Matplotlib

Data Science using Python

Avinash A S

Module 1: Introduction to MatplotLib

1. **Matplotlib**

As a machine learning engineer or data scientist, data analysis is a part of model development. Looking numeric data, you can't get correct insights. So the best way to plot a graph using that data and take a decision for further process. Data visualization is a process take after the data cleaning.

Matplotlib is a 2D and 3D graph plotting python library. It also supports to create animations and images. This is an advanced part of **python matplotlib**.

Using python matplotlib, you can draw a different graph like below:

- Line plot
- Histogram
- Bar Chart
- Pie Chart
- Scatterplots
- Area Plot
- Error charts
- Power Spectra

2. **Install pandas using pip**

```
pip install matplotlib
```

3. **Importing Matplotlib**

```
from matplotlib import pyplot as plt
# or
import matplotlib.pyplot as plt
```

Matplotlib Line Plot

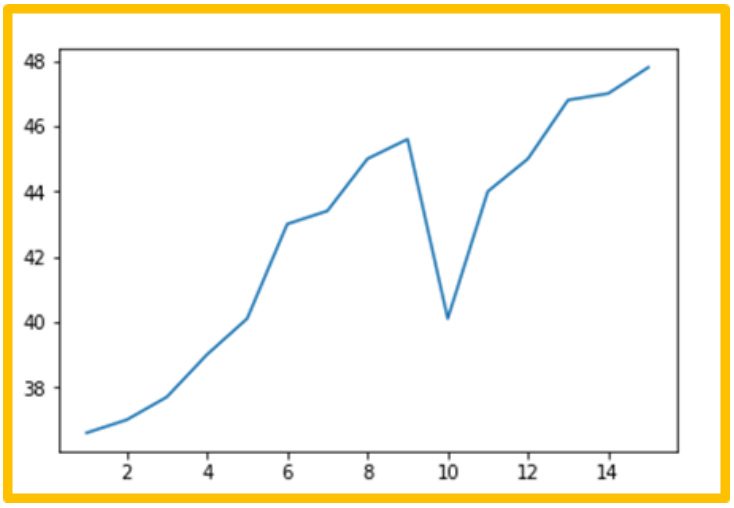
Import Dataset of 15 days Delhi temperature record. The dataset in the form of list data type, you can use NumPy array, tuple, etc.

```
Import matplotlib.pyplot as plt

days = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
temperature = [36.6, 37, 37.7,39,40.1,43,43.4,45,45.6,40.1,44,45,46.8,47,47.8]
```

Plot the line using plt.plot() method and show it using plt.show() method. Here, give a parameter x as a days and y as a temperature to plt.plot()

```
plt.plot(days, temperature)
plt.show()
```

Output >>>

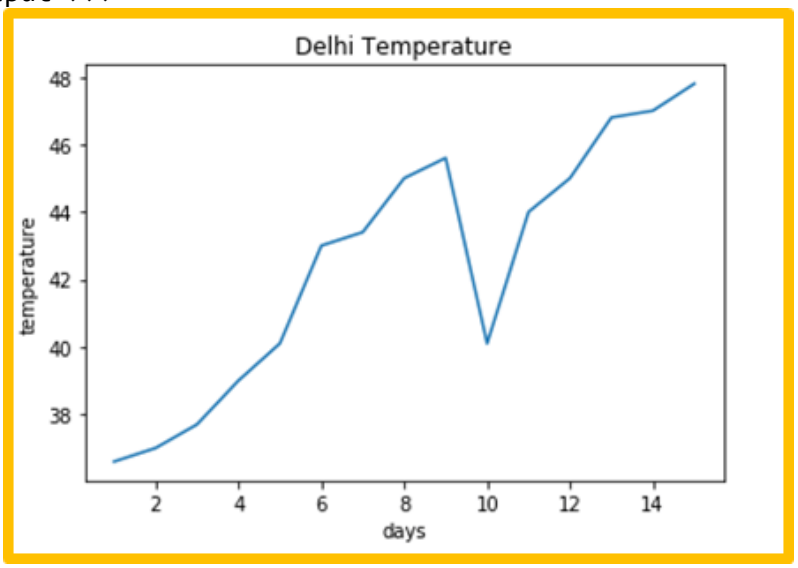


15 days temperature record of Delhi city

plot line with x-axis, y-axis and title

```
Pl. Plot(days, temperature)
Pl. Title("Delhi Temperature")
plt.xlabel("days")
plt.ylabel("temperature")
plt.show()
```

