MatplotLib

Types Of Plots :

Function	Description
Bar	Make a bar plot.
Barh	Make a horizontal bar plot.
Boxplot	Make a box and whisker plot.
Hist	Plot a histogram.
hist2d	Make a 2D histogram plot.
Pie	Plot a pie chart.
Plot	Plot lines and/or markers to the Axes.
Polar	Make a polar plot..
Scatter	Make a scatter plot of x vs y.
Stackplot	Draws a stacked area plot.
Stem	Create a stem plot.
Step	Make a step plot.
quiver	Plot a 2-D field of arrows.

Image Functions

Function	Description
Imread	Read an image from a file into an array.
Imsave	Save an array as in image file.
imsave	Display an image on the axes.

Figure Functions

Function	Description
Figtext	Add text to figure.
Figure	Creates a new figure
Show	Display a figure
Savefig	Save the current figure.
Close	Close a figure window.

Axis Functions

Function	Description
Axes	Add axes to the figure.
Text	Add text to the axes.
Title	Set a title of the current axes.
Xlabel	Set the x axis label of the current axis.
Xlim	Get or set the x limits of the current axes.
Xscale	
Xticks	Get or set the x-limits of the current tick locations and labels.
Ylabel	Set the y axis label of the current axis.
Ylim	Get or set the y-limits of the current axes.
Yscale	Set the scaling of the y-axis.
Yticks	Get or set the y-limits of the current tick locations and labels.

Pyplot :

- Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias:
- ```
  import matplotlib.pyplot as plt
  import numpy as np

  xpoints = np.array([0, 6])
  ypoints = np.array([0, 250])

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

Matplotlib Plotting :

- Plotting x and y points :
- Parameter 1 is an array containing the points on the **x-axis**.
- Parameter 2 is an array containing the points on the **y-axis**.
- Draw a line in a diagram from position (1, 3) to position (8, 10):
- ```
  import matplotlib.pyplot as plt
  import numpy as np
  xpoints = np.array([1, 8])
  ypoints = np.array([3, 10])
  plt.plot(xpoints, ypoints)
  plt.show()
  ```
  ```

Plotting Without Line :

- Draw two points in the diagram, one at position (1, 3) and one in position (8, 10):
- ```
  import matplotlib.pyplot as plt
  import numpy as np
  xpoints = np.array([1, 8])
  ypoints = np.array([3, 10])
  plt.plot(xpoints, ypoints, 'o')
  plt.show()
  ```
  ```

Multiple Points:

- Draw a line in a diagram from position (1, 3) to (2, 8) then to (6, 1) and finally to position (8, 10):
- ```
  import matplotlib.pyplot as plt
  import numpy as np

  xpoints = np.array([1, 2, 6, 8])
  ypoints = np.array([3, 8, 1, 10])

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

Default X-Points :

- If we do not specify the points in the x-axis, they will get the default values 0, 1, 2, 3, (etc. depending on the length of the y-points.
- import matplotlib.pyplot as plt
  import numpy as np
  ypoints = np.array([3, 8, 1, 10, 5, 7])
  plt.plot(ypoints)
  plt.show()

Matplotlib Markers :

- Mark each point with a circle:
- ```
  import matplotlib.pyplot as plt
  import numpy as np
  ypoints = np.array([3, 8, 1, 10])
  plt.plot(ypoints, marker = 'o')
  plt.show()
  ```
- Mark each point with a star:
  ```
  plt.plot(ypoints, marker = '*')
  ```
  ```

Format string fmt

- You can use also use the *shortcut string notation* parameter to specify the marker.
- This parameter is also called fmt, and is written with this syntax:
  marker|line|color
- ```
  import matplotlib.pyplot as plt
  import numpy as np

  ypoints = np.array([3, 8, 1, 10])

  plt.plot(ypoints, 'o:r')
  plt.show()
  ```
  ```

Line reference:

Line Syntax	Description
‘-’	Solid line
‘:’	Dotted line
‘--’	Dashed line
‘-.’	Dashed/dotted line

Color Reference :

Color Syntax	Description
r	Red
g	Green
b	Blue
c	Cyan
m	Magenta
y	Yellow
k	Black
W	White

Marker Size and color :

- Set the size of the markers to 20:
- ```
  import matplotlib.pyplot as plt
  import numpy as np
  ypoints = np.array([3, 8, 1, 10])
  plt.plot(ypoints, marker = 'o', ms = 20, mec = 'r')
  plt.show()
  ```
  ```

Matplotlib Line

You can use the keyword argument linestyle, to change the style of the plotted line:

Eg.1:

```
import matplotlib.pyplot as plt
import numpy as np
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, linestyle = 'dotted')
plt.show()
```

Eg.2:

```
plt.plot(ypoints, linestyle = 'dashed')
```

Shorter Syntax

The line style can be written in a shorter syntax:

linestyle can be written as ls. dotted can be written as :. dashed can be written as --.

Line Styles :

Style	Method
'solid' (default)	‘-’
'dotted'	‘:’
'dashed'	‘--’
'dashdot'	‘-.’
'None'	‘’ or “

Line Color :

- Set the line color to red:
- import matplotlib.pyplot as plt
  import numpy as np

  ypoints = np.array([3, 8, 1, 10])

  plt.plot(ypoints, color = 'r')
  plt.show()

Line Width :

- You can use the keyword argument linewidth or the shorter lw to change the width of the line.
- Plot with a 20.5pt wide line:
- ```
  import matplotlib.pyplot as plt
  import numpy as np

  ypoints = np.array([3, 8, 1, 10])

  plt.plot(ypoints, linewidth = '20.5')
  plt.show()
  ```
  ```

Multiple Lines

- Draw two lines by specifiyng the x- and y-point values for both lines:
- ```
  import matplotlib.pyplot as plt
  import numpy as np

  x1 = np.array([0, 1, 2, 3])
  y1 = np.array([3, 8, 1, 10])
  x2 = np.array([0, 1, 2, 3])
  y2 = np.array([6, 2, 7, 11])

  plt.plot(x1, y1, x2, y2)
  plt.show()
  ```
  ```

Matplotlib Labels and Title :

Create Labels for a Plot :
With Pyplot, you can use the xlabel() and ylabel() functions to set a label for the x- and y-axis.

```
import numpy as np
import matplotlib.pyplot as plt
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])
plt.plot(x, y)
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
plt.show()
```

Title for a Plot and Set font Properties :

- ```
  import numpy as np
  import matplotlib.pyplot as plt
  x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
  y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])
  plt.plot(x, y)
  plt.title("Sports Watch Data", fontdict = font1)
  plt.xlabel("Average Pulse", fontdict = font2)
  plt.ylabel("Calorie Burnage", fontdict = font2)
  plt.show()
  ```
  ```

Matplotlib Adding Grid Lines :

- Add Grid Lines to a Plot :
- With Pyplot, you can use the grid() function to add grid lines to the plot.
- ```
  import numpy as np
  import matplotlib.pyplot as plt
  x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
  y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])
  plt.title("Sports Watch Data")
  plt.xlabel("Average Pulse")
  plt.ylabel("Calorie Burnage")
  plt.plot(x, y)
  plt.grid()
  plt.show()
  ```
  ```

Specify Which Grid Lines to Display

You can use the axis parameter in the grid() function to specify which grid lines to display.

Legal values are: 'x', 'y', and 'both'. Default value is 'both'.

```
import numpy as np
import matplotlib.pyplot as plt
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])
plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
plt.plot(x, y)
plt.grid(axis = 'x')
plt.show()
```

Set Line Properties for the Grid :

- You can also set the line properties of the grid, like this: grid(color = 'color', linestyle = 'linestyle', linewidth = number).
- import numpy as np
  import matplotlib.pyplot as plt
  x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
  y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])
  plt.title("Sports Watch Data")
  plt.xlabel("Average Pulse")
  plt.ylabel("Calorie Burnage")
  plt.plot(x, y)
  plt.grid(color = 'green', linestyle = '--', linewidth = 0.5)
  plt.show()

Matplotlib Subplots :

- Display Multiple Plots:Draw 2 plots:
- ```
  import matplotlib.pyplot as plt
  import numpy as np
  #plot 1:
  x = np.array([0, 1, 2, 3])
  y = np.array([3, 8, 1, 10])
  plt.subplot(1, 2, 1)
  plt.plot(x,y)
  #plot 2:
  x = np.array([0, 1, 2, 3])
  y = np.array([10, 20, 30, 40])
  plt.subplot(1, 2, 2)
  plt.plot(x,y)
  plt.show()
  ```
  ```

The subplots() Function

The subplots() function takes three arguments that describes the figure.
The layout is organized in rows and columns, which are represented by the *first* and *second* argument.
The third argument represents the index of the current plot.

```
plt.subplot(1, 2, 1)
#the figure has 1 row, 2 columns, and this plot is the first plot.
plt.subplot(1, 2, 2)
#the figure has 1 row, 2 columns, and this plot is the second plot.
```

Title and Super Title:

- ```
  import matplotlib.pyplot as plt
  import numpy as np
  x = np.array([0, 1, 2, 3])            #plot 1
  y = np.array([3, 8, 1, 10])
  plt.subplot(1, 2, 1)
  plt.plot(x,y)
  plt.title("SALES")
  x = np.array([0, 1, 2, 3])            #plot 2:
  y = np.array([10, 20, 30, 40])
  plt.subplot(1, 2, 2)
  plt.plot(x,y)
  plt.title("INCOME")

  plt.suptitle("MY SHOP")
  plt.show()
  ```
  ```

Matplotlib Scatter :

- Creating Scatter Plots :
- import matplotlib.pyplot as plt
  import numpy as np

  x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
  y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

  plt.scatter(x, y)
  plt.show()

Compare Plots :

- ```
  import matplotlib.pyplot as plt
  import numpy as np
  #day one, the age and speed of 13 cars:
  x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
  y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
  plt.scatter(x, y)
  #day two, the age and speed of 15 cars:
  x = np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
  y = np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])
  plt.scatter(x, y)
  plt.show()
  ```
  ```

Colors :

- ```
  import matplotlib.pyplot as plt
  import numpy as np
  x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
  y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
  plt.scatter(x, y, color = ‘red')
  x = np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
  y = np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])
  plt.scatter(x, y, color = '#88c999')
  plt.show()
  ```
  ```

Color each dots:

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

  x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
  y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
  colors =
  np.array(["red","green","blue","yellow","pink","black","orange","purple","beige",
  "brown","gray","cyan","magenta"])

  plt.scatter(x, y, c=colors)

  plt.show()
  ```
  ```

Size :

You can change the size of the dots with the s argument.

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

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
sizes = np.array([20,50,100,200,500,1000,60,90,10,300,600,800,75])

plt.scatter(x, y, s=sizes)

plt.show()
```

Alpha :

You can adjust the transparency of the dots with the alpha argument.

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

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
sizes = np.array([20,50,100,200,500,1000,60,90,10,300,600,800,75])

plt.scatter(x, y, s=sizes, alpha=0.5)

plt.show()
```

Combine Color Size and Alpha

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

  x = np.random.randint(100, size=(100))
  y = np.random.randint(100, size=(100))
  colors = np.random.randint(100, size=(100))
  sizes = 10 * np.random.randint(100, size=(100))

  plt.scatter(x, y, c=colors, s=sizes, alpha=0.5, cmap='nipy_spectral')

  plt.colorbar()

  plt.show()
  ```
  ```

Matplotlib Bars :

- Creating Bars
- With Pyplot, you can use the bar() function to draw bar graphs:
- ```
  import matplotlib.pyplot as plt
  import numpy as np

  x = np.array(["A", "B", "C", "D"])
  y = np.array([3, 8, 1, 10])

  plt.bar(x,y)
  plt.show()
  ```
  ```

Horizontal Bars :

- If you want the bars to be displayed horizontally instead of vertically, use the barh() function:
- ```
  import matplotlib.pyplot as plt
  import numpy as np

  x = np.array(["A", "B", "C", "D"])
  y = np.array([3, 8, 1, 10])

  plt.barh(x, y)
  plt.show()
  ```
  ```

Bar Color :

The bar() and barh() takes the keyword argument color to set the color of the bars:

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

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x, y, color = "red")
plt.show()
```

Bar Width :

The bar() takes the keyword argument width to set the width of the bars:

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

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x, y, width = 0.1)
plt.show()
```

Bar Height :

The barh() takes the keyword argument height to set the height of the bars:

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

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.barh(x, y, height = 0.1)
plt.show()
```

Matplotlib Histograms :

- In Matplotlib, we use the hist() function to create histograms.
- The hist() function will use an array of numbers to create a histogram, the array is sent into the function as an argument.
- ```
  import matplotlib.pyplot as plt
  import numpy as np

  x = np.random.normal(170, 10, 250)

  plt.hist(x)
  plt.show()
  ```
  ```

Matplotlib Pie Charts :

- With Pyplot, you can use the pie() function to draw pie charts:
- ```
  import matplotlib.pyplot as plt
  import numpy as np

  y = np.array([35, 25, 25, 15])

  plt.pie(y)
  plt.show()
  ```
  ```

Labels :

- Add labels to the pie chart with the label parameter.
- The label parameter must be an array with one label for each wedge:
- ```
  import matplotlib.pyplot as plt
  import numpy as np

  y = np.array([35, 25, 25, 15])
  mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

  plt.pie(y, labels = mylabels)
  plt.show()
  ```
  ```

Start Angle:

- import matplotlib.pyplot as plt
  import numpy as np

  y = np.array([35, 25, 25, 15])
  mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

  plt.pie(y, labels = mylabels, startangle = 90)
  plt.show()

Explode

- ```
  import matplotlib.pyplot as plt
  import numpy as np
  y = np.array([35, 25, 25, 15])
  mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
  myexplode = [0.2, 0, 0, 0]
  plt.pie(y, labels = mylabels, explode = myexplode)
  plt.show()
  ```
  ```

Shadow

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

  y = np.array([35, 25, 25, 15])
  mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
  myexplode = [0.2, 0, 0, 0]

  plt.pie(y, labels = mylabels, explode = myexplode, shadow = True)
  plt.show()
  ```
  ```

Colors

- import matplotlib.pyplot as plt
  import numpy as np

  y = np.array([35, 25, 25, 15])
  mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
  mycolors = ["black", "hotpink", "b", "#4CAF50"]

  plt.pie(y, labels = mylabels, colors = mycolors)
  plt.show()

Legend

- import matplotlib.pyplot as plt
  import numpy as np
  y = np.array([35, 25, 25, 15])
  mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
  plt.pie(y, labels = mylabels)
  plt.legend()
  plt.show()
- Legend With Header
- y = np.array([35, 25, 25, 15])
  mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
  plt.pie(y, labels = mylabels)
  plt.legend(title = "Four Fruits:")
  plt.show()