plot

plt.bar(courses, values, color ='maroon',

        width = 0.4)
```

```
plt.xlabel("Courses offered")

plt.ylabel("No. of students enrolled")

plt.title("Students enrolled in different courses")

plt.show()
```

Output-



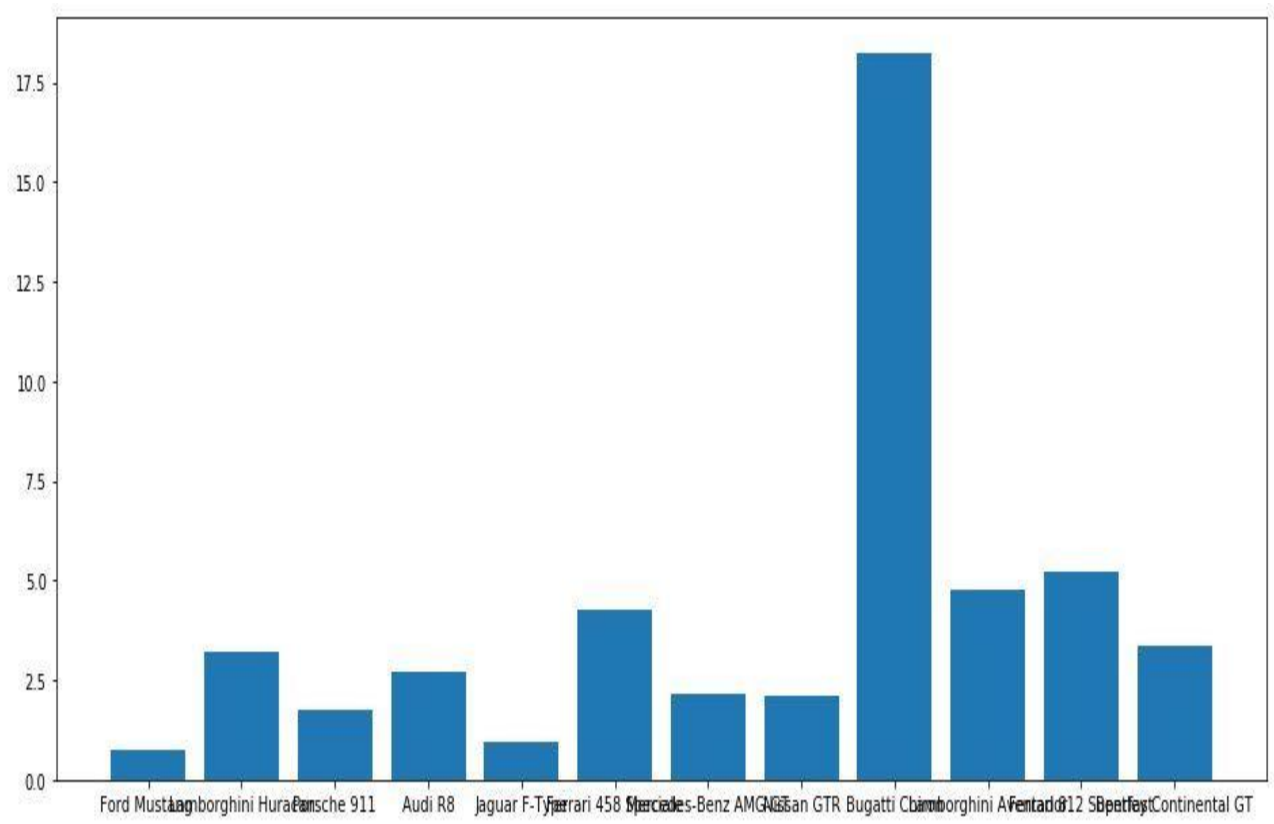
Here plt.bar(courses, values, color='maroon') is used to specify that the bar chart is to be plotted by using the courses column as the X-axis, and the values as the Y-axis. The color attribute is used to set the color of the bars(maroon in this case).plt.xlabel("Courses offered") and plt.ylabel("students enrolled") are used to label the corresponding axes.plt.title() is used to make a title for the graph.plt.show() is used to show the graph as output using the previous commands.

Customizing the bar plot

```
import pandas as pd

from matplotlib import pyplot as plt



# Read CSV into pandas

data = pd.read_csv(r"cars.csv")

data.head()

df = pd.DataFrame(data)



name = df['car'].head(12)

price = df['price'].head(12)



# Figure Size

fig = plt.figure(figsize =(10, 7))
```

```
# Horizontal Bar Plot

plt.bar(name[0:10], price[0:10])



# Show Plot

plt.show()
```

Output:



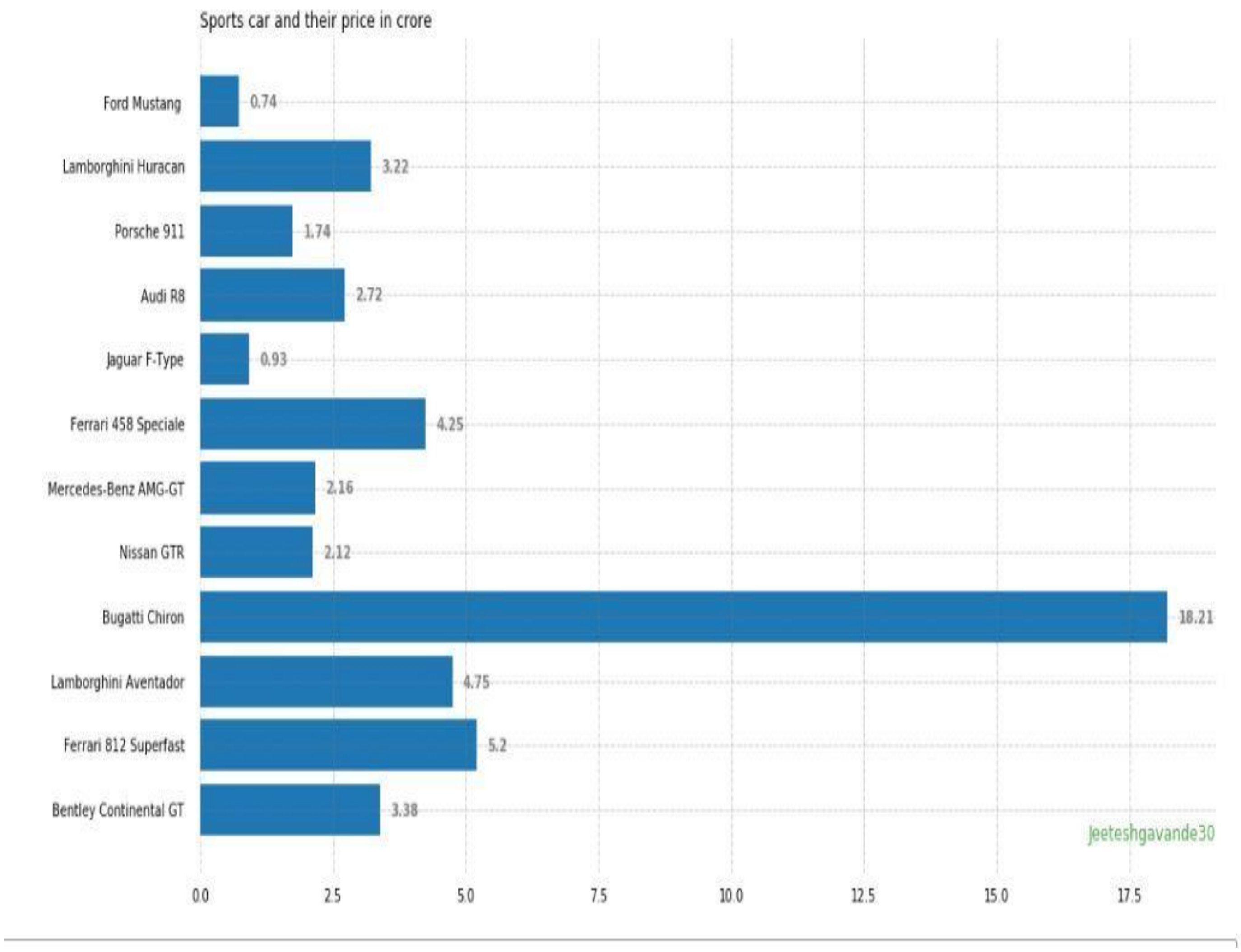
It is observed in the above bar graph that the X-axis ticks are overlapping each other thus it cannot be seen properly. Thus by rotating the X-axis ticks, it can be visible clearly. That is why customization in bar graphs is required.

```
import pandas as pd

from matplotlib import pyplot as plt



# Read CSV into pandas

data = pd.read_csv(r"cars.csv")

data.head()

df = pd.DataFrame(data)



name = df['car'].head(12)

price = df['price'].head(12)



# Figure Size

fig, ax = plt.subplots(figsize =(16, 9))



# Horizontal Bar Plot

ax.barh(name, price)



# Remove axes splines
```

```
for s in ['top', 'bottom', 'left', 'right']:

    ax.spines[s].set_visible(False)



# Remove x, y Ticks

ax.xaxis.set_ticks_position('none')

ax.yaxis.set_ticks_position('none')



# Add padding between axes and labels

ax.xaxis.set_tick_params(pad = 5)

ax.yaxis.set_tick_params(pad = 10)



# Add x, y gridlines

ax.grid(b = True, color ='grey',

        linestyle ='-.', linewidth = 0.5,

        alpha = 0.2)



# Show top values

ax.invert_yaxis()
```

```
# Add annotation to bars
for i in ax.patches:
    plt.text(i.get_width()+0.2, i.get_y()+0.5,
             str(round((i.get_width()), 2)),
             fontsize = 10, fontweight ='bold',
             color ='grey')


# Add Plot Title
ax.set_title('Sports car and their price in crore',
             loc ='left', )


# Add Text watermark
fig.text(0.9, 0.15, 'Jeeteshgavande30', fontsize = 12,
         color ='grey', ha ='right', va ='bottom',
         alpha = 0.7)


# Show Plot
plt.show()
```

Output:



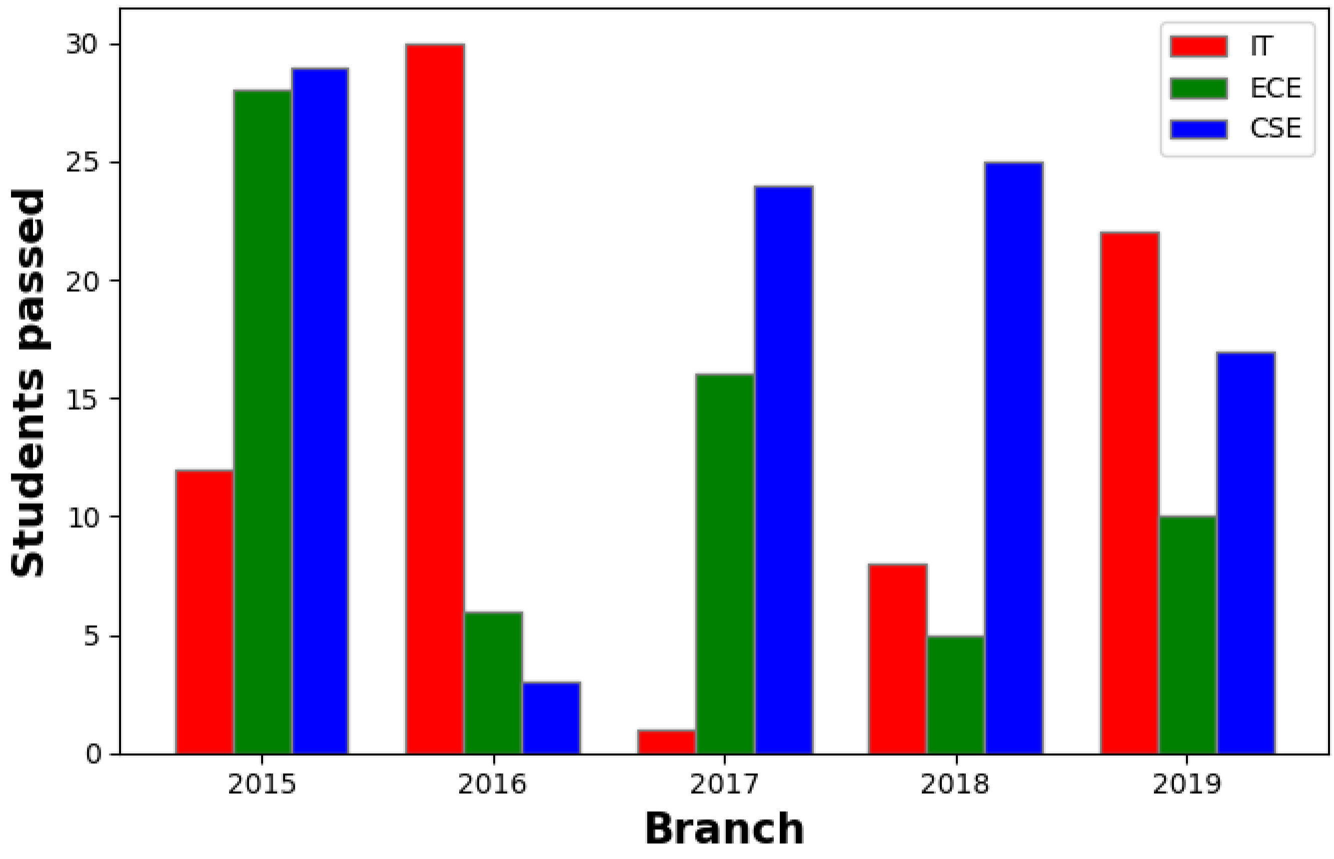
There are many more Customizations available for bar plots.

Multiple bar plots

Multiple bar plots are used when comparison among the data set is to be done when one variable is changing. We can easily convert it as a stacked area bar chart, where each subgroup is displayed by one on top of the others. It can be plotted by varying the thickness and position of the bars. Following bar plot shows the number of students passed in the engineering branch:

```
import numpy as np

import matplotlib.pyplot as plt



# set width of bar

barWidth = 0.25

fig = plt.subplots(figsize =(12, 8))



# set height of bar

IT = [12, 30, 1, 8, 22]

ECE = [28, 6, 16, 5, 10]

CSE = [29, 3, 24, 25, 17]



# Set position of bar on X axis

br1 = np.arange(len(IT))

br2 = [x + barWidth for x in br1]

br3 = [x + barWidth for x in br2]



# Make the plot

plt.bar(br1, IT, color ='r', width = barWidth,
        edgecolor ='grey', label ='IT')

plt.bar(br2, ECE, color ='g', width = barWidth,

        edgecolor ='grey', label ='ECE')

plt.bar(br3, CSE, color ='b', width = barWidth,

        edgecolor ='grey', label ='CSE')



# Adding Xticks

plt.xlabel('Branch', fontweight ='bold', fontsize = 15)

plt.ylabel('Students passed', fontweight ='bold', fontsize = 15)

plt.xticks([r + barWidth for r in range(len(IT))],

        ['2015', '2016', '2017', '2018', '2019'])



plt.legend()

plt.show()
```

Output:



Stacked bar plot

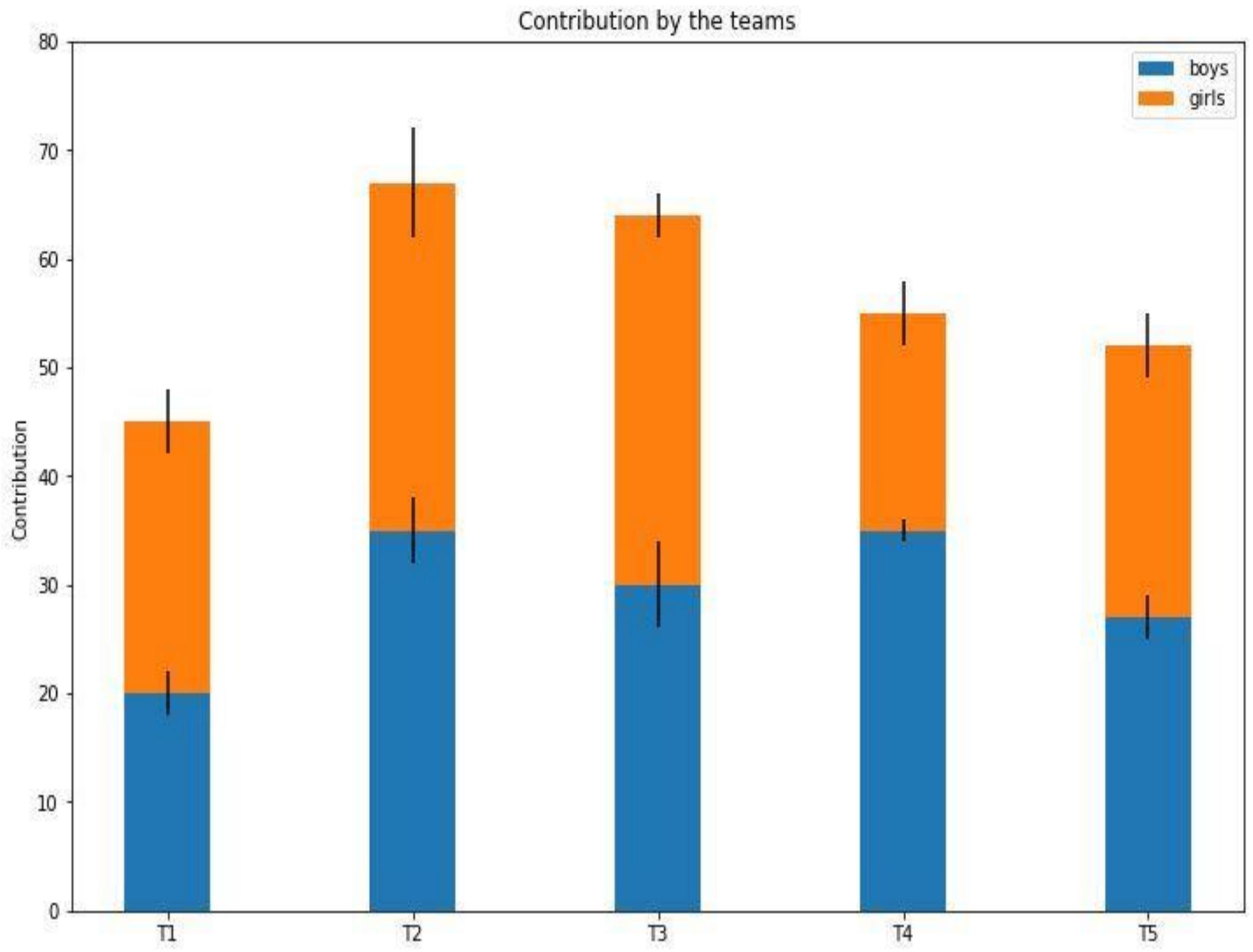
Stacked bar plots represent different groups on top of one another. The height of the bar depends on the resulting height of the combination of the results of the groups. It goes from the bottom to the value instead of going from zero to value. The following bar plot represents the contribution of boys and girls in the team.

- Python3

```
import numpy as np

import matplotlib.pyplot as plt
```

```
N = 5



boys = (20, 35, 30, 35, 27)

girls = (25, 32, 34, 20, 25)

boyStd = (2, 3, 4, 1, 2)

girlStd = (3, 5, 2, 3, 3)

ind = np.arange(N)

width = 0.35



fig = plt.subplots(figsize =(10, 7))

p1 = plt.bar(ind, boys, width, yerr = boyStd)

p2 = plt.bar(ind, girls, width,

             bottom = boys, yerr = girlStd)



plt.ylabel('Contribution')

plt.title('Contribution by the teams')

plt.xticks(ind, ('T1', 'T2', 'T3', 'T4', 'T5'))

plt.yticks(np.arange(0, 81, 10))
plt.legend((p1[0], p2[0]), ('boys', 'girls'))



plt.show()
```

Output-



Plotting Histogram in Python using Matplotlib

- Last Updated : 27 Apr, 2020

A histogram is basically used to represent data provided in a form of some groups.It is accurate method for the graphical representation of numerical data distribution.It is a type of bar plot where X-axis represents the bin ranges while Y-axis gives information about frequency.

Creating a Histogram

To create a histogram the first step is to create bin of the ranges, then distribute the whole range of the values into a series of intervals, and the count the values which fall into each of the intervals.Bins are clearly identified as consecutive, non-overlapping intervals of variables.The `matplotlib.pyplot.hist()` function is used to compute and create histogram of x.
The following table shows the parameters accepted by `matplotlib.pyplot.hist()` function :

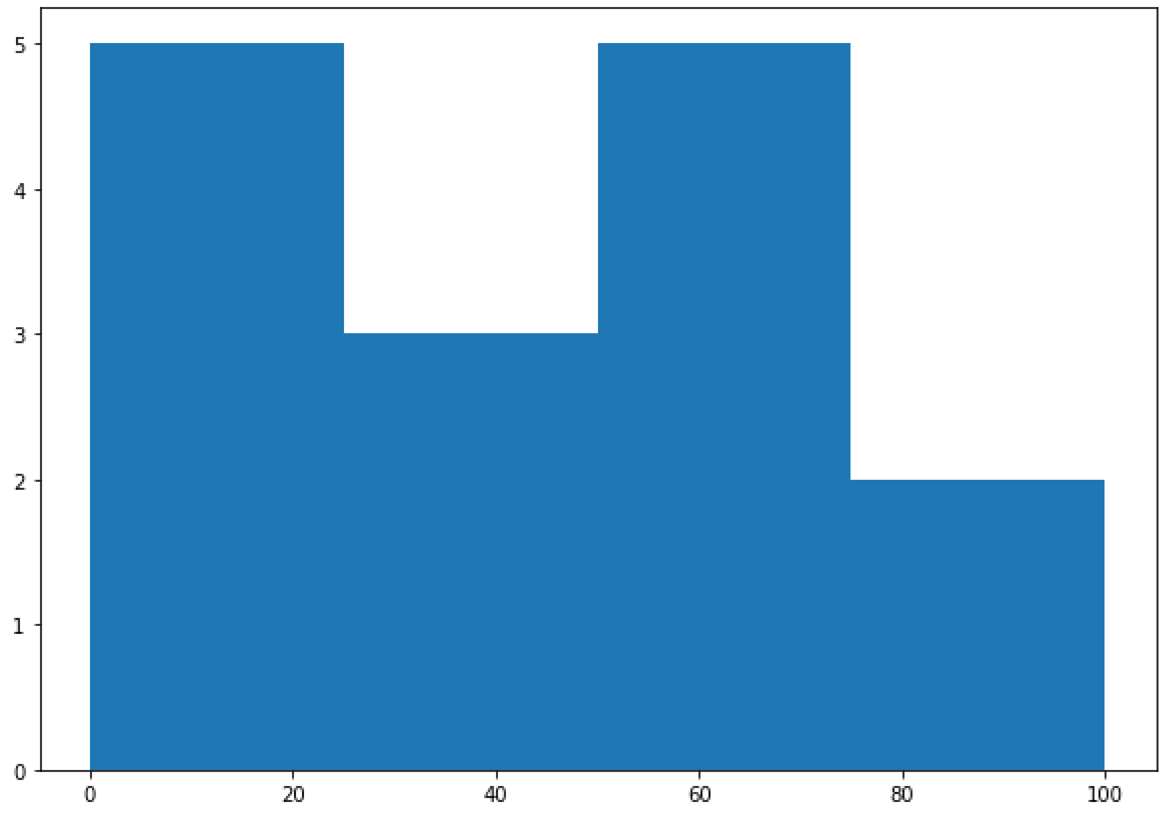
Attribute	parameter
x	array or sequence of array
bins	optional parameter contains integer or sequence or strings
density	optional parameter contains boolean values
range	optional parameter represents upper and lower range of bins
histtype	optional parameter used to creae type of histogram [bar, barstacked, step, stepfilled], default is “bar”
align	optional parameter controls the plotting of histogram [left, right, mid]
weights	optional parameter contains array of weights having same dimensions as x
bottom	location of the basline of each bin
rwidth	optional parameter which is relative width of the bars with respect to bin width
color	optional parameter used to set color or sequence of color specs
label	optional parameter string or sequence of string to match with multiple datasets

Attribute	parameter
log	optional parameter used to set histogram axis on log scale

Let's create a basic histogram of some random values.Below code creates a simple histogram of some random values:

```
from matplotlib import pyplot as plt

import numpy as np




# Creating dataset

a = np.array([22, 87, 5, 43, 56,

              73, 55, 54, 11,

              20, 51, 5, 79, 31,

              27])


# Creating histogram

fig, ax = plt.subplots(figsize =(10, 7))

ax.hist(a, bins = [0, 25, 50, 75, 100])


# Show plot

plt.show()
```

Output :



Customization of Histogram

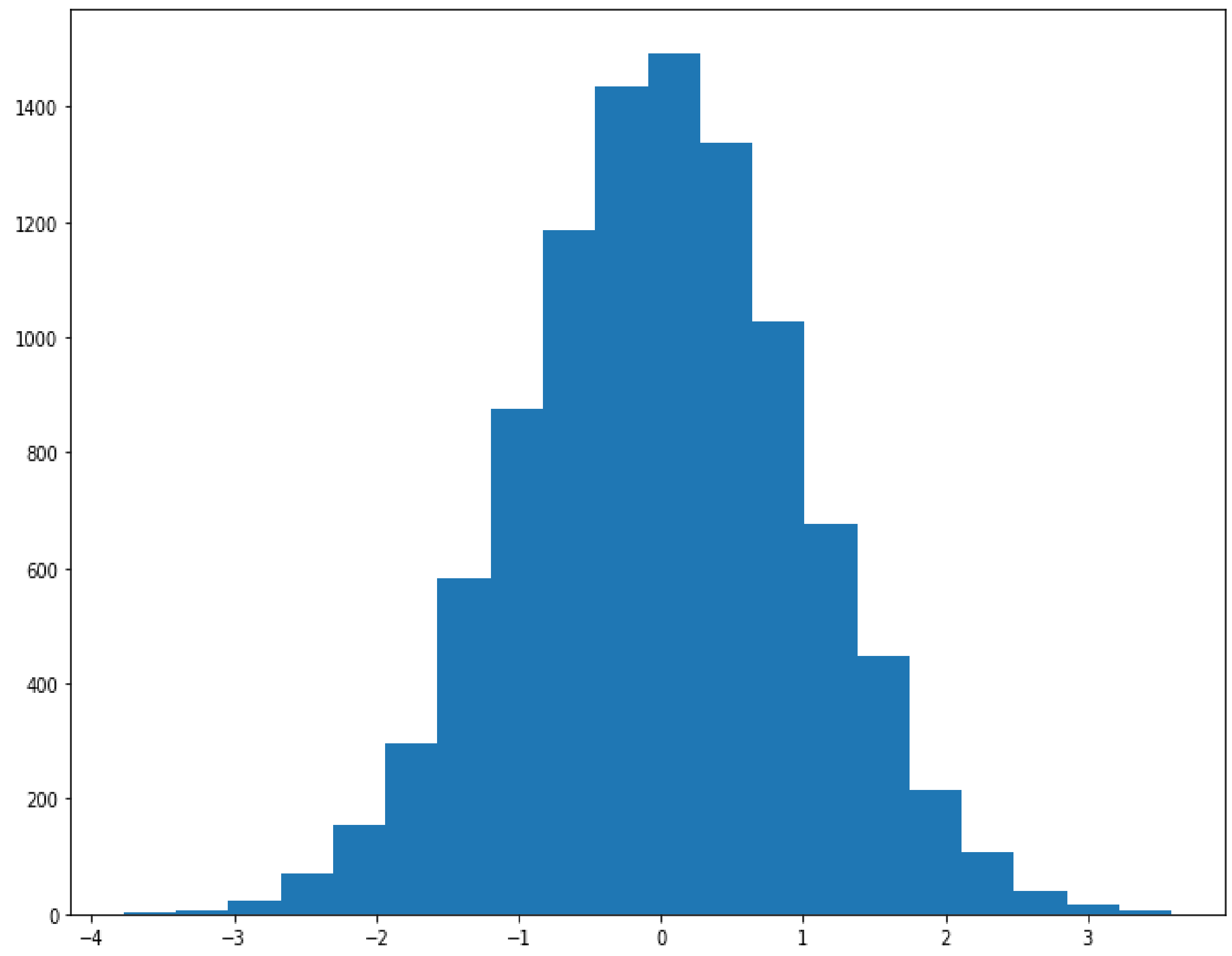
Matplotlib provides a range of different methods to customize histogram. `matplotlib.pyplot.hist()` function itself provides many attributes with the help of which we can modify a histogram.The `hist()` function provide a `patches` object which gives access to the properties of the created objects, using this we can modify the plot according to our will.

Example 1:

```
import matplotlib.pyplot as plt

import numpy as np

from matplotlib import colors

from matplotlib.ticker import PercentFormatter
```

```
# Creating dataset

np.random.seed(23685752)

N_points = 10000

n_bins = 20



# Creating distribution

x = np.random.randn(N_points)

y = .8 ** x + np.random.randn(10000) + 25



# Creating histogram

fig, axs = plt.subplots(1, 1,

                        figsize =(10, 7),

                        tight_layout = True)



axs.hist(x, bins = n_bins)