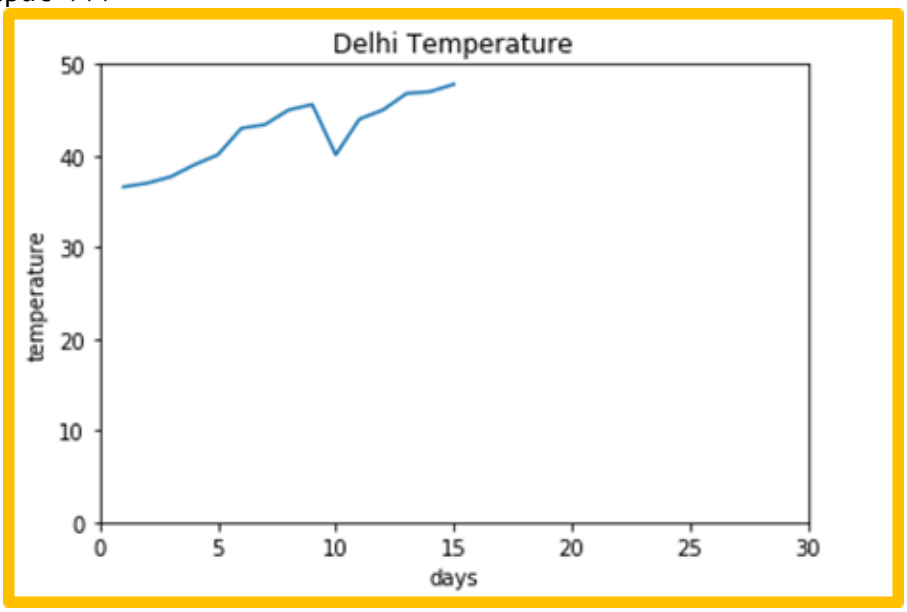
Output >>>



15 days temperature record of Delhi city with information

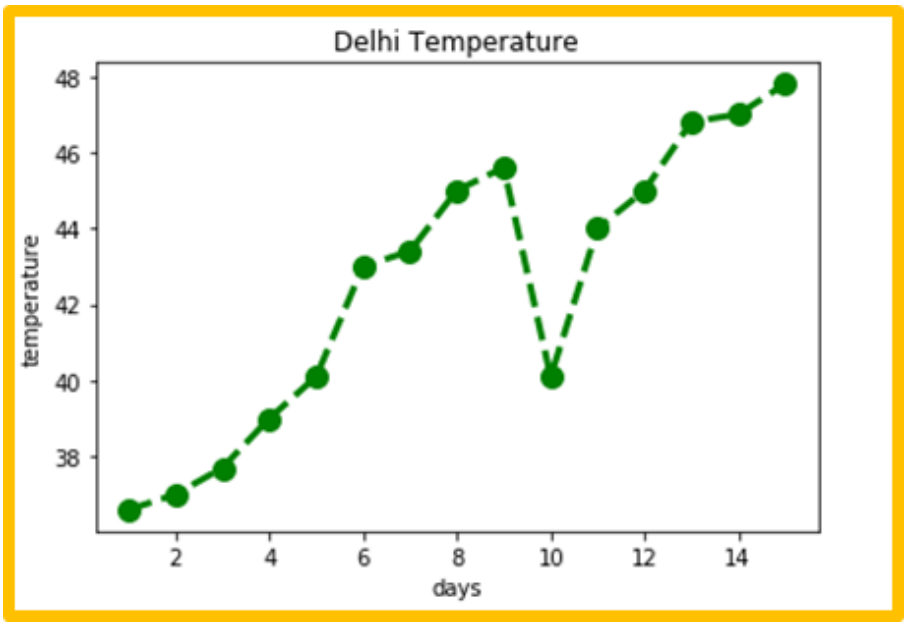
plt.plot() method has much more parameter. So, let's play with some of them.

```
plt.plot(days, temperature)
plt.axis([0,30, 0,50]) # set axis
plt.title("Delhi Temperature")
plt.xlabel("days")
plt.ylabel("temperature")
plt.show()
Output >>>
```



15 days temperature record of Delhi city set with the axis

```
plt.plot(days, temperature, color = "g", marker = "o", linestyle= "--",
linewidth = 3, markersize = 10)

plt.title("Delhi Temperature")
plt.xlabel("days")
plt.ylabel("temperature")
plt.show()

Output >>>
```



15 days temperature record of Delhi city with parameters

Color Parameter Values

Character	Color
b	blue
g	green
r	red
c	cyan
m	magenta
y	yellow
k	black
w	white

Marker Parameter Values

Character	Description
.	point marker
,	pixel marker
o	circle marker
v	triangle_down marker
^	triangle_up marker
<	triangle_left marker
>	triangle_right marker
1	tri_down marker
2	tri_up marker
s	square marker
p	pentagon marker
*	star marker
h	hexagon1 marker
H	hexagon2 marker
+	plus marker
x	x marker
D	diamond marker
d	thin_diamond marker
|	vline marker
_	hline marker

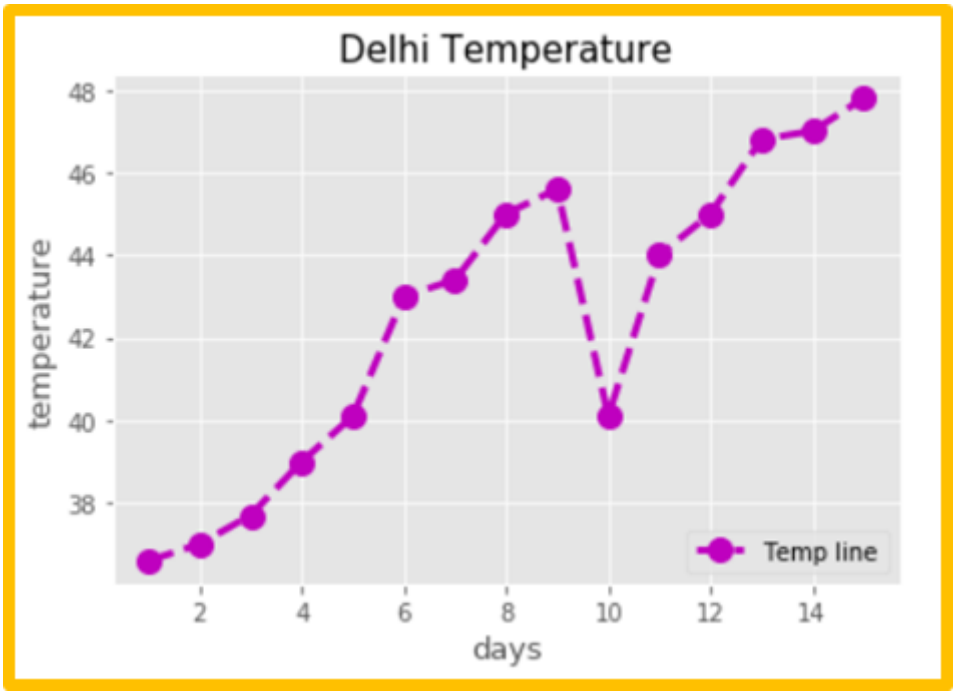
If you want to change the bar chart's background color and add grid then use **style.use()** method. For this first, need to import the style module from matplotlib.

