Pandas Plot

Pandas provides a convenient way to visualize data directly from DataFrames and Series using the `plot()` method.

This method uses the Matplotlib library behind the scenes to create various types of plots.

Let's learn about visualization techniques in Pandas.

Dataset For Data Visualization

We'll use the following dataset to visualize data.

Car	Weight
Caterham	0.48 tons
Tesla	1.7 tons
Audi	2 tons
BMW	2 tons
Ford	2.5 tons
Jeep	3 tons

Line Plot For Data Visualization

In Pandas, a line plot displays data as a series of points connected by a line. We use the `plot()` function to create a line plot, which takes two arguments: x and y coordinates.

Let's look at an example.

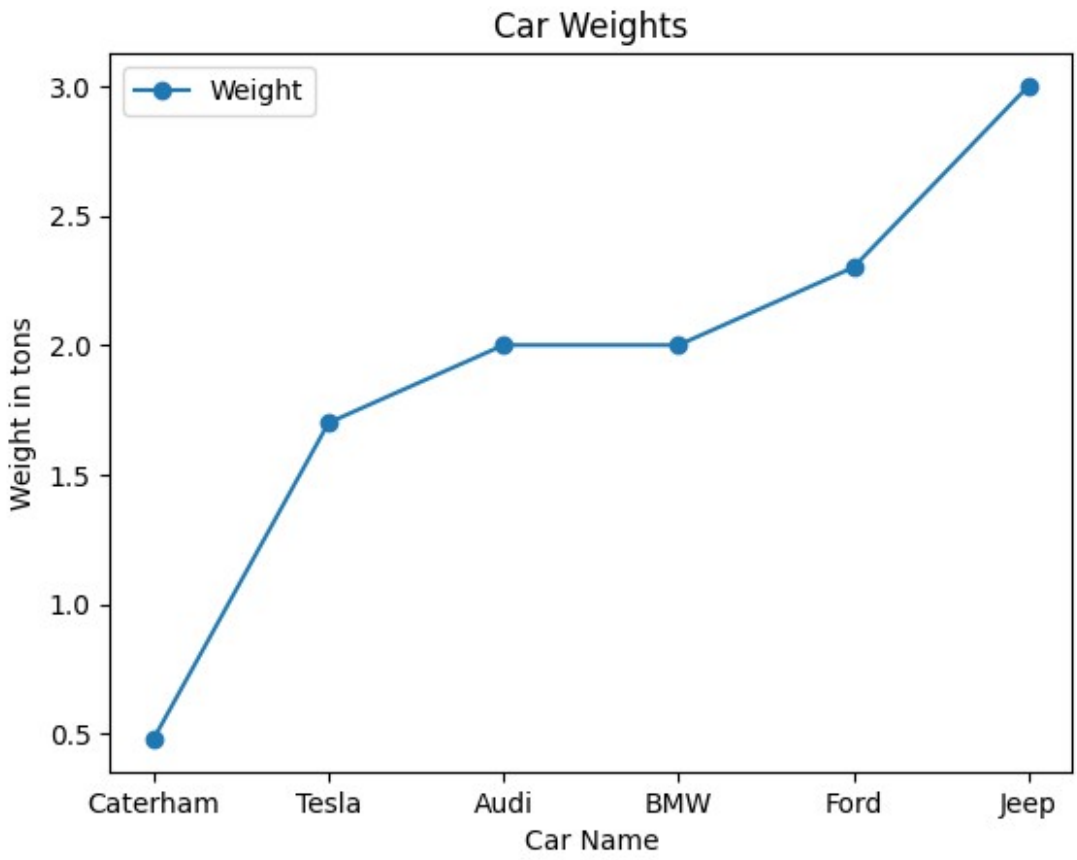
```
!pip install matplotlib

Requirement already satisfied: matplotlib in c:\users\rmnjs\appdata\
local\programs\python\python39\lib\site-packages (3.8.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\rmnjs\
appdata\local\programs\python\python39\lib\site-packages (from
matplotlib) (1.1.1)
Requirement already satisfied: cycler>=0.10 in c:\users\rmnjs\appdata\
local\programs\python\python39\lib\site-packages (from matplotlib)
(0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\rmnjs\
appdata\local\programs\python\python39\lib\site-packages (from
matplotlib) (4.42.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\rmnjs\
appdata\local\programs\python\python39\lib\site-packages (from
matplotlib) (1.4.5)
Requirement already satisfied: numpy<2,>=1.21 in c:\users\rmnjs\
```

```
appdata\local\programs\python\python39\lib\site-packages (from
matplotlib) (1.26.0)
Requirement already satisfied: packaging>=20.0 in c:\users\rmnjs\
appdata\local\programs\python\python39\lib\site-packages (from
matplotlib) (23.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\rmnjs\
appdata\local\programs\python\python39\lib\site-packages (from
matplotlib) (9.5.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\rmnjs\
appdata\local\programs\python\python39\lib\site-packages (from
matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\rmnjs\
appdata\local\programs\python\python39\lib\site-packages (from
matplotlib) (2.8.2)
Requirement already satisfied: importlib-resources>=3.2.0 in c:\users\
rmnjs\appdata\local\programs\python\python39\lib\site-packages (from
matplotlib) (6.1.0)
Requirement already satisfied: zipp>=3.1.0 in c:\users\rmnjs\appdata\
local\programs\python\python39\lib\site-packages (from importlib-
resources>=3.2.0->matplotlib) (3.17.0)
Requirement already satisfied: six>=1.5 in c:\users\rmnjs\appdata\
local\programs\python\python39\lib\site-packages (from python-
dateutil>=2.7->matplotlib) (1.16.0)
```

```
import matplotlib

print(matplotlib.__version__)
```

```
3.8.0
```

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

car = ["Caterham", "Tesla", "Audi", "BMW", "Ford", "Jeep"]
weight = [0.48, 1.7, 2, 2, 2.3, 3]

# create a DataFrame
data = {'Car': car, 'Weight': weight}
df = pd.DataFrame(data)

# plot using Pandas
data = {'Weight': weight}
df.plot(x='Car', y='Weight', kind='line', marker='o' ) #linestyle =
'dashed', linestyle = 'dotted'
plt.xlabel('Car Name')
plt.ylabel('Weight in tons')
plt.title('Car Weights')
plt.show()
```



Here, we have used the `plot()` function to line plot the given dataset. We set the `x` and `y` coordinate of `plot()` as the `car` and `weight`.

The `kind` parameter is set to `'line'` to create the line plot, and marker is set to `'o'` to display circular markers at data points.

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 `--`.

`'solid' (default)` or `'-'`

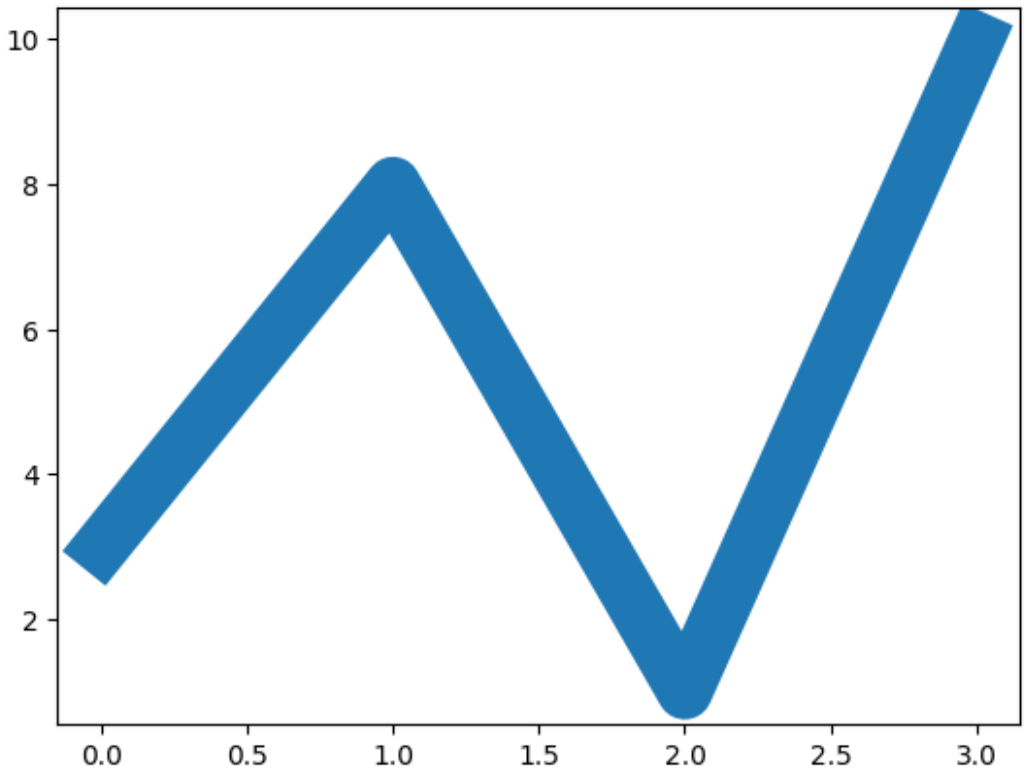
`'dashdot'` or `'-.'`

`'None'` or `'' or ' '`

Line Color

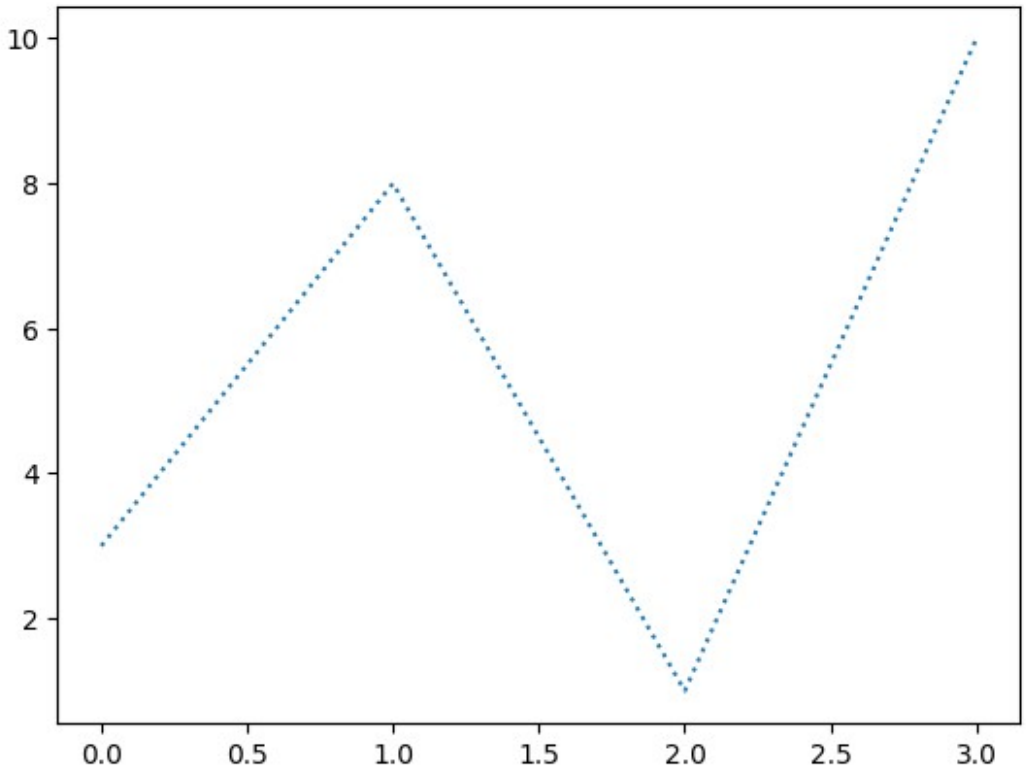
You can use the keyword argument color or the shorter c to set the color of the line:

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

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

plt.plot(ypoints, color = 'r')
#plt.plot(ypoints, c = '#4CAF50')
#plt.plot(ypoints, c = 'hotpink')
#plt.plot(ypoints, linewidth = '20.5')
plt.show()
```



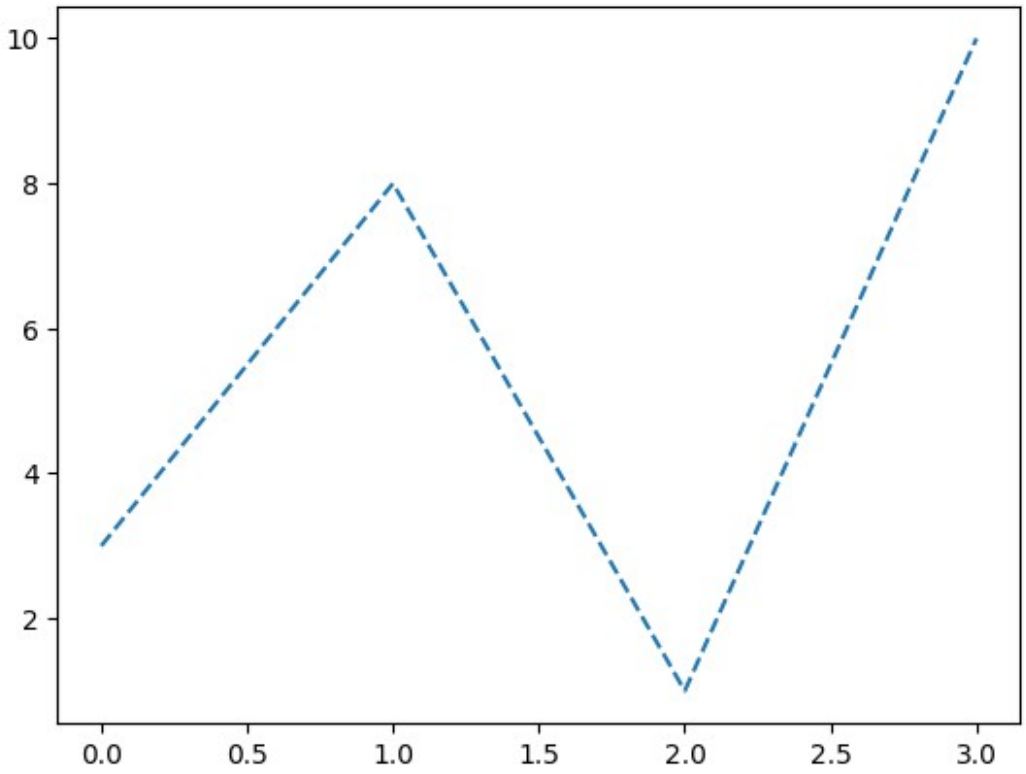

```
import matplotlib.pyplot as plt
import numpy as np

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

plt.plot(ypoints, linestyle = 'dotted')
plt.show()
```

```
[ 3  8  1 10]
```




```
plt.plot(ypoints, ls = '--')
```