# Show plot

plt.show()
```

Output :



Example 2: The code below modifies the above histogram for a better view and accurate readings.

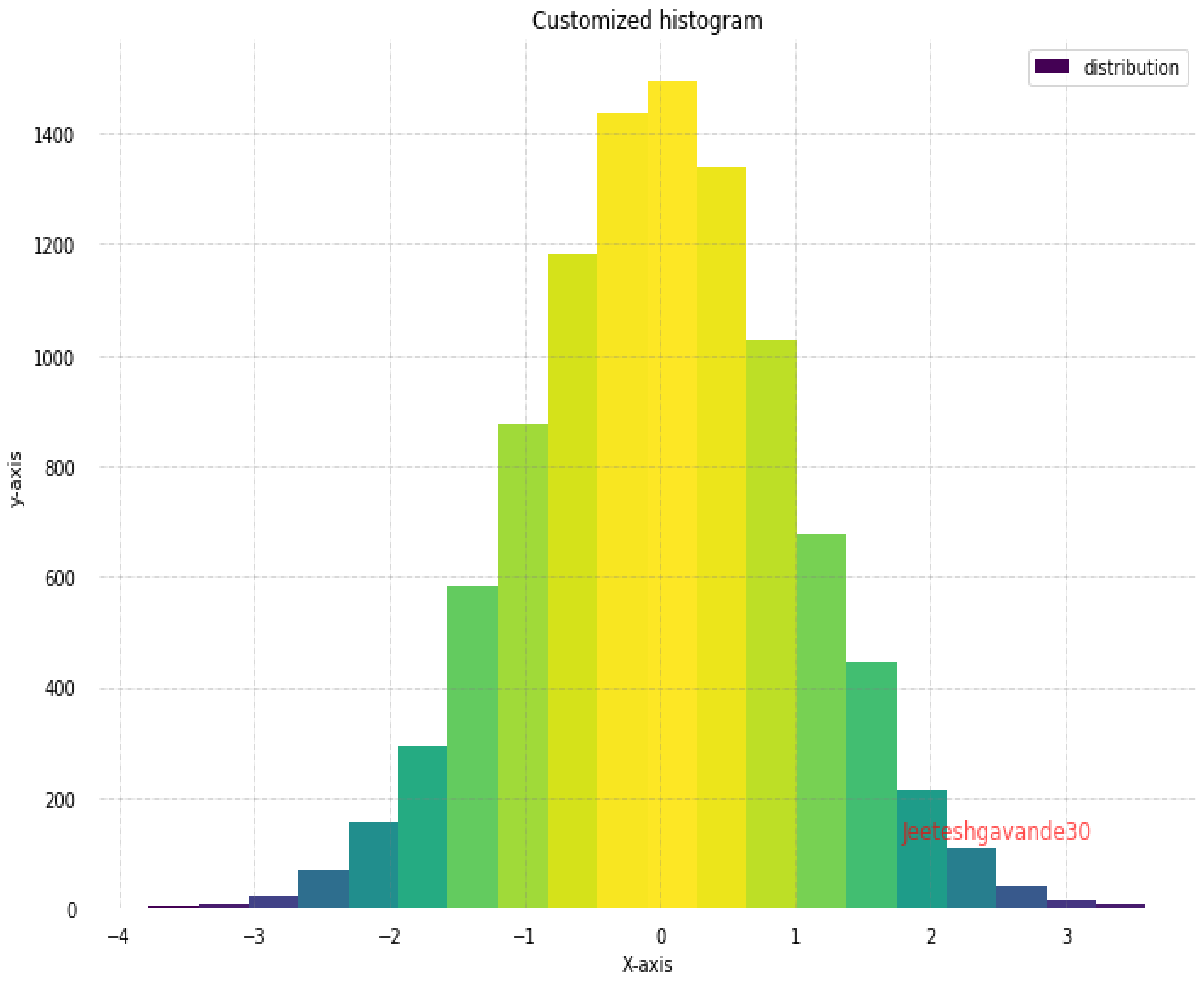
```
import matplotlib.pyplot as plt

import numpy as np

from matplotlib import colors

from matplotlib.ticker import PercentFormatter
```

```
# Creating dataset

np.random.seed(23685752)

N_points = 10000

n_bins = 20


# Creating distribution

x = np.random.randn(N_points)

y = .8 ** x + np.random.randn(10000) + 25

legend = ['distribution']


# Creating histogram

fig, axs = plt.subplots(1, 1,

                        figsize =(10, 7),

                        tight_layout = True)



# Remove axes splines

for s in ['top', 'bottom', 'left', 'right']:

    axs.spines[s].set_visible(False)
```

```
# Remove x, y ticks

axs.xaxis.set_ticks_position('none')

axs.yaxis.set_ticks_position('none')



# Add padding between axes and labels

axs.xaxis.set_tick_params(pad = 5)

axs.yaxis.set_tick_params(pad = 10)



# Add x, y gridlines

axs.grid(b = True, color ='grey',

        linestyle ='-.', linewidth = 0.5,

        alpha = 0.6)



# Add Text watermark

fig.text(0.9, 0.15, 'Jeeteshgavande30',

         fontsize = 12,

         color ='red',

         ha ='right',

         va ='bottom',
         alpha = 0.7)



# Creating histogram

N, bins, patches = axs.hist(x, bins = n_bins)



# Setting color

fracs = ((N**(1 / 5)) / N.max())

norm = colors.Normalize(fracs.min(), fracs.max())



for thisfrac, thispatch in zip(fracs, patches):

    color = plt.cm.viridis(norm(thisfrac))

    thispatch.set_facecolor(color)



# Adding extra features

plt.xlabel("X-axis")

plt.ylabel("y-axis")

plt.legend(legend)

plt.title('Customized histogram')



# Show plot
```

```
plt.show()
```

Output :


Customized histogram
distribution
y-axis
X-axis
Jeeteshgavande30