Plotting

Pandas uses the plot() method to create diagrams.

We can use Pyplot, a submodule of the Matplotlib library to visualize the diagram on the screen.

Following steps were followed:

1. Define the x-axis and corresponding y-axis values as lists.
2. Plot them on canvas using . plot() function.
3. Give a name to x-axis and y-axis using . xlabel() and . ylabel() functions.
4. Give a title to your plot using . title() function.
5. Finally, to view your plot, we use . show() function.

Matplotlib Pyplot

Matplotlib is a low level graph plotting library in python that serves as a visualization utility.

Matplotlib was created by John D. Hunter.

Matplotlib is open source and we can use it freely.

Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

Pyplot

Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias:

```
import matplotlib.pyplot as plt
```

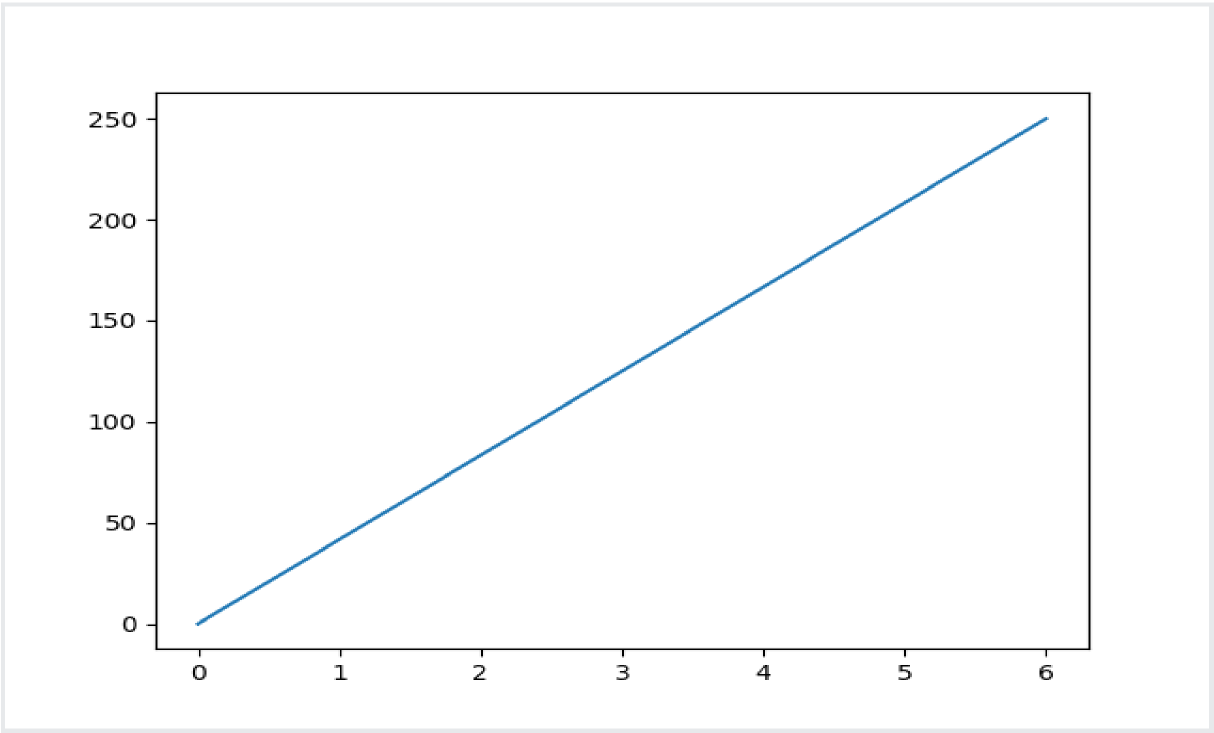
Now the Pyplot package can be referred to as plt.

Example

Draw a line in a diagram from position (0,0) to position (6,250):

```
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()
```

Result:



Types of Matplotlib in Python

- Bar Graph Using Matplotlib. The bar graphs are used in data comparison where we can measure the changes over a period of time. ...
- Histogram Using Matplotlib. ...
- Scatter Plot Using Matplotlib. ...
- Area Plot Using Matplotlib. ...
- Pie Chart Using Matplotlib.
- line plot Using Matplotlib

Matplotlib use

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy.

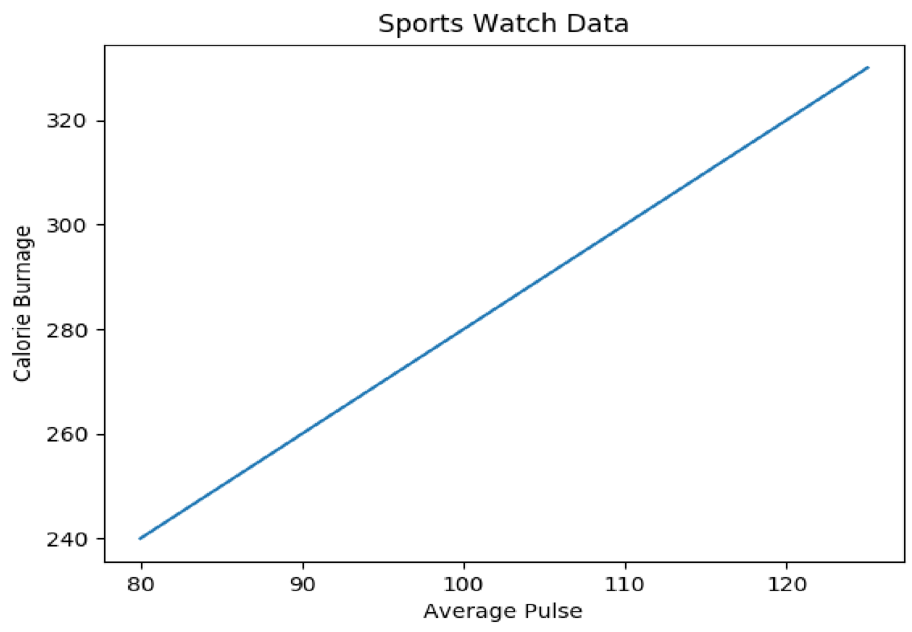
Matplotlib Labels and Title

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

With Pyplot, you can use the title() function to set a title for 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.plot(x, y)
plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
plt.show()
```

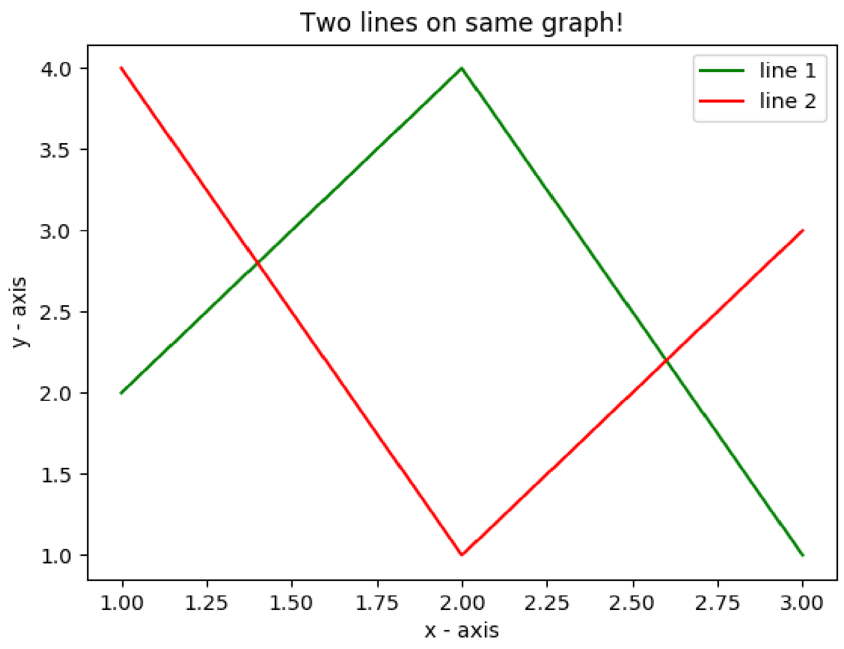
Result:



legend

- The small rectangular box giving information about the type of line and its color is called a legend. We can add a legend to our plot using **.legend()** function.

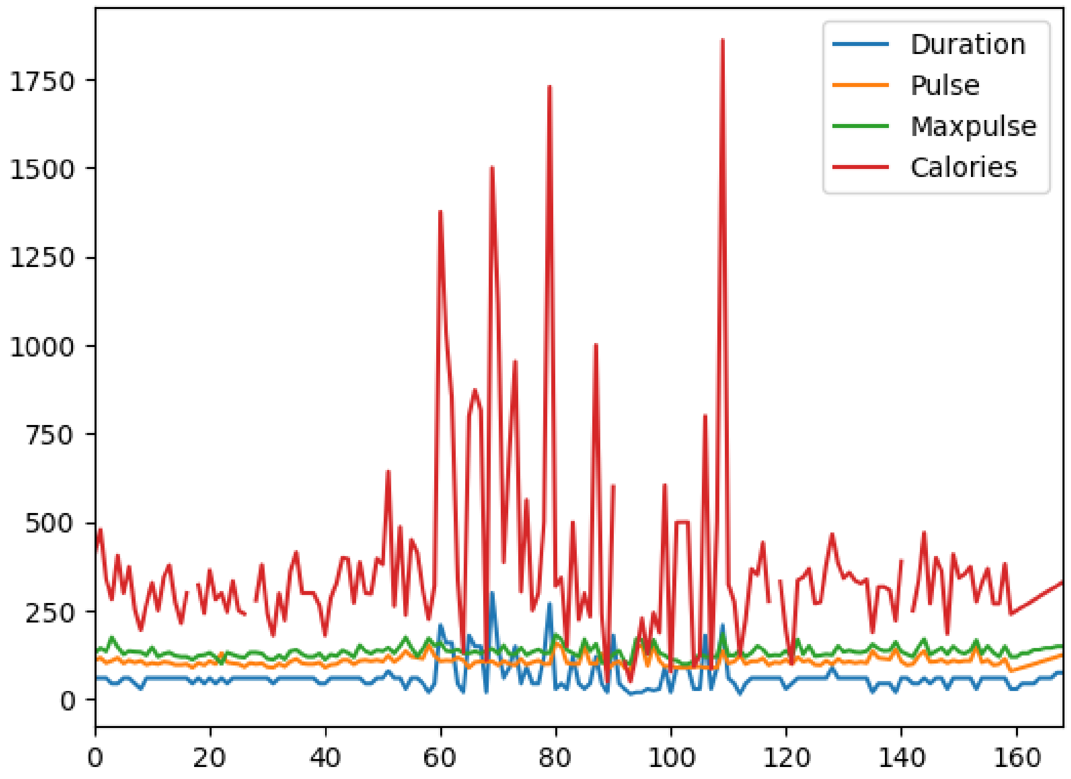
```
plt.legend()
```



Example

Import pyplot from Matplotlib and visualize our DataFrame:

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('data.csv')
df.plot()
plt.show()
```



Data.csv

Scatter Plot

Specify that you want a scatter plot with the kind argument:

kind = 'scatter'

A scatter plot needs an x- and a y-axis.

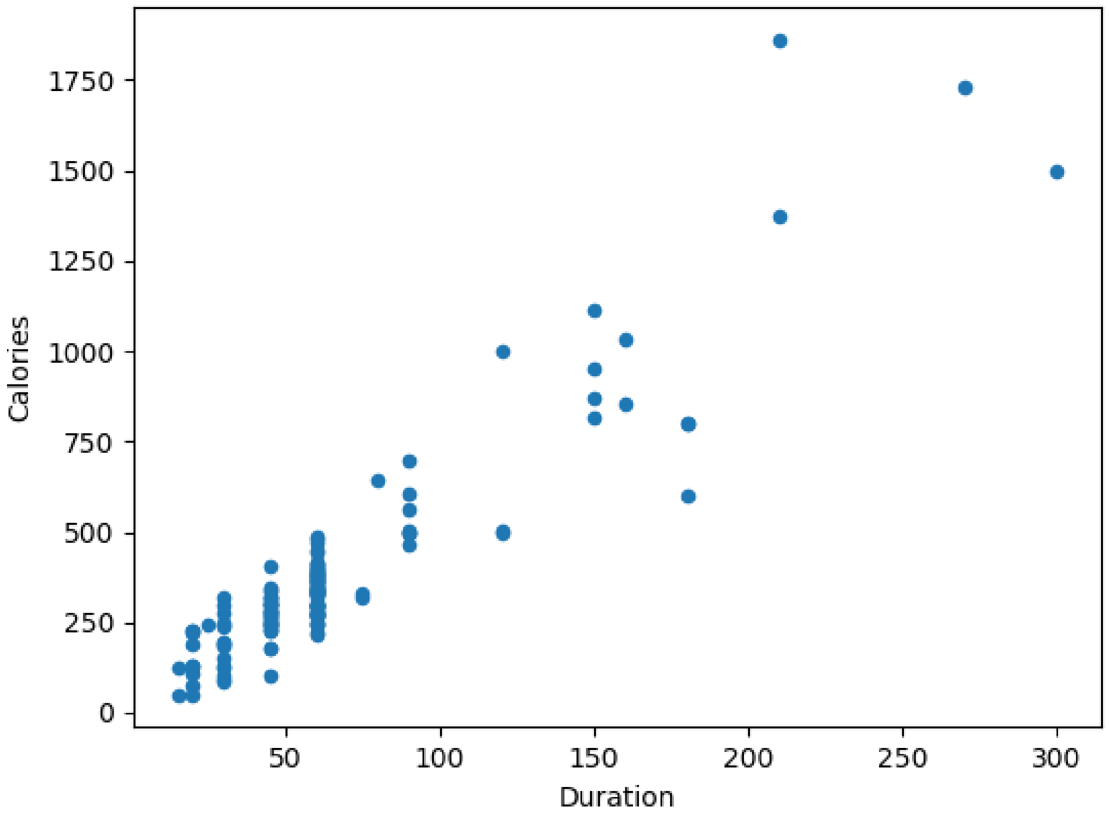
In the example below we will use "Duration" for the x-axis and "Calories" for the y-axis.

Include the x and y arguments like this:

x = 'Duration', y = 'Calories'

Example

```
import pandas as pd

import matplotlib.pyplot as plt

df = pd.read_csv('data.csv')

df.plot(kind = 'scatter', x = 'Duration', y = 'Calories')

plt.show()
```



Histogram

Use the kind argument to specify that you want a histogram:

kind = 'hist'

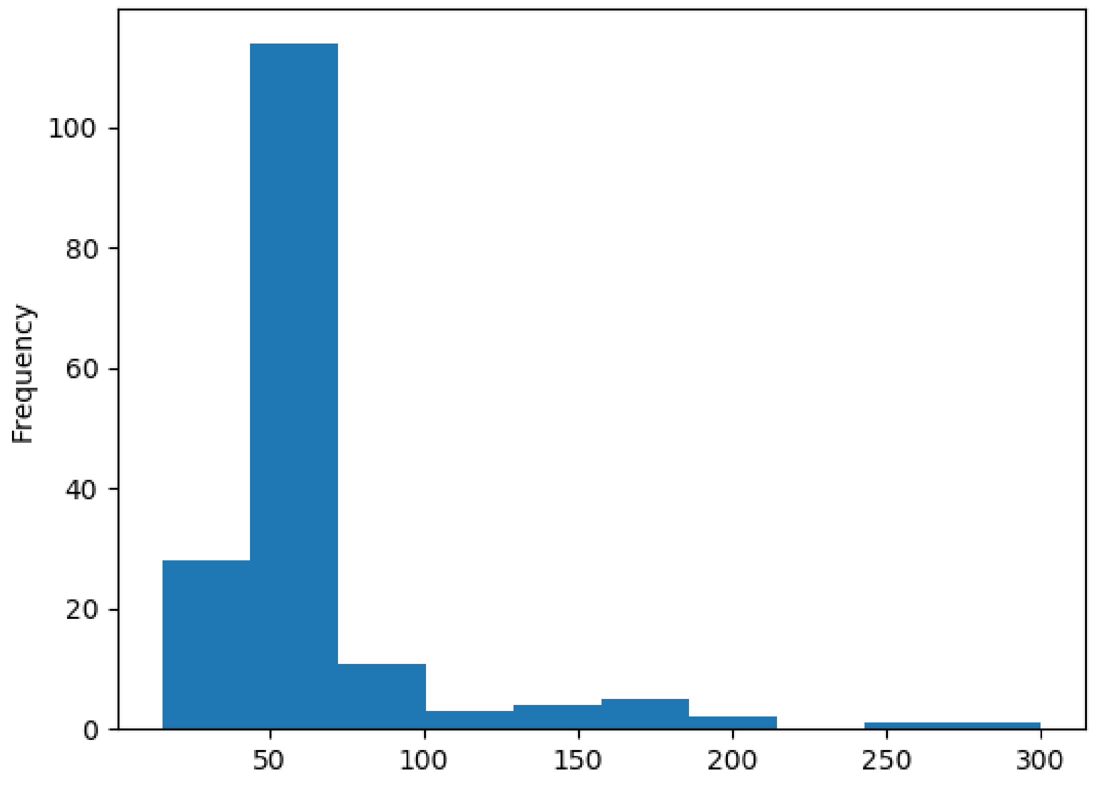
A histogram needs only one column.

A histogram shows us the frequency of each interval, e.g. how many workouts lasted between 50 and 60 minutes?

In the example below we will use the "Duration" column to create the histogram:

Example

```
df["Duration"].plot(kind = 'hist')
```



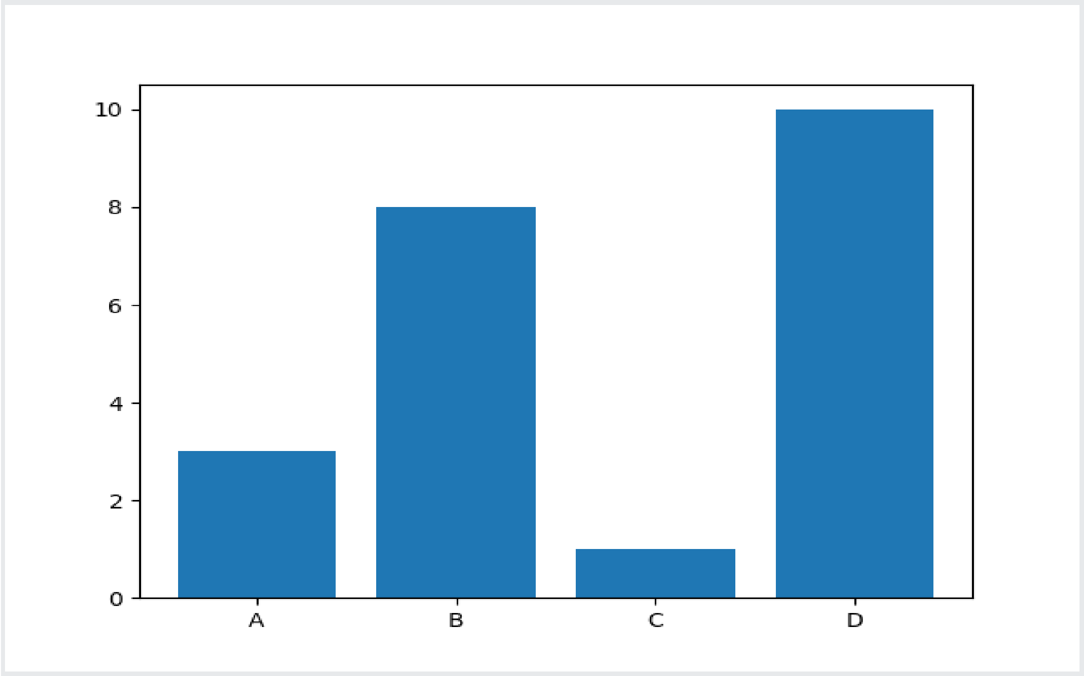
Matplotlib Bars

With Pyplot, you can use the bar() function to draw bar graphs:

Example

```
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()
```



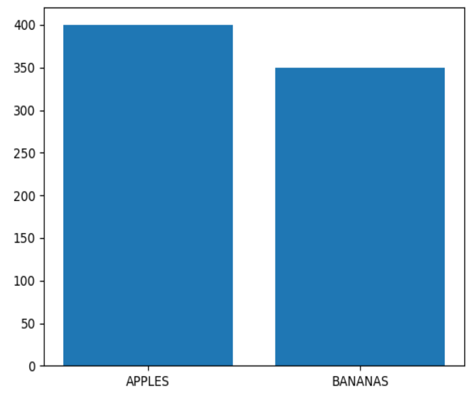
The bar() function takes arguments that describes the layout of the bars.

The categories and their values represented by the *first* and *second* argument as arrays.

Example

```
x = ["APPLES", "BANANAS"]

y = [400, 350]
```

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



Horizontal Bars

If you want the bars to be displayed horizontally instead of vertically, use the barh() function:

Creating Pie Charts

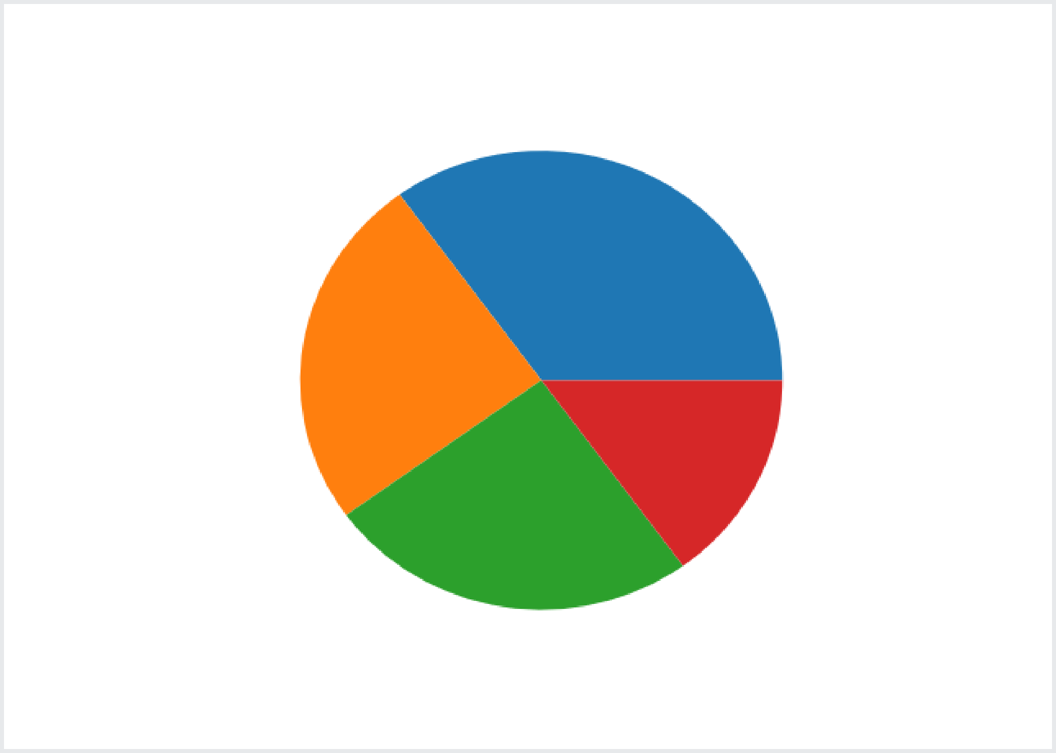
With Pyplot, you can use the pie() function to draw pie charts:

Example

A simple pie chart:

```
import matplotlib.pyplot as plt

import numpy as np

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

plt.pie(y)

plt.show()
```

Result:



As you can see the pie chart draws one piece (called a wedge) for each value in the array (in this case [35, 25, 25, 15]).

By default the plotting of the first wedge starts from the x-axis and move *counterclockwise*:

Labels

Add labels to the pie chart with the label parameter.

The label parameter must be an array with one label for each wedge:

Example

A simple pie chart:

```
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

As mentioned the default start angle is at the x-axis, but you can change the start angle by specifying a startangle parameter.

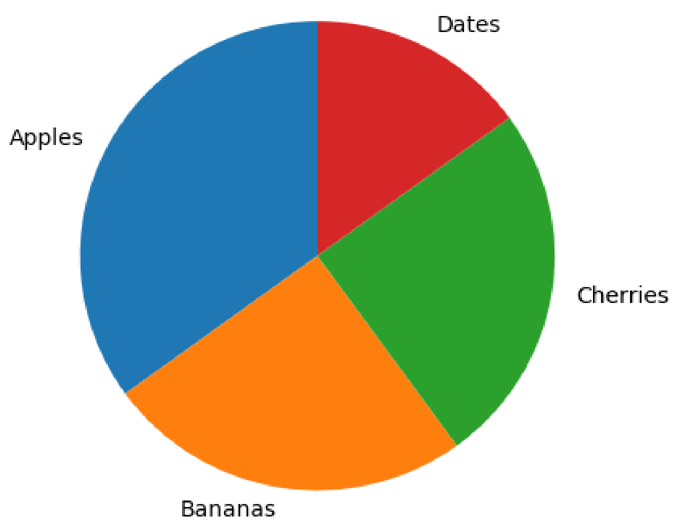
The startangle parameter is defined with an angle in degrees, default angle is 0:

Example

Start the first wedge at 90 degrees:

```
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()
```

Result:



Explode

Maybe you want one of the wedges to stand out? The explode parameter allows you to do that.

The explode parameter, if specified, and not None, must be an array with one value for each wedge.

Each value represents how far from the center each wedge is displayed:

Shadow

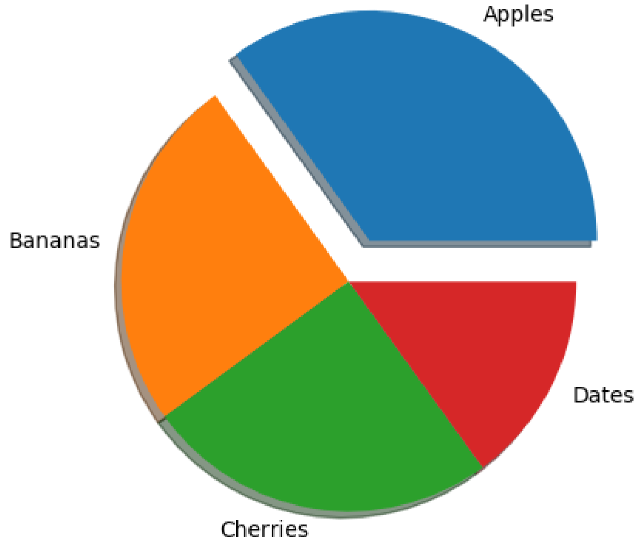
Add a shadow to the pie chart by setting the shadows parameter to True:

Example

```
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()
```

Result:



Colors

You can set the color of each wedge with the colors parameter.

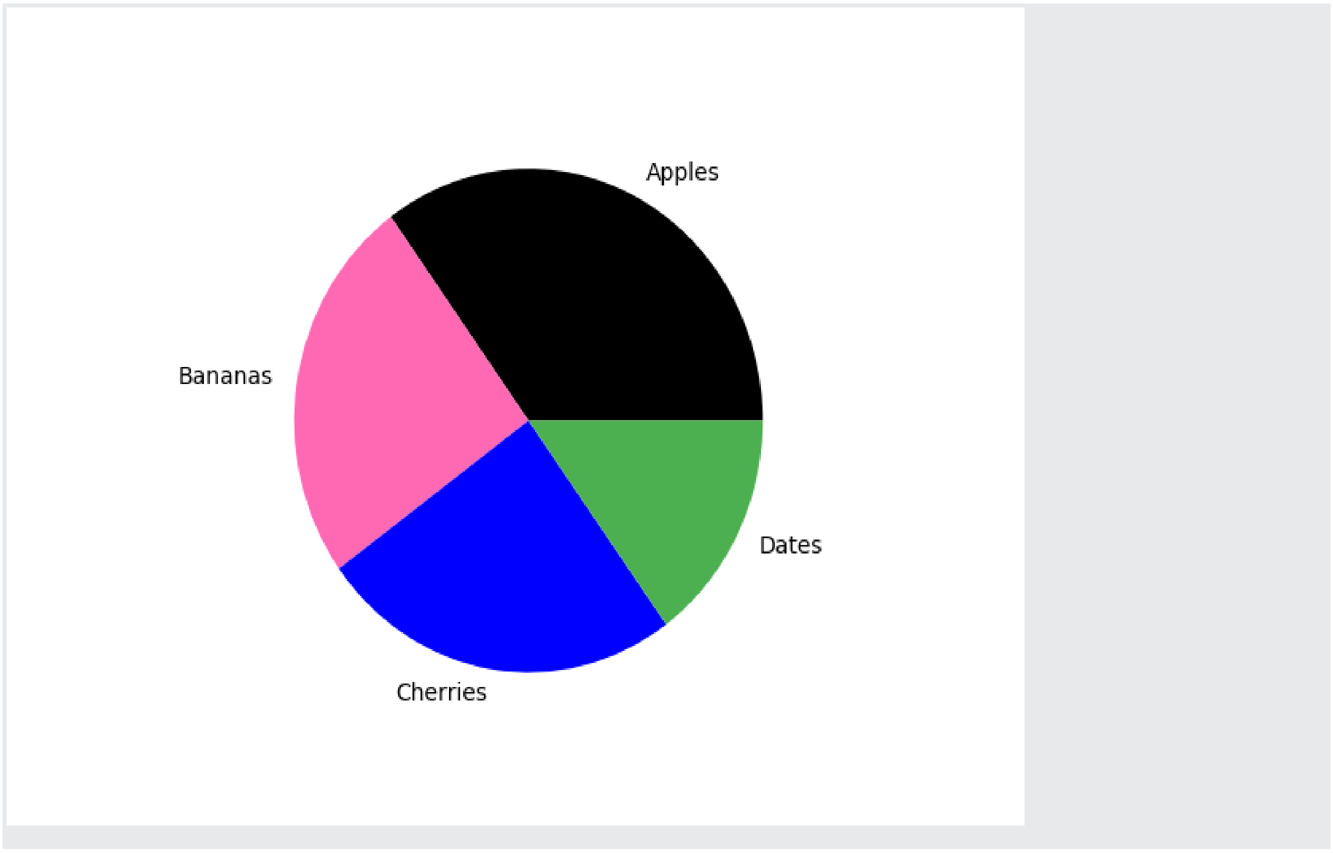
The colors parameter, if specified, must be an array with one value for each wedge:

Example

Specify a new color for each wedge:

```
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()
```

Result:


Apples
Bananas
Cherries
Dates