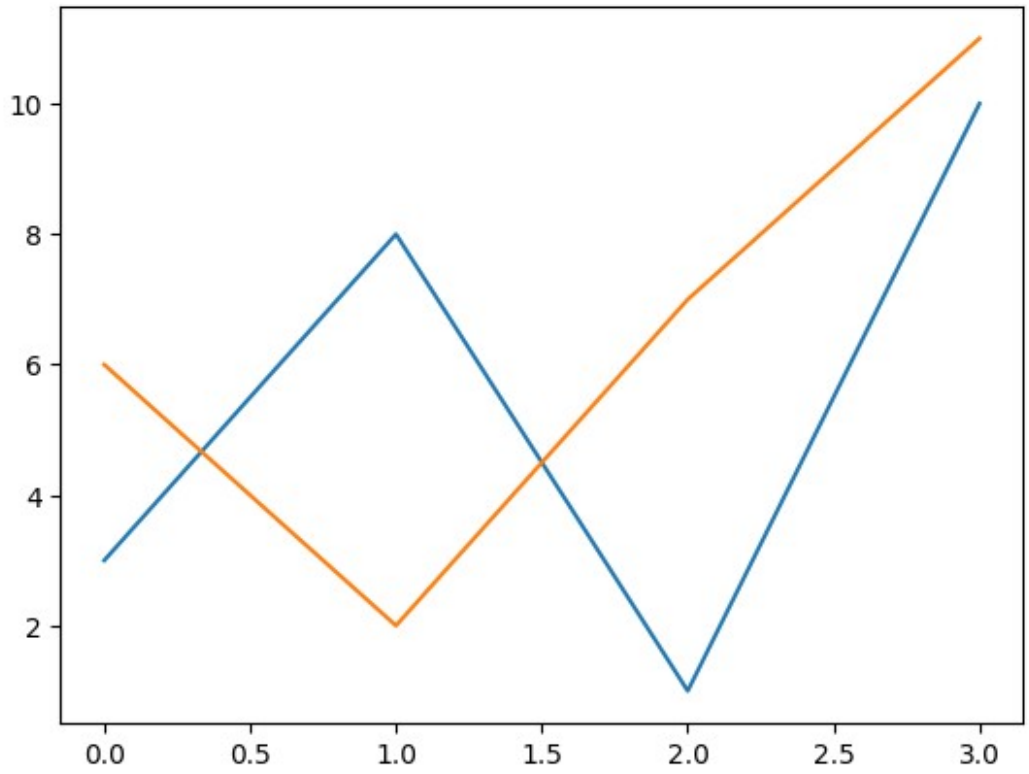
```
[<matplotlib.lines.Line2D at 0x15b81d5cf40>]
```



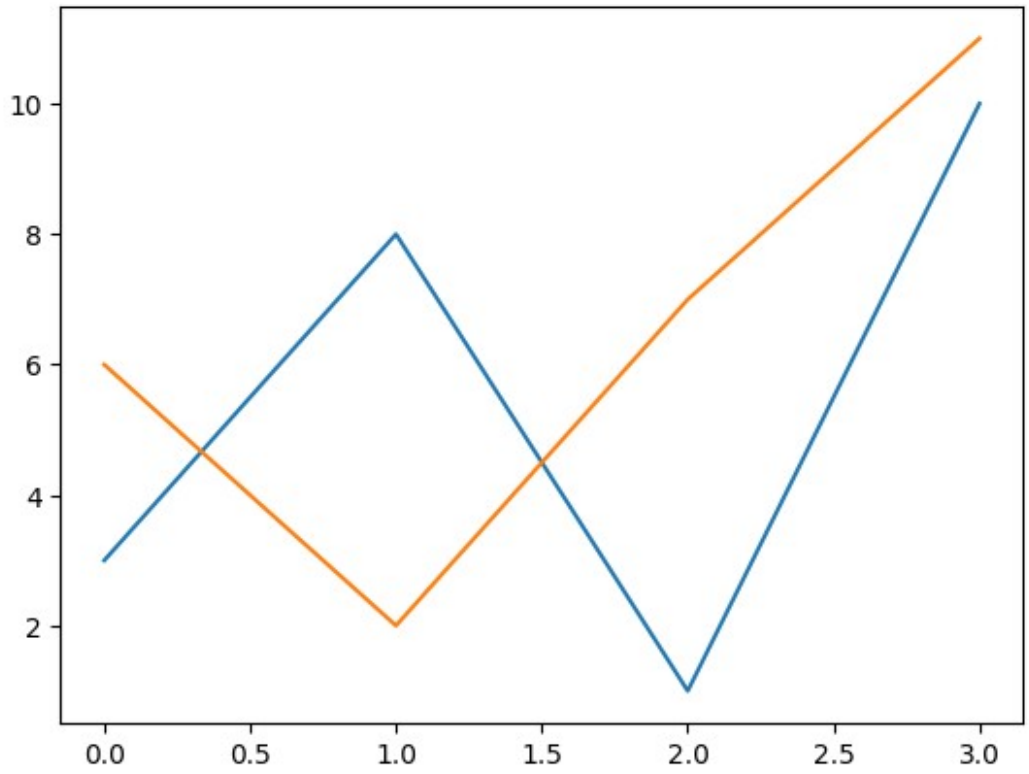
Multiple Lines

You can plot as many lines as you like by simply adding more `plt.plot()` functions:

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

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

plt.plot(y1)
plt.plot(y2)

plt.show()
```



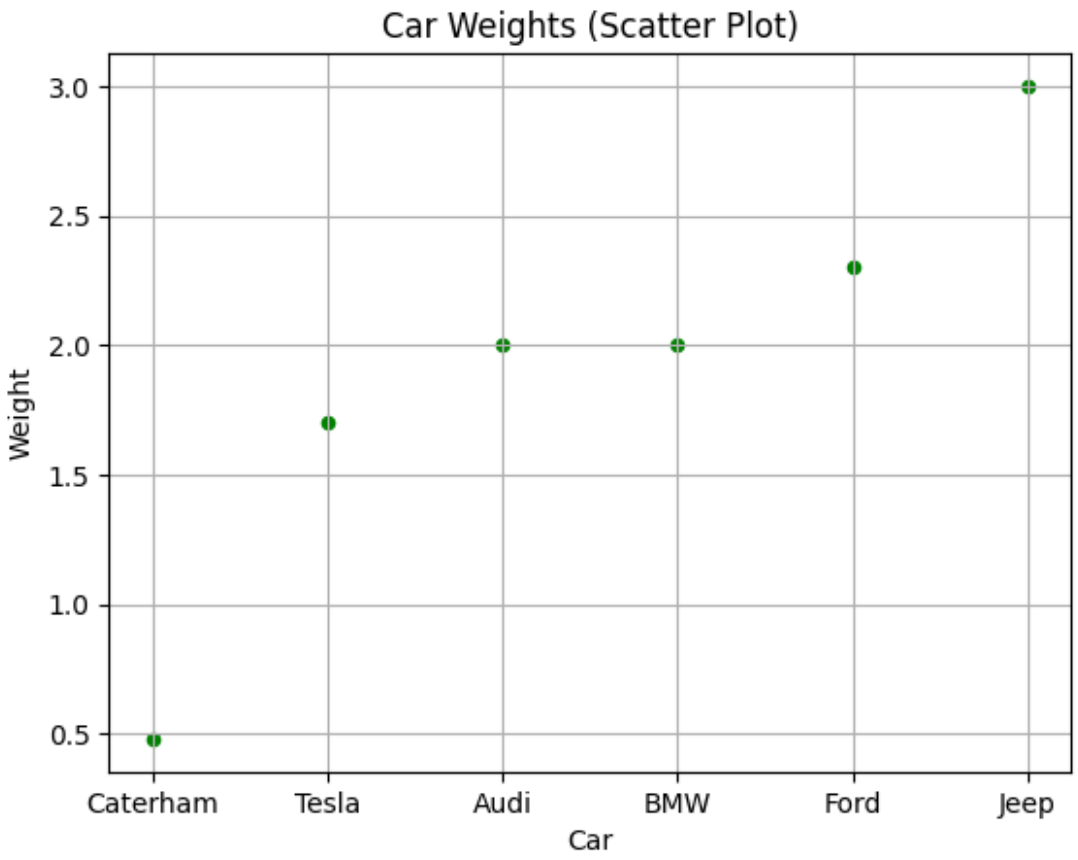

```
import matplotlib.pyplot as plt
import numpy as np

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()
```



Scatter Plots For Data Visualization

Scatter Plot displays data as a collection of points. We use the `plot()` function with `kind = 'scatter'` to scatter plot the data points. For example,

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

car = ["Caterham", "Tesla", "Audi", "BMW", "Ford", "Jeep"]
weight = [0.48, 1.7, 2, 2, 2.3, 3]

# create a DataFrame
data = {'Car': car, 'Weight': weight}
df = pd.DataFrame(data)

# scatter plot using Pandas
df.plot(x='Car', y='Weight', kind='scatter', marker='o',
color='green')
plt.xlabel('Car')
plt.ylabel('Weight')
plt.title('Car Weights (Scatter Plot)')
plt.grid(True)
plt.show()
```



In this example, we've used the `kind='scatter'` parameter in the `plot()` method to create a scatter plot.

The `marker` parameter is set to `'o'` to display circular markers, and the `color` parameter is set to `'blue'` to specify the marker color.

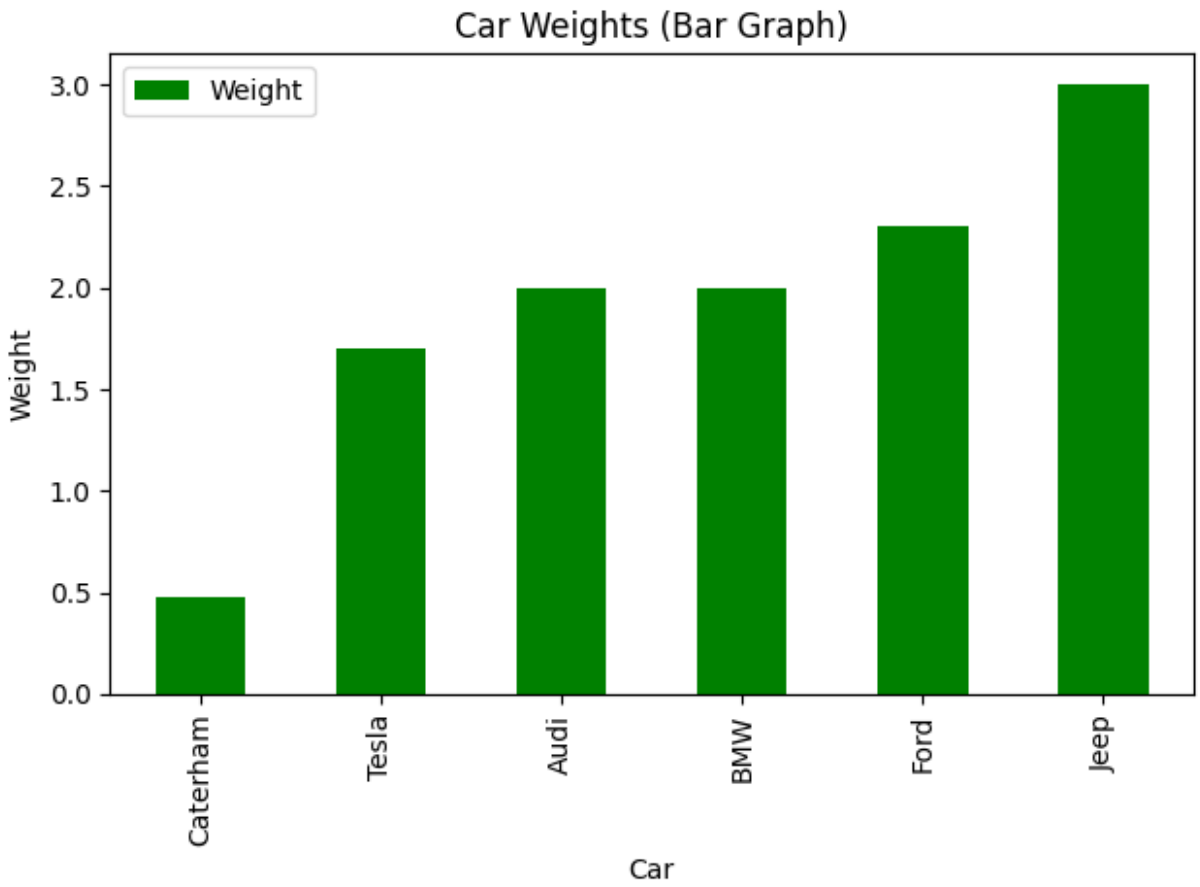
Bar Graphs For Data Visualization

Bar Graphs represent data using rectangular boxes. In Pandas, we pass`kind = 'scatter'` inside `plot()` to plot data in a bar graph.

Let's see an example.

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


# create a DataFrame
data = {'Car': ["Caterham", "Tesla", "Audi", "BMW", "Ford", "Jeep"],
        'Weight': [0.48, 1.7, 2, 2, 2.3, 3]
       }
df = pd.DataFrame(data)
```

```
# bar graph using Pandas
df.plot(x='Car', y='Weight', kind='bar', color='green')
plt.xlabel('Car')
plt.ylabel('Weight')
plt.title('Car Weights (Bar Graph)')
plt.tight_layout()
plt.show()
```



Here, we've used the `kind='bar'` parameter in the `plot()` method to create a bar graph. The `color` parameter is set to `'green'` to specify the color of the bars.

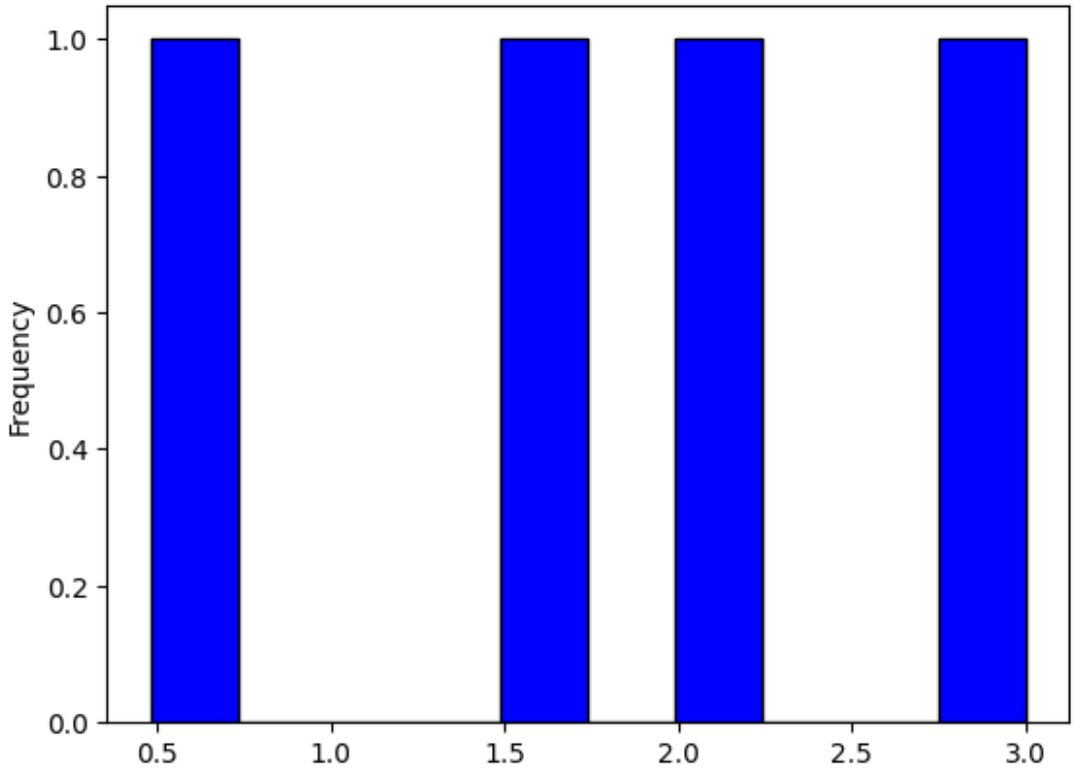
The `plt.tight_layout()` function is used to ensure that the plot layout is adjusted properly

Histograms For Data Visualization

In Pandas, we use `kind='hist'` inside `plot()` to create a histogram. For example,

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

weight = [0.48, 1.7, 2, 3]
```

```
# create a DataFrame
data = {'Weight': weight}
df = pd.DataFrame(data)

# histogram using Pandas
df['Weight'].plot(kind='hist', bins=10, edgecolor='black',
color='blue')
plt.show()
```



In this example, we created a histogram of the weights using the `plot()` method and then displayed it using `plt.show()`.