Learning Objectives

- **Learn how to add Visualizations to your plots**
- **Create a pie chart**
- **Generate a donut chart**
- **Present multiple charts in Matplotlib**

definition

Assumptions

- Learners are comfortable working around Jupyter and Matplotlib.

Limitations

In this assignment, we only work with the Matplotlib library.
* In this section we will cover visualization functions function using bar and line plots which we already covered in the last module to start creating charts right away.

Pie Charts

Last time we covered bar, column and line graphs. Another excellent visualization tool is pie charts.

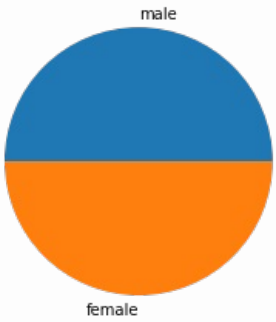
Each slice of the pie chart represents one component and all slices added together equal the whole.

> Pie charts are generally used to demonstrate categorical or nominal data. Usually, the slices of the pie represent different groups of a data.

For example, a class of 50 students and the class is divided into 25 Females and 25 males. Another example is a school being split into grade classes (freshman, sophomore, junior and senior).

Creating Pie Charts

```
import matplotlib.pyplot as plt

groups2=['male','female']

number=[25,25]

# Creating plot

plt.pie(number, labels = groups2)

# show plot
plt.show()
```



images/plot6

Let's change the proportion of students in the class. Use the code below to compare our graphs.

```
number=[30,20]
```

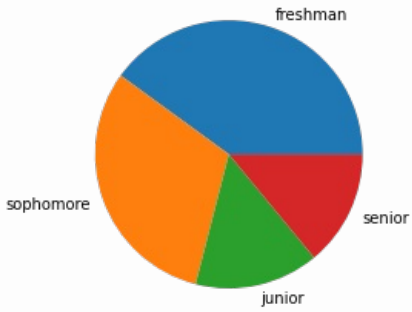
The basic syntax is:

```
plt.pie( numbers,labels)
```

Add the following code into the text editor and this time we will work with a larger group. In this example the school size is 100 students

```
groups2= ['freshman','sophomore','junior','senior']
total=[40,31,15,14]
plt.pie(total, labels = groups2)
plt.show()
```

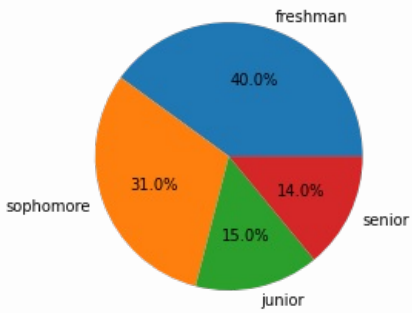
Plot Result:



images/Capture1

We can add the percentages by adding an extra option in our `plt.pie`:

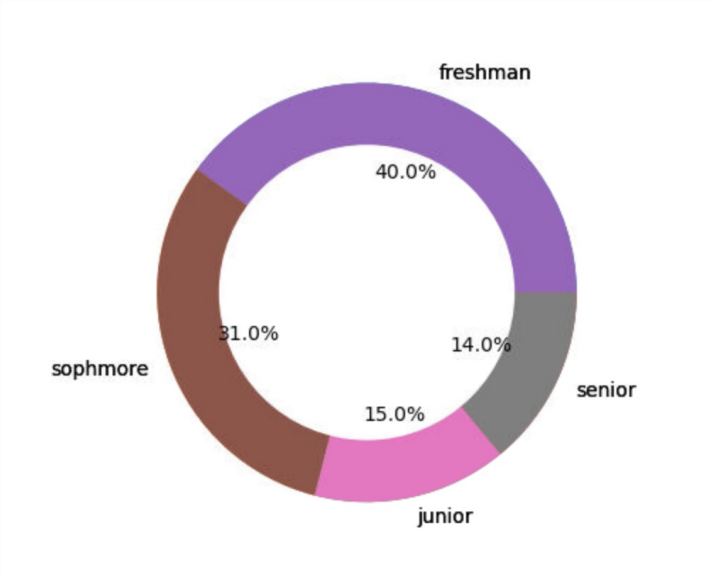
```
plt.pie(total, labels = groups2, autopct='%1.1f%%')
```




images/plot9

Donut Chart

A **Donut chart** is a Pie Chart with an area of the center cut out. We use donