IITM DELHI

Matplotlib

UNIT -2

PYTHON

Btech 6th Sem)

ASHISH

(COMPUTER SCIENCE)

CONTENT

- **Introduction**
- **Line Plot- Basics**

Introduction to Matplotlib

- **Numpy** --->Data Analysis Library
- **Pandas**--->Data Analysis Library/Visualization library
- **Matplotlib/Seaborn/Plotly** --->Data Visualization Libraries

Need of Data visualization

- **Data** can be represented either in **text** form or in **graphical** form

- Data visualization is the representation of data in visual format.

Advantages

- We can **compare** very **easily**.
- We can **identify relationships** very easily.
- We can **identity symmetry** and **patterns** between data.
- We can **analyze** very **easily**. etc

More Than one Option

- There are multiple python based data visualization libraries:
- Matplotlib
- Seaborn
- Plotly

Basic Introduction to Matplotlib

- Most popular and **oldest data visulization** library.
- Python's **alternative** to **MatLab**
- It is **open source and freeware** where as **Matlab** is **not open source** (closed source) and not freeware.
- By using this library we can plot data in graphical form very easily. That graphical form can be **either 2-D or 3-D**.
- John Hunter **developed** matplotlib on **top of Numpy**
- It has very large community support.
- Every data scientist used this library atleast once in his life.
- Advanced libraries like seaborn, plotly are developed on top of matplotlib

WEBSITE

- The official website: https://matplotlib.org

Installing Matplotlib

There are 2 ways
With **Anaconda distribution**, this library will be available automatically.
conda install matplotlib

In our system, if python is already available, then we can install by using python package manager(pip)
pip install matplotlib

How to check installation

```
import matplotlib
print(matplotlib.__version__)
```

we can check the matplotlib installation using

pip list

pip freeze

.

Line Plot- Basics

There are **multiple types** are available to represent our data in graphical form.

The important are:

1. **Line** Plots
2. **Bar** charts
3. **Pie** charts
4. **Histogram**
5. **Scatter** plots

.

Basic of plot

Based on input data and requirement, we can choose the corresponding plot.

Note

matplotlib ==> package/library

pyplot ==> module name

pyplot module

defines several functions to create plots

a.plot()

b. bar()

c. pie()

d. hist()

e. scatter()

Line plots

- We can **mark data points** from the **input** data and we can **connect** these data points with lines. Such type of plots are called **line plots**.
- We can use **line plots** to determine the relationship between two data sets.
- **Data** set is a **collection** of values like **ndarray,python's** list. etc

- wickets = [1,2,3,4,5,6,7,8,9,10]
- overs = [1,4,5,,,,..20]
- The values from each data set will be plotted along an axis.(x-axis,y-axis)

Examples

- import matplotlib.pyplot as plt
- **plt.plot()** To create line plot
- **plt.bar()** To create bar chart
- **plt.pie()** To create pie chart
- **plt.hist()** To create Histograms
- **plt.scatter()** To created Scatter plots

Creation of line plot by passing 2 nd-arrays

- plt.plot(x,y)
- The data points will be considered from x and y values.
- x=[10,20,30] y=[1,2,3]
- Data points: (10,1), (20,2),(30,3)

Code 1

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

xpoints = np.array([10, 20,30])
ypoints = np.array([1, 2,3])

plt.plot(xpoints, ypoints)
plt.show()
```

Code 2

```
# Creation of line plot by passing 2 nd-arrays

import matplotlib.pyplot as plt
import numpy as np
 x = np.arange(1,11)
y = x**2
plt.plot(x,y)          #(1,1),(2,4),(3,9)...
plt.show()
```

How to add title and label(x,y)

- plt.title('Square Function Line Plot')
- plt.xlabel('N value')
- plt.ylabel('Square of N')

Note

- **plt.plot(x,y)** ==> To draw line plot
- **plt.title()** ==> To provide title to the line plot
- **plt.xlabel()** ==> Describes information about x-axis data

  plt.ylabel() ==> Describes information about y-axis data

  plt.show() ==> To show the line plot

Line Plots-Advanced

- A line drawn on the graph has several properties like **color**, **style**, **width** of the line, **transparency** etc.
- We can customize these based on our requirement.

Marker property

- We can use **marker** property to **highlight** data points on the line plot.
- We have to use **marker** keyword argument.

- plt.plot(a,b,marker='o') ==> o means circle

Marker property/Rules

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
'3'	tri_left marker
'4'	tri_right 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

Code 3

- import matplotlib.pyplot as plt
- import numpy as np
- x = np.arange(1,11)
- y = x**2
- **plt.plot(x,y,marker='o')** #(1,1),(2,4),(3,9)...
- plt.title('Square Function Line Plot')
- plt.xlabel('N value')
- plt.ylabel('Square of N')
- plt.show()

Linestyle Property

- Specifies the line style ==> **solid, dotted,dashed**
- we can use by using **linestyle** keyword argument

- plt.plot(a,b,marker='o',linestyle='--')

Linestyle Description

Line Styles

character	description
`'-'`	solid line style
`'--'`	dashed line style
`'-.'`	dash-dot line style
`':'`	dotted line style

Solid line style ==> default

```
import matplotlib.pyplot as plt
import numpy as np
 x = np.arange(1,11)
 y = x**2
 plt.plot(x,y,marker='o',linestyle='-')
    plt.title('Square Function Line Plot')
    plt.xlabel('N value') plt.ylabel('Square of N')
    plt.show()
```

dashed line style

```
import matplotlib.pyplot as plt
import numpy as np
 x = np.arange(1,11)
 y = x**2
plt.plot(x,y,marker='o',linestyle='--')
plt.title('Square Function Line Plot')
 plt.xlabel('N value')
 plt.ylabel('Square of N')
plt.show()
```

dash-dot line style

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(1,11)
y = x**2
 plt.plot(x,y,marker='o',linestyle='-.')
plt.title('Square Function Line Plot')
 plt.xlabel('N value')
plt.ylabel('Square of N')
plt.show()
```

dotted line style

```
import matplotlib.pyplot as plt
 import numpy as np
 x = np.arange(1,11)
y = x**2
plt.plot(x,y,marker='o',linestyle=':')
plt.title('Square Function Line Plot')
plt.xlabel('N value')
 plt.ylabel('Square of N')
 plt.show()
```

color property

- By using color keyword argument we can provide **colors** to our plot
- We can **specify** our **required** color for the line plot.
- We can use any color even **hexa** code also.

color property

- matplotlib defines some short codes for commonly used colors
- 'b' ==> blue
- 'g' ==> green
- 'r' ==> red
- 'c' ==> cyan
- 'm' ==> magento
- 'y' ==> yellow
- 'k' ==> black
- 'w' ==> white

Code 1

```
import matplotlib.pyplot as plt
 import numpy as np
 x = np.arange(1,11)
 y = x**2
plt.plot(x,y,marker='o',linestyle='-',color='green')
 plt.title('Square Function Line Plot')
 plt.xlabel('N value')
 plt.ylabel('Square of N')
plt.show()
```

Code 2

```
import matplotlib.pyplot as plt
 import numpy as np
 x = np.arange(1,11)
 y = x**2
plt.plot(x,y,marker='o',linestyle='-',color='#780257')
 plt.title('Square Function Line Plot')
 plt.xlabel('N value')
 plt.ylabel('Square of N')
plt.show()
```

default color

- If we are not specify color then default color will be selected from the style circle
- To find the default color plt.rcParams['axes.prop_cycle'].by_key() blue
- **first default orange**
- **second default green**
- **third default red**
- **fourth default**

default color Code

- import matplotlib.pyplot as plt
- import numpy as np
- a = np.arange(1,11)
- plt.plot(x,x) # blue
- plt.plot(x,x*2) # orange
- plt.plot(x,x*3) # green
- plt.plot(x,x*4) # red
- plt.show()

Bar Chart/ Bar Graph/ Bar Plot

- In a **line** plot, the **data points** will be marked and these markers will be connected by **line**.

- But in **bar** chart, data will be represented in the form of **bars**.

types of bar charts

- 4 types of bar charts
- Simple bar chart/vertical bar chart
- Horizontal bar chart

  Stacked Bar chart
- Clustered Bar Chart/Grouped Bar Char

Simple bar chart/vertical bar chart:

- The data will be represented in the form of **vertical** bars.
- Each **vertical** bar represents an **individual** category.
- The **height/length** of the bar is based on **value** it represents.
- Most of the times the width of the bar is **fixed**, but we can **customize**.
- The **default width: 0.8** .
- By using **bar**() function we can create bar chart

Simple bar chart/vertical bar chart:

```
import matplotlib.pyplot as plt
heroes = ['Prabhas','Pawan','Chiranjeevi','Sharukh','Amitabh'] # x-axis values
 movies = [100,300,200,600,1000] #height of bars, values for y-axis
plt.bar(heroes,movies)
plt.xlabel('HeroName',color='b',fontsize=15)
plt.ylabel('Number of Movies',color='b',fontsize=15)
plt.title('Hero wise number of movies',color='r',fontsize=15)
plt.show()
```

We can customize several things like

changing color of each bar

changing width of each bar

changing bottom of each bar

changing alignments etc

Observations

Observations

a) plt.bar(heroes,movies,color='r') ==> Now all bars with RED color

b) Separate color for each bar

```
c = ['r','b','k','g','orange']
plt.bar(heroes,movies,color=c)
```

c) The width of each bar should be 0.5(default is 0.8)

```
plt.bar(heroes,movies,width=0.5)
```

d) Different widths for bars

```
w = [0.8,0.6,0.7,0.9,0.5]
plt.bar(heroes,movies,width=w)
```

e) bottom should be 50 instead of 0

```
plt.bar(heroes,movies,bottom=50)
```

f) Different bottom values for bar?

```
b=[0,10,30,50,70]
plt.bar(heroes,movies,bottom=b)
```

g) alignment: center

for **left alignment**: plt.bar(heroes,movies,align='edge')

for **right alignment**: plt.bar(heroes,movies,width=-0.8,align='edge')

All bars with RED color

```
#plt.bar(heroes,movies,color='r')


import matplotlib.pyplot as plt
heroes = ['Venkatesh','Pawan','Chiranjeevi','Sharukh','Amitabh']
movies = [100,300,200,600,1000]
plt.bar(heroes,movies,color='r')
plt.xlabel('Hero Name',color='b',fontsize=15)
plt.ylabel('Number of Movies',color='b',fontsize=15)
plt.title('Hero wise number of movies',color='r',fontsize=15)
plt.show()
```

Separate color for each bar

c = ['r','b','k','g','orange']
plt.bar(heroes,movies,color=c)

```
import matplotlib.pyplot as plt
heroes=['Venkatesh','Pawan','Chiranjeevi','Sharukh','Amitabh']
movies = [100,300,200,600,1000]
c = ['r','b','k','g','orange']
plt.bar(heroes,movies,color=c)
plt.xlabel('Hero Name',color='b',fontsize=15)
 plt.ylabel('Number of Movies',color='b',fontsize=15)
plt.title('Hero wise number of movies',color='r',fontsize=15)
 plt.show()
```

WIDTH OF BAR

```
plt.bar(heroes,movies,width=0.5 )
```

```
import matplotlib.pyplot as plt
heroes=['Venkatesh','Pawan','Chiranjeevi','Sharukh','Amitabh']
movies = [100,300,200,600,1000]
plt.bar(heroes,movies,width=0.5 )
plt.xlabel('Hero Name',color='b',fontsize=15)
 plt.ylabel('Number of Movies',color='b',fontsize=15)
plt.title('Hero wise number of movies',color='r',fontsize=15)
 plt.show()
```

Different widths for bars

```
import matplotlib.pyplot as plt
heroes=['Venkatesh','Pawan','Chiranjeevi','Sharukh','Amitabh']
w = [0.8,0.6,0.7,0.9,0.5]
plt.bar(heroes,movies,width=w )
plt.xlabel('Hero Name',color='b',fontsize=15)
 plt.ylabel('Number of Movies',color='b',fontsize=15)
plt.title('Hero wise number of movies',color='r',fontsize=15)
 plt.show()
```

Nokia Mobile Sales in the last Decade

years = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]

sales = [10000, 25000, 45000, 30000, 10000, 5000,70000,60000,65000,50000]

Nokia Mobile Sales in the last Decade

```
import matplotlib.pyplot as plt
years = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]
sales = [10000, 25000, 45000, 30000, 10000,
 5000,70000,60000,65000,50000]
c = ['r','k','y','g','orange','m','c','b','lime','violet']
plt.bar(years,sales,color=c)
plt.xlabel('Year',color='b',fontsize=15)
plt.ylabel('Number of Sales',color='b',fontsize=15)
plt.title('Nokia Mobile Sales in the last Decade',color='r',fontsize=15)
plt.show()
```

Nokia Mobile Sales in the last Decade

```
import matplotlib.pyplot as plt
years = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]
sales = [10000, 25000, 45000, 30000, 10000,
 5000,70000,60000,65000,50000]
c = ['r','k','y','g','orange','m','c','b','lime','violet']
plt.bar(years,sales,color=c)
plt.xlabel('Year',color='b',fontsize=15)
plt.ylabel('Number of Sales',color='b',fontsize=15)
plt.title('Nokia Mobile Sales in the last Decade',color='r',fontsize=15)
plt.xticks(years,rotation=30)
plt.grid(axis='y')
plt.show()
```

Pie Chart

- Pie chart is a **circular** chart divided into **segments**.
- These segments are called **wedges**.
- Each **wedge** represents an **individual category**.
- The area of the wedge is proprotional to value of that category.
- Pie chart is very helpful for comparison of categories.

The number of categories are less mostly <=5.

Eg: **20 overs, overwise scores-**-->bar chart but not pie chart

The chance of winning match-->pie chart

By using pie() function of pyplot -→pie chart

CODE 1

```
import matplotlib.pyplot as plt
import numpy as np
 marks = np.array([25,30,43,12])
plt.pie(marks)
plt.show()
```

Adding Labels ==> labels argument

- import matplotlib.pyplot as plt
- import numpy as np
- marks = np.array([25,30,43,12])
- mylabels = ['Python','Java','Devops','DataScience']
- plt.pie(marks,labels=mylabels)
- plt.show()

auto percentage

- import matplotlib.pyplot as plt
- import numpy as np
- marks = np.array([25,30,43,12])
- mylabels = ['Python','Java','Devops','DataScience']
- plt.pie(marks,labels=mylabels,autopct = '%.1f')
- plt.pie(marks,labels=mylabels)
- plt.show()

explode

- If we want to explode/highlight a particular category then we should use explode argument.

- import matplotlib.pyplot as plt
- import numpy as np
- marks = np.array([25,30,43,12])
- mylabels = ['Python','Java','Devops','DataScience']
- myexplode = [0.0,0.0,0.0,**0.2**]
- plt.pie(marks,labels=mylabels,autopct = '%.2f',explode=myexplode)
- plt.show()

explode

- If we want to explode/highlight a particular category then we should use explode argument.

- import matplotlib.pyplot as plt
- import numpy as np
- marks = np.array([25,30,43,12])
- mylabels = ['Python','Java','Devops','DataScience']
- myexplode = [0.0,0.0,0.0,**0.2**]
- mycolors = ['g','b','r','yellow']
- plt.pie(marks,labels=mylabels,autopct = '%.2f',explode=myexplode, colors=mycolors)
- plt.show()