Syntax: style.use(style)

To add a legend in the graph to describe more information about it, use **plt.legend().**

```
    from matplotlib import style # import style module
style.use("ggplot") # give ggplot parameter value to use() method
plt.plot(days, temperature, "mo--", linewidth = 3, markersize = 10, label = "Temp line")
plt.title("Delhi Temperature", fontsize=15) plt.xlabel("days",fontsize=13)
plt.ylabel("temperature",fontsize=13) plt.legend(loc = 4) plt.show()
```

Output >>>



15 days temperature record of Delhi city with style and legend

Matplotlib line plot

```
Here, we have 15 days temperature record of Delhi and Mumbai city
days = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
delhi_tem = [36.6, 37, 37.7,39,40.1,43,43.4,45,45.6,40.1,44,45,46.8,47,47.8]
mumbai_tem = [39,39.4,40,40.7,41,42.5,43.5,44,44.9,44,45,45.1,46,47,46]
plt.plot(days, delhi_tem, "mo--" ,linewidth = 3, markersize = 10, label = "Delhi
tem")
plt.plot(days, mumbai_tem, "yo:", linewidth = 3, markersize = 10, label = "Mumbai
tem")
```

The strings **"mo--"** and **"yo:"** are shorthand notations in Matplotlib to define **color, marker, and line style** in a single argument.

Let's break them down:

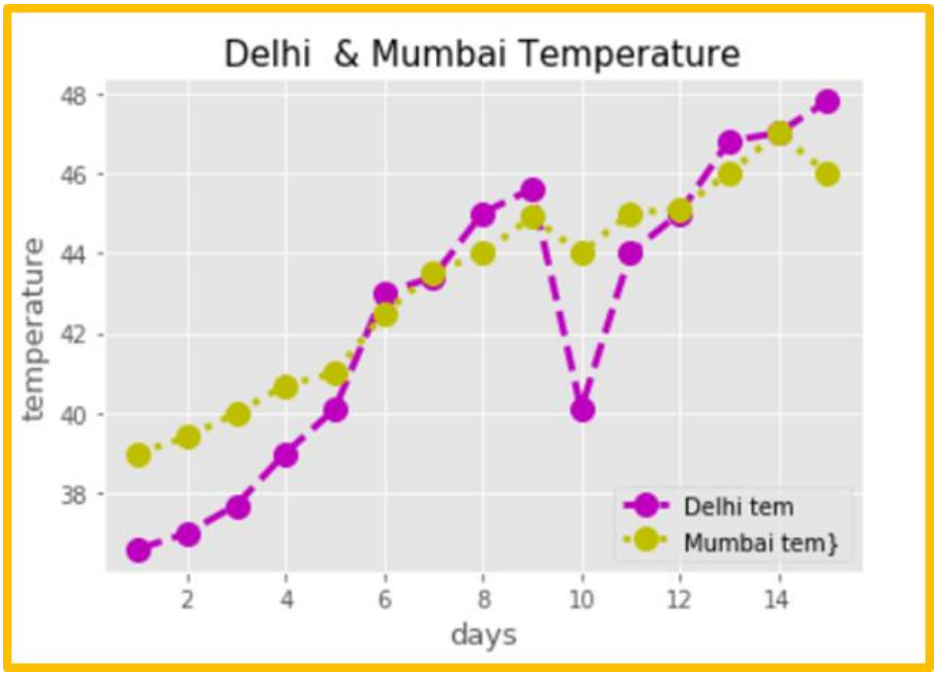
"mo--"

m → Magenta (color)
o → Circle (marker)
-- → Dashed line (line style)

"yo:"

y → Yellow (color)
o → Circle (marker)
: → Dotted line (line style)

Output >>>



15 days temperature record of Delhi & Mumbai city

Matplotlib provides **number-based** and **string-based** locations:

Code	Location Name	Position
0	"best"	Automatically selects the best position
1	"upper right"	Top-right corner
2	"upper left"	Top-left corner
3	"lower left"	Bottom-left corner
4	"lower right"	Bottom-right corner
5	"right"	Center-right
6	"center left"	Center-left
7	"center right"	Center-right
8	"lower center"	Bottom-center
9	"upper center"	Top-center
10	"center"	Center of the plot

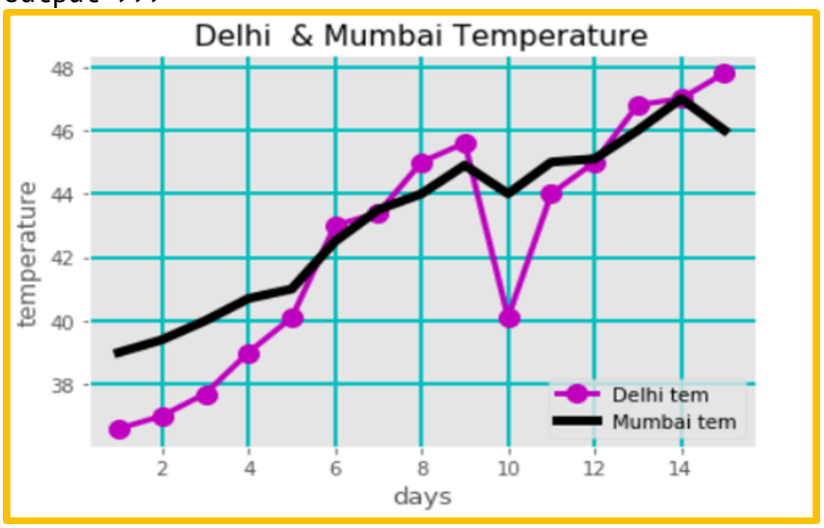
```
plt.title("Delhi & Mumbai Temperature", fontsize=15)
plt.xlabel("days",fontsize=13)
plt.ylabel("temperature",fontsize=13)
plt.legend(loc = 4) plt.show()
```

Output >>>

If you want to change or add grid then use **plt.grid()** method.

```
plt.plot(days, delhi_tem, "mo-", linewidth = 3, markersize = 10, label = "Delhi
tem")
plt.plot(days, mumbai_tem, "k-", linewidth = 5, markersize = 10, label = "Mumbai
tem")
plt.title("Delhi & Mumbai Temperature", fontsize=15)
plt.xlabel("days",fontsize=13)
plt.ylabel("temperature",fontsize=13)
plt.legend(loc = 4)
plt.grid(color='c', linestyle='-', linewidth=2) # grid with parameter
plt.show()
```

Output >>>



15 days temperature record of Delhi & Mumbai city with a grid

Matplotlib Histogram

The **pyplot.hist()** method from Matplotlib is used to create histograms.

Importing Required Libraries

```
import matplotlib.pyplot as plt
import numpy as np
```

Generating Random Data

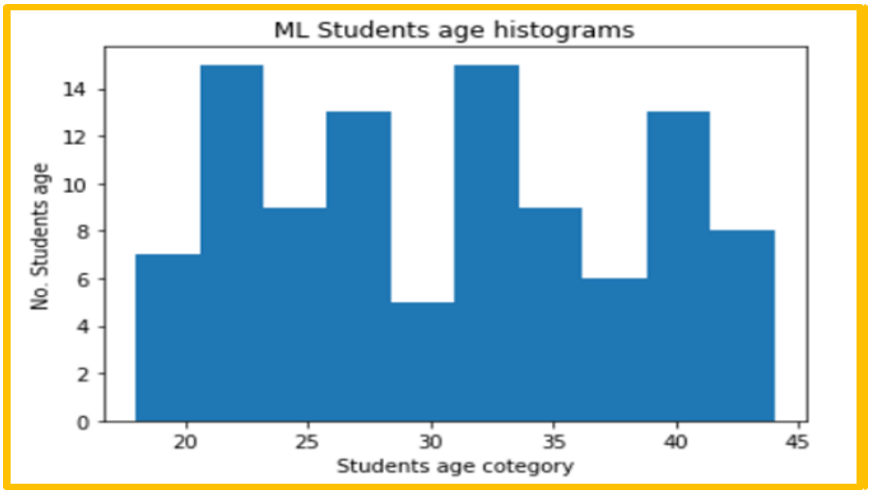
```
ml_students_age = np.random.randint(18, 45, 100)
py_students_age = np.random.randint(15, 40, 100)
print(ml_students_age)
print(py_students_age)
```

Histogram Plot

Plot a histogram for ML students' age distribution.

```
plt.hist(ml_students_age)
plt.title("ML Students Age Histogram")
plt.xlabel("Age Categories")
plt.ylabel("Number of Students")
plt.show()
```

Output >>>



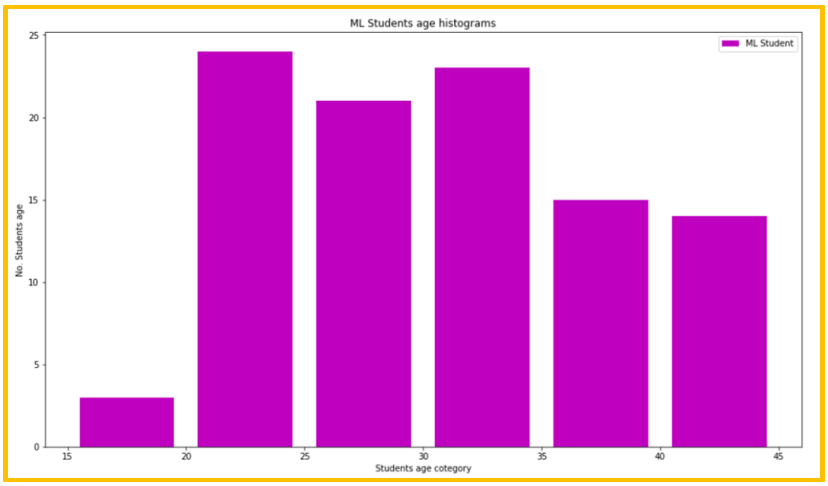
Matplotlib Histogram of ML students age

Customizing the Histogram

We add bins, color, orientation, and labels for better visualization.

```
bins = [15, 20, 25, 30, 35, 40, 45]  # Age categories
plt.figure(figsize=(16, 9))
plt.hist(ml_students_age, bins, rwidth=0.8, histtype='bar', orientation='vertical',
color='m', label='ML Students')
plt.title("ML Students Age Histogram")
plt.xlabel("Age Categories")
plt.ylabel("Number of Students")
plt.legend()
plt.show()
```

Output >>>



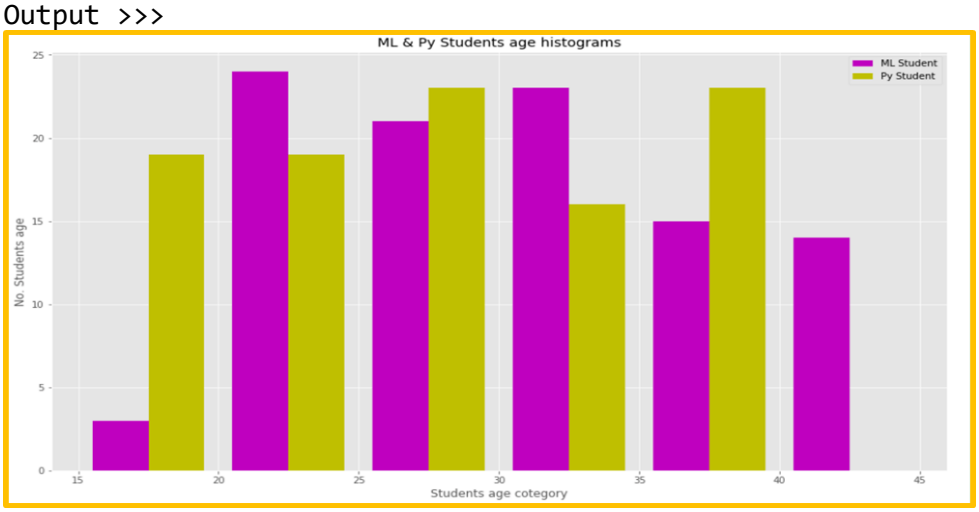
Matplotlib Histogram of ML students age with parameters

Plotting Multiple Histograms

We plot ML and Python students' ages together using different colors.

```
from matplotlib import style
style.use("ggplot")

plt.figure(figsize=(16, 9))
plt.hist([ml_students_age, py_students_age], bins, rwidth=0.8, histtype='bar',
orientation='vertical', color=["m", "y"], label=["ML Students", "Python Students"])
plt.title("ML & Python Students Age Histogram")
plt.xlabel("Age Categories")
plt.ylabel("Number of Students")
plt.legend()
plt.show()
```

Output >>>



Matplotlib Histogram of ML & Py students age with parameters

Matplotlib Bar Plot

To visualize value associated with categorical data in the bar format use matplotlib bar chart **plt.bar()** or **plt.barh()** methods.

Importing Required Libraries

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import style
```

Creating a Sample Dataset

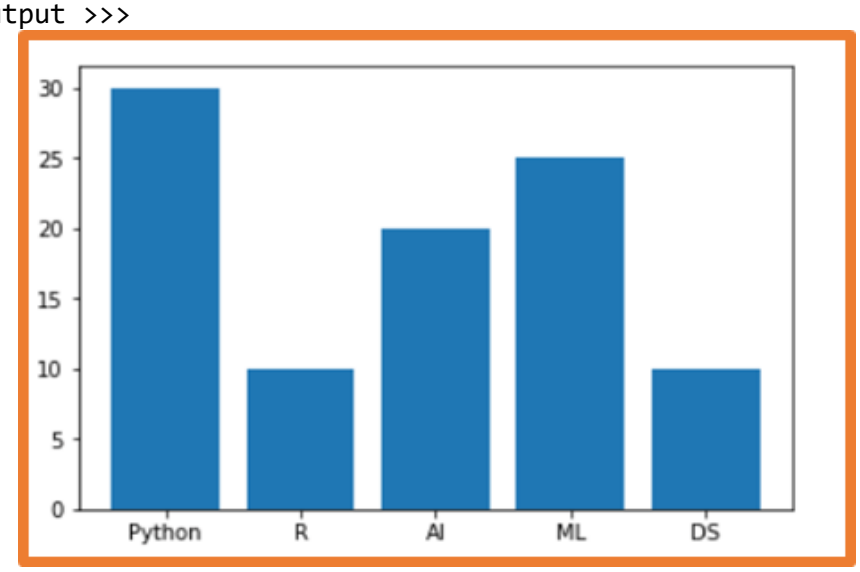
Let's create a dataset representing the number of students in different programming classes.

