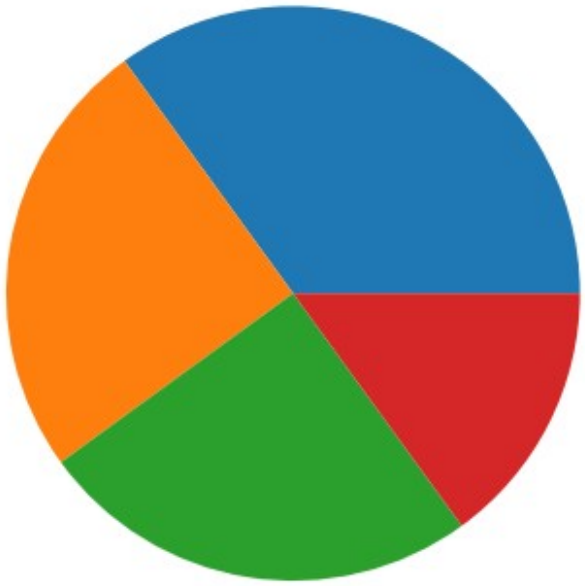
Matplotlib Pie Charts

Creating 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()
```

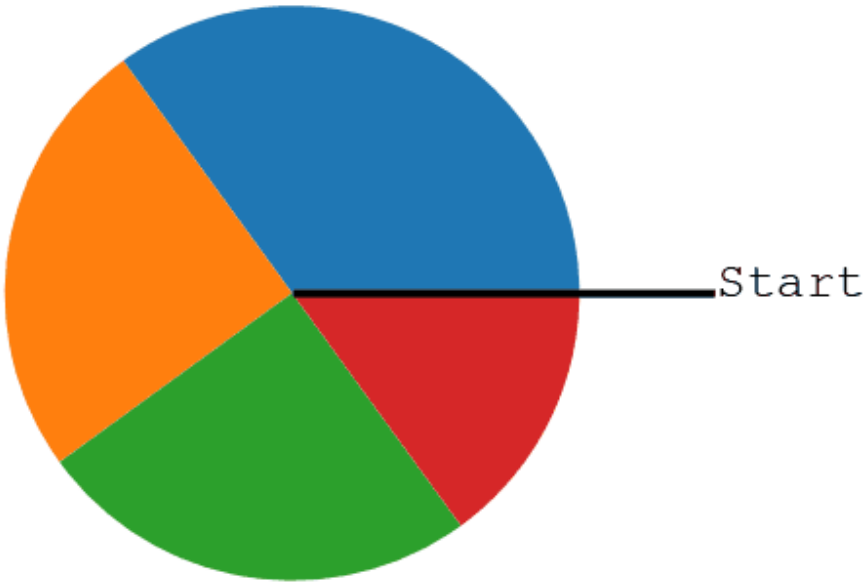


As you can see the pie chart draws one piece (called a wedge) for each value in the array (in this case [35, 25, 25, 15]).

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

Note: The size of each wedge is determined by comparing the value with all the other values, by using this formula:

The value divided by the sum of all values: x/sum(x)

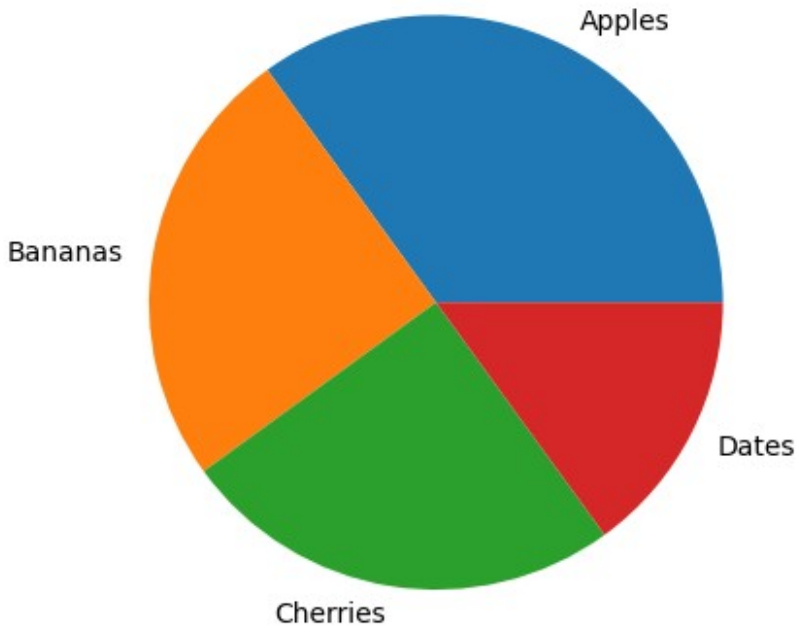


Labels

Add labels to the pie chart with the `labels` parameter.

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

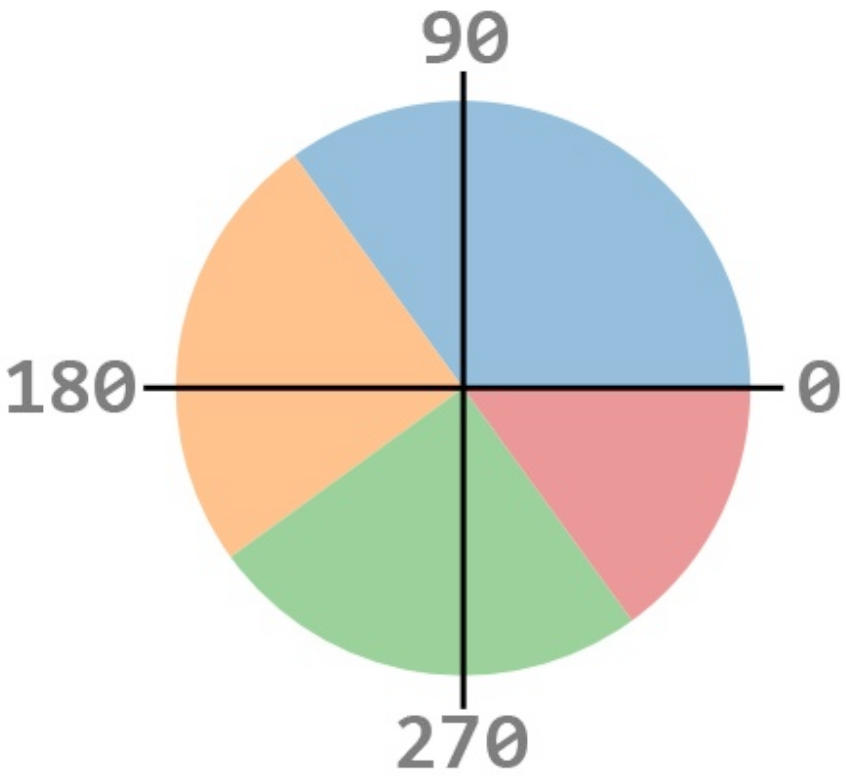
```
#A simple pie chart:

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()
```



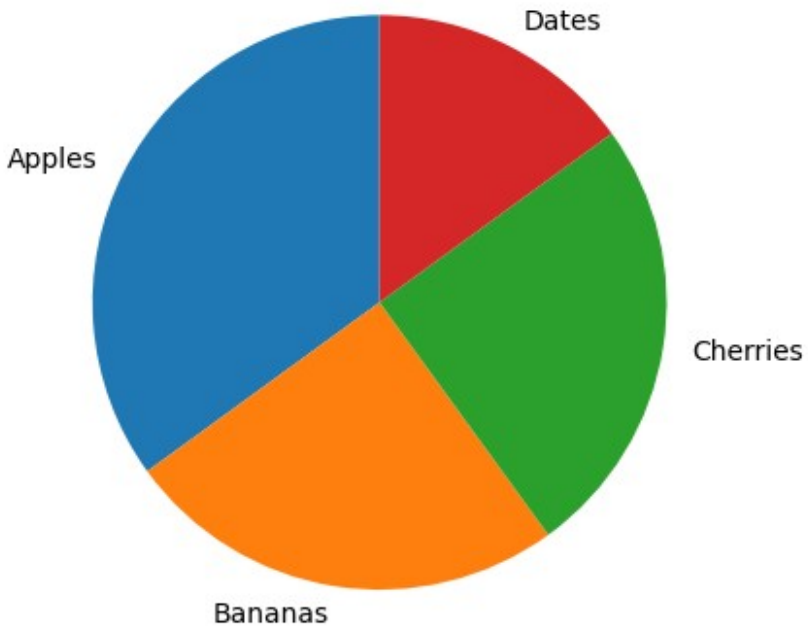
Start Angle

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:




```
#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()
```



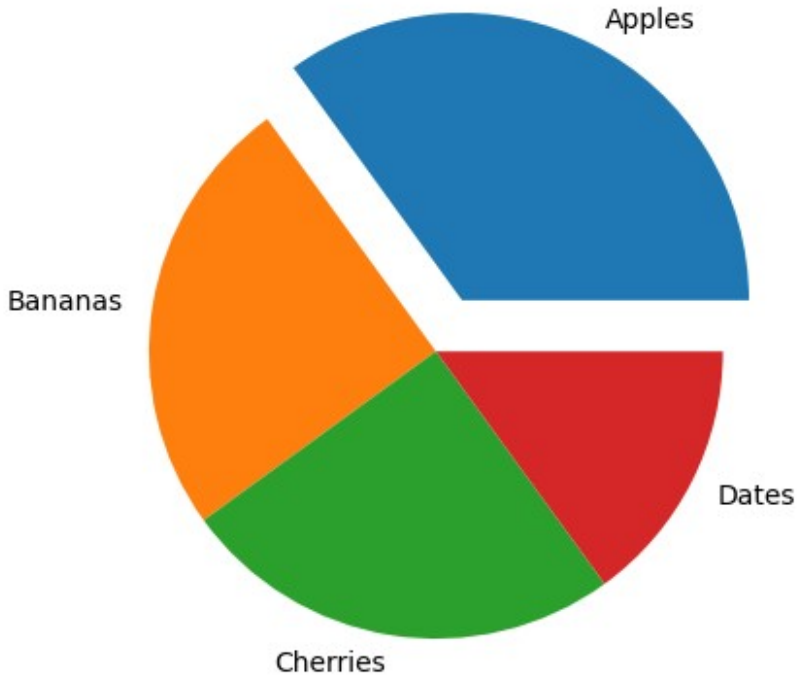
Explode

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:

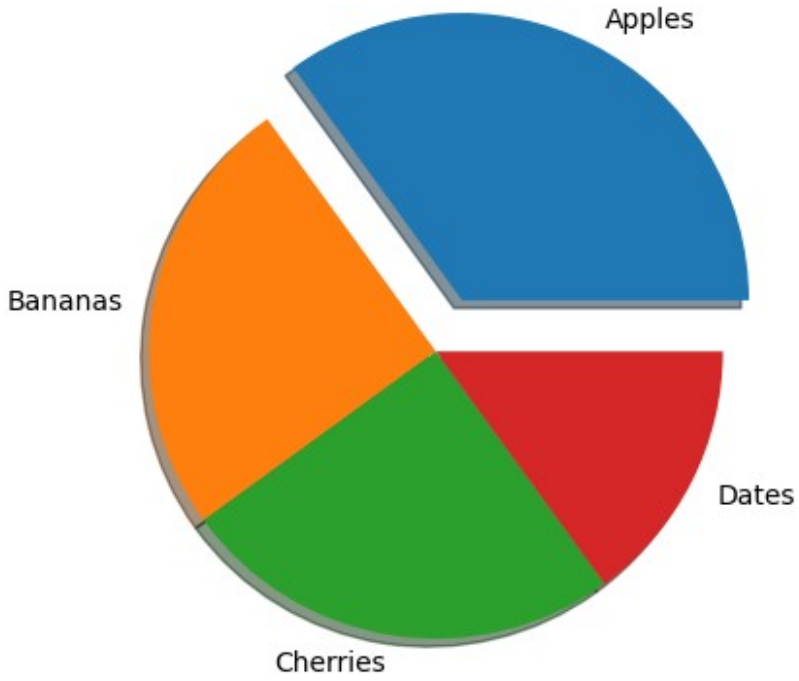
```
#Pull the "Apples" wedge 0.2 from the center of the pie:

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

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

```
#Add a 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()
```



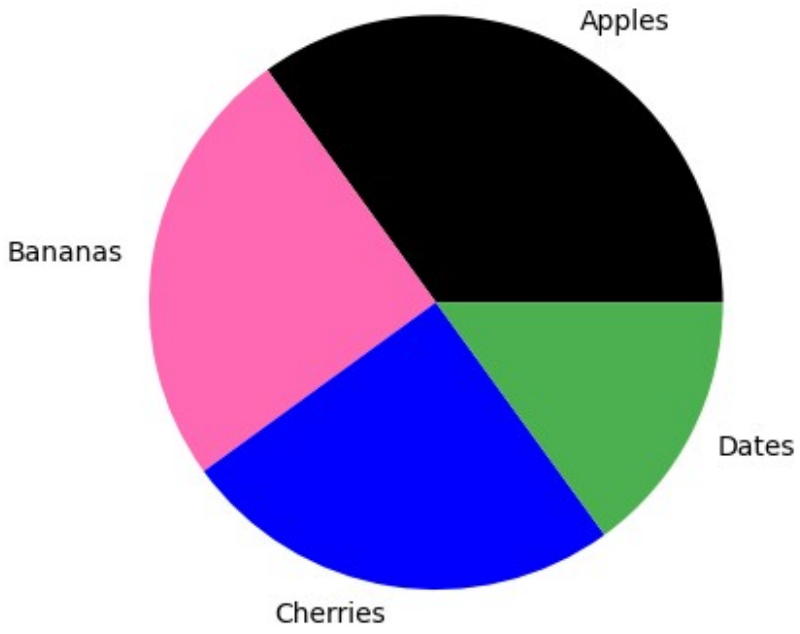
Colors

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:

'r' - Red, 'g' - Green, 'b' - Blue, 'c' - Cyan, 'm' - Magenta, 'y' - Yellow, 'k' - Black, 'w' - White

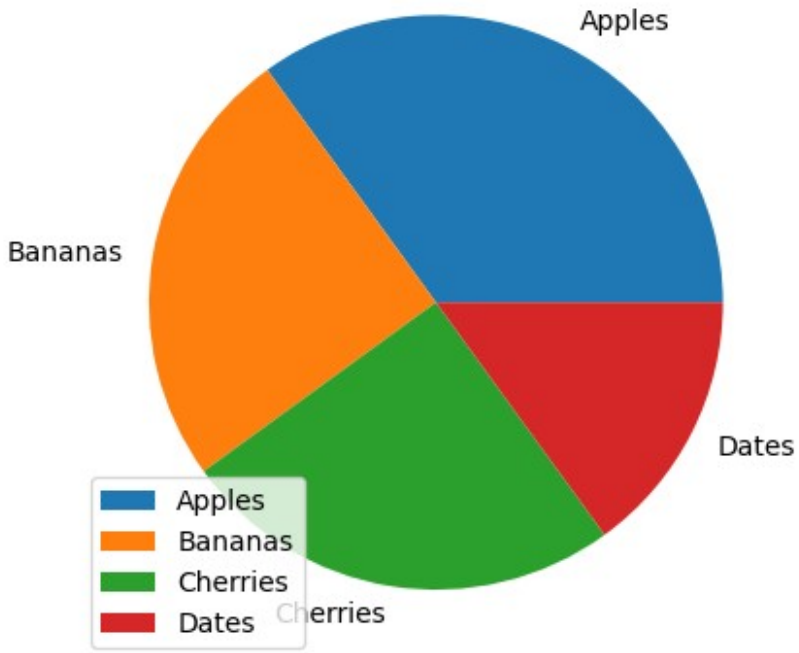
```
import matplotlib.pyplot as plt
import numpy as np

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

To add a list of explanation for each wedge, use the `legend()` function:

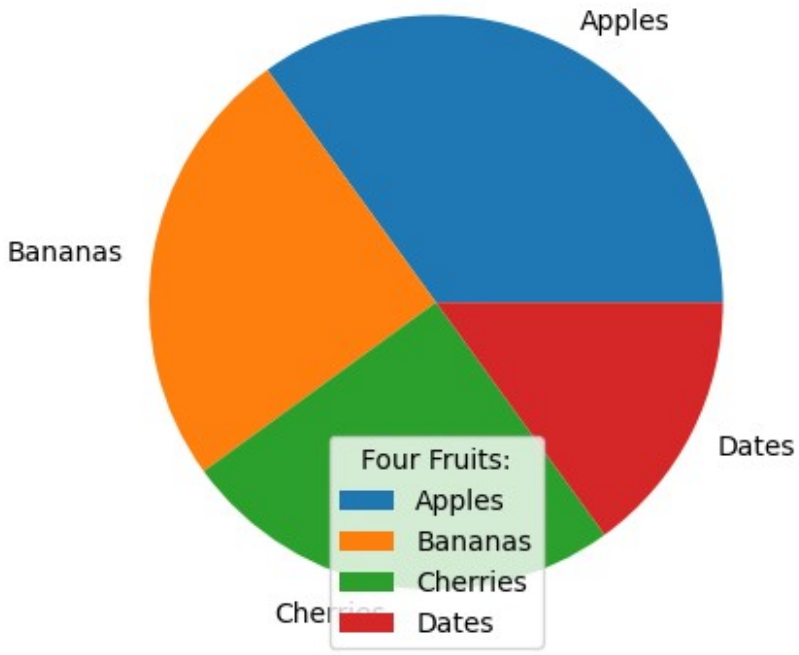
```
#Add a legend:

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

To add a header to the legend, add the `title` parameter to the `legend` function.

```
#Add a legend with a header:

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(title = "Four Fruits:")
plt.show()
```

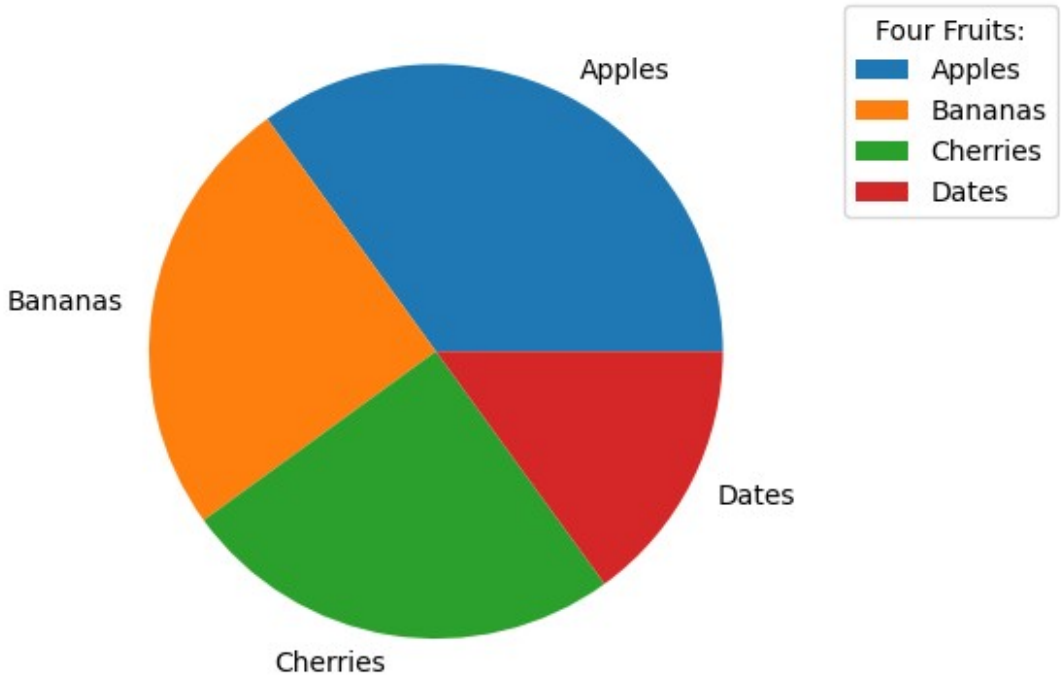


To change the position of the legend in your pie chart, you can use the `bbox_to_anchor` parameter in the `plt.legend()` function. This parameter allows you to specify the position of the legend relative to the plot.

Here's how you can modify your code to change the position of the legend:

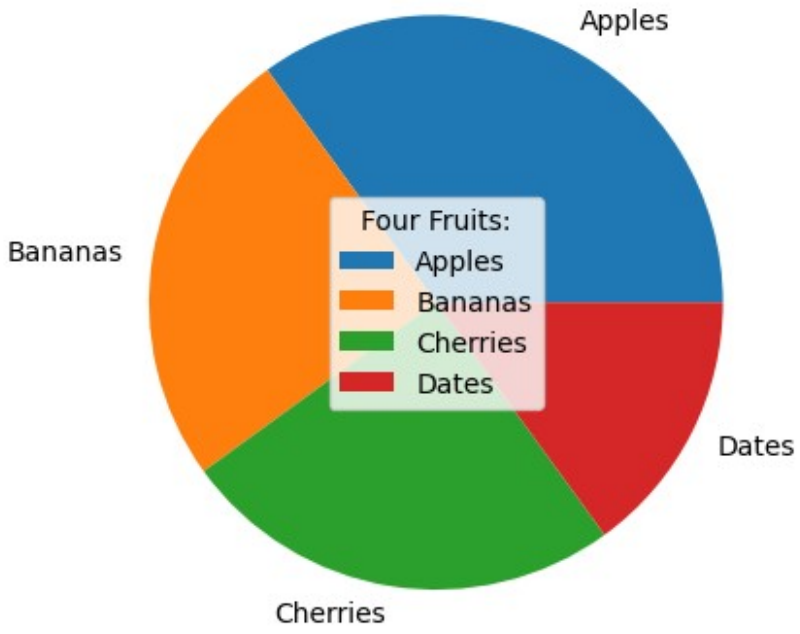
1. **Position the legend outside the pie chart:** This will place the legend outside the pie chart on the right side.

```
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(title="Four Fruits:", bbox_to_anchor=(1.05, 1), loc='upper
left')
plt.show()
```

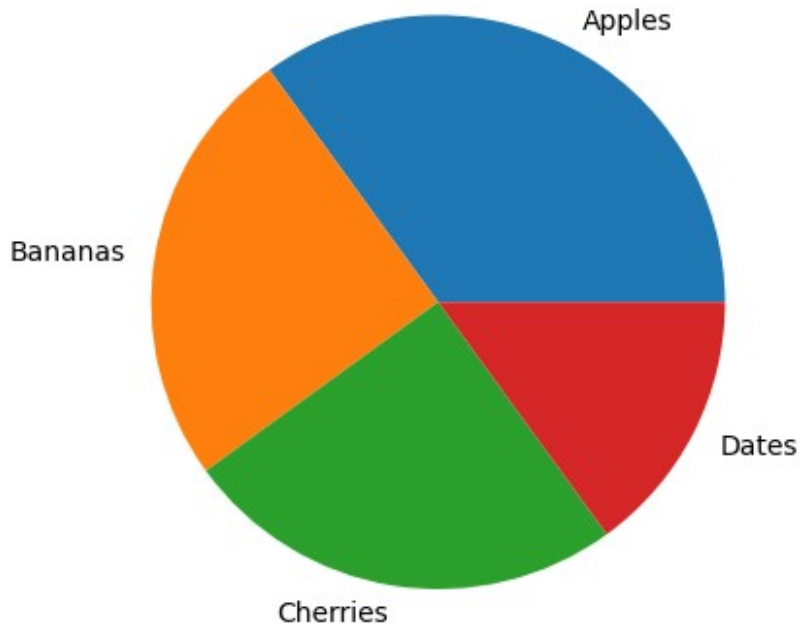


1. **Position the legend inside the pie chart:** This will place the legend inside the pie chart at a specified position.

```
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(title="Four Fruits:", bbox_to_anchor=(0.5, 0.5),
loc='center')
plt.show()
```




```
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(title="Four Fruits:", bbox_to_anchor=(0.5, -0.1),
loc='upper center', ncol=4)
plt.show()
```



In these examples:

- `bbox_to_anchor` specifies the position of the legend relative to the plot.
- `loc` specifies the anchor point of the legend.
- `ncol` is used to specify the number of columns for the legend (useful for arranging the legend items neatly). You can adjust the `bbox_to_anchor` values to place the legend wherever you prefer.