```
classes = ["Python", "R", "AI", "ML", "DS"]
class1_students = [30, 10, 20, 25, 10]
class2_students = [40, 5, 20, 20, 10]
class3_students = [35, 5, 30, 15, 15]
```

Plotting a Simple Bar Chart

```
plt.bar(classes, class1_students)
```

Output >>>

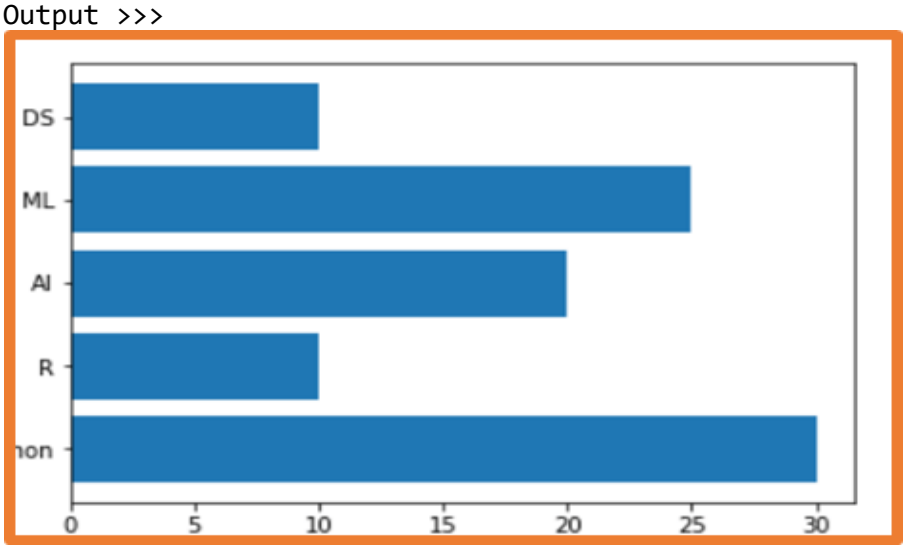


Matplotlib Vertical Bar Chart

Plotting a Horizontal Bar Chart

```
plt.barh(classes, class1_students)
```

Output >>>



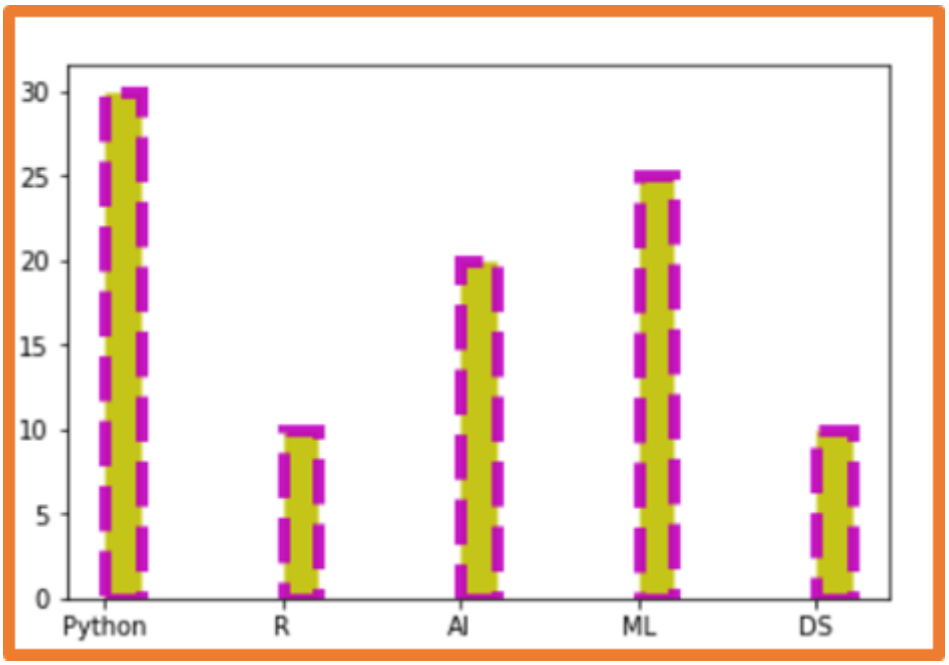
Matplotlib Horizontal Bar Chart

Customizing the Bar Chart

Let's modify the appearance of the bar chart using additional parameters.

```
plt.bar(classes, class1_students, width=0.6, align='edge', color='k', edgecolor='m',
linewidth=5, alpha=0.9, linestyle='--', label =" Class 1 Students", visible=False)
```

Output >>>

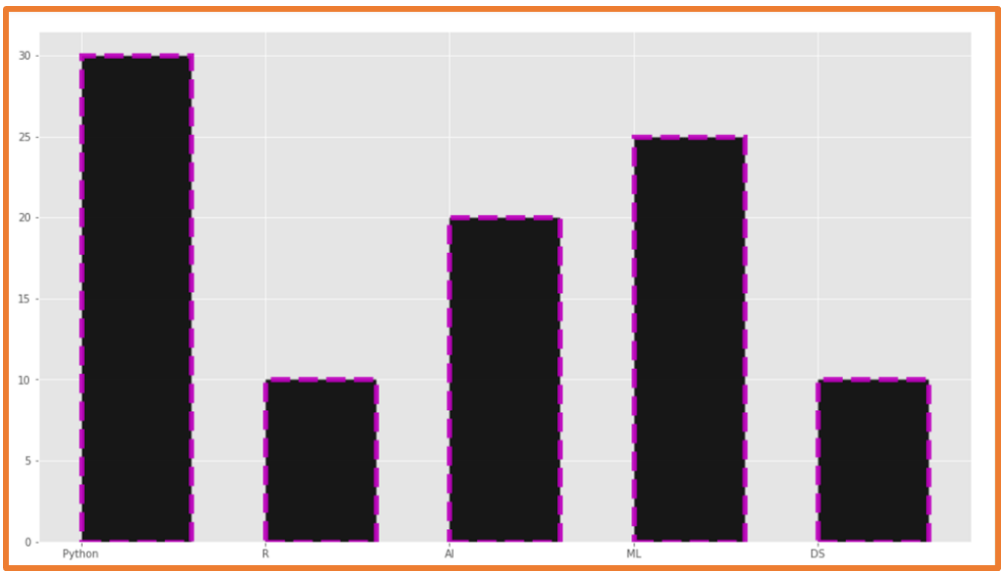


Matplotlib Vertical Bar Chart with multiple parameters

Increase figure size and use style

```
style.use("ggplot")
plt.figure(figsize=(16,9)) # figure size with ratio 16:9
plt.bar(classes, class1_students, width = 0.6, align = "edge", color = "k",
       edgecolor = "m", linewidth = 5, alpha = 0.9, linestyle = "--",
       label =" Class 1 Students") #visible=False
```

Output >>>

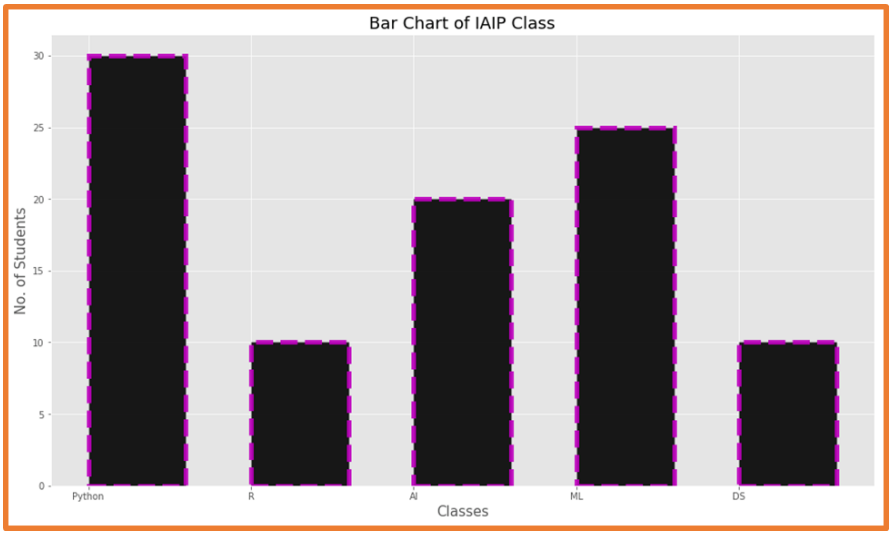


Matplotlib Vertical Bar Chart with multiple parameters

Decorating bar chart using multiple functions

```
plt.figure(figsize=(16,9))
plt.bar(classes, class1_students, width = 0.6, align = "edge", color = "k",
       edgecolor = "m", linewidth = 5, alpha = 0.9, linestyle = "--",
       label =" Class 1 Students") #visible=False

plt.title("Bar Chart of IAIP Class", fontsize = 18)
plt.xlabel("Classes",fontsize = 15)
plt.ylabel("No. of Students", fontsize = 15)
plt.show()
```

Output >>>

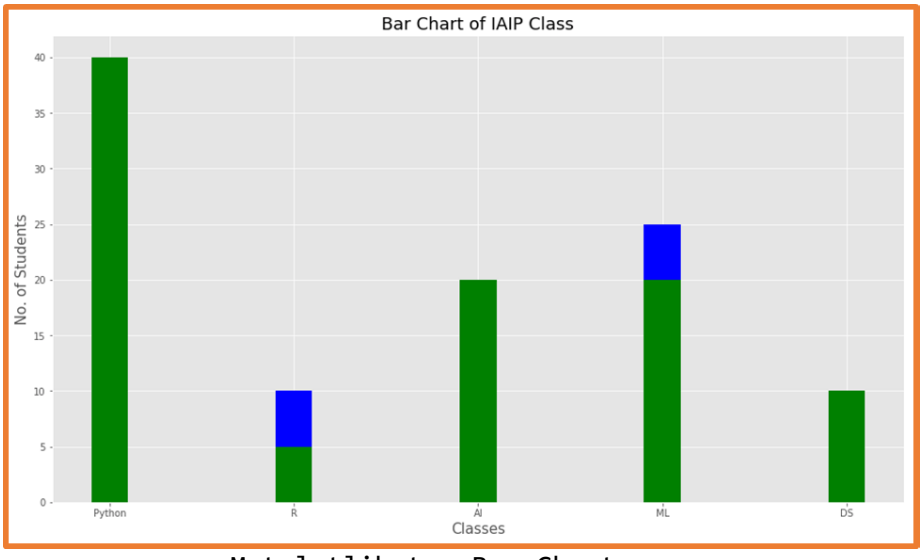


Matplotlib Horizontal Bar Chart with multiple functions

Trying to plot two bar charts with a different dataset.

```
plt.figure(figsize=(16,9))
plt.barh(classes_index, class1_students, width, color='b', label='Class 1 Students')
plt.barh(classes_index + width, class2_students, width, color='g', label='Class 2
Students')
plt.title("Bar Chart of IAIP Class", fontsize = 18)
plt.xlabel("Classes",fontsize = 15)
plt.ylabel("No. of Students", fontsize = 15)
plt.show()
```

Output >>>

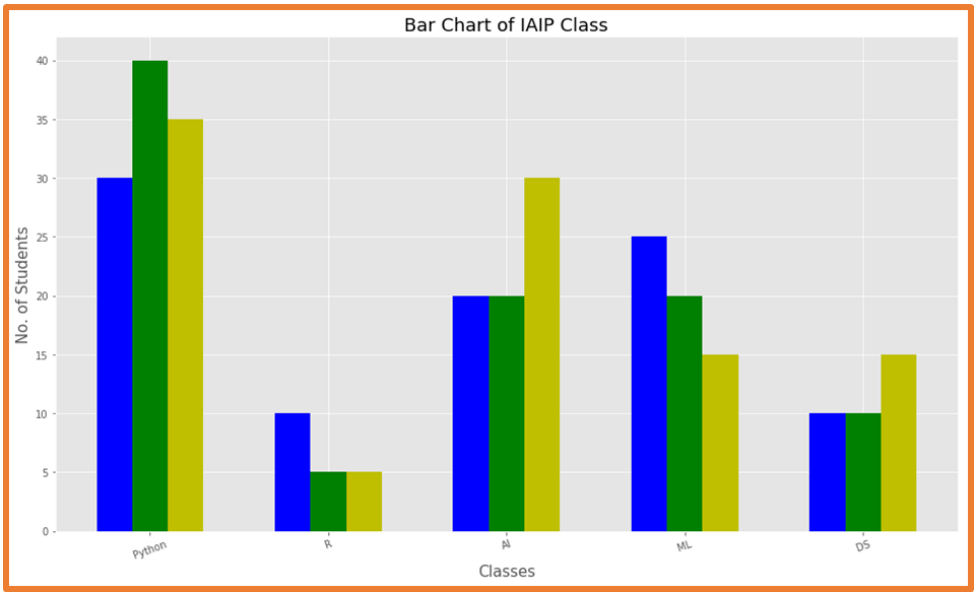


Matplotlib two Bar Chart

In below chart plot three bar charts using three different datasets.

```
plt.figure(figsize=(16,9))
classes_index = np.arange(len(classes))
width = 0.2
plt.bar(classes_index, class1_students, width , color = "b",
        label =" Class 1 Students") #visible=False
plt.bar(classes_index + width, class2_students, width , color = "g",
        label =" Class 2 Students")
plt.bar(classes_index + width + width, class3_students, width , color = "y",
        label =" Class 2 Students")

plt.xticks(classes_index + width, classes, rotation = 20)
plt.title("Bar Chart of IAIP Class", fontsize = 18)
plt.xlabel("Classes",fontsize = 15)
plt.ylabel("No. of Students", fontsize = 15)
plt.show()
```

Output >>>

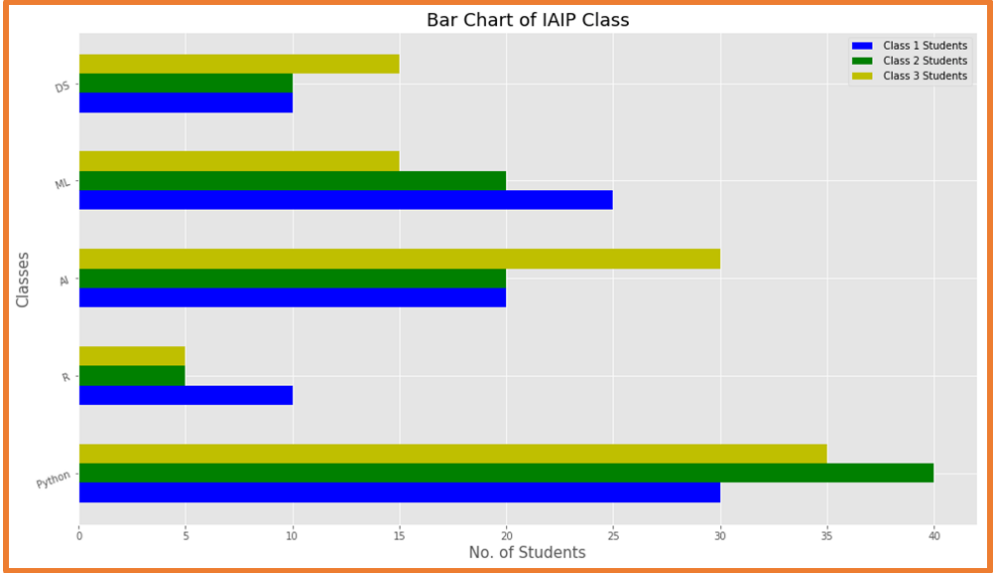


Matplotlib Three Bar Chart

Plot Matplotlib Horizontal bar chart with the above specification

```
plt.figure(figsize=(16,9))
classes_index = np.arange(len(classes))
width = 0.2

plt.barh(classes_index, class1_students, width , color = "b",
        label =" Class 1 Students") #visible=False

plt.barh(classes_index + width, class2_students, width , color = "g",
        label =" Class 2 Students")

plt.barh(classes_index + width + width, class3_students, width , color = "y",
        label =" Class 3 Students")

plt.yticks(classes_index + width, classes, rotation = 20)
plt.title("Bar Chart of IAIP Class", fontsize = 18)
plt.ylabel("Classes",fontsize = 15)
plt.xlabel("No. of Students", fontsize = 15)
plt.legend()
plt.show()
```

Output >>>



Matplotlib Three Horizontal Bar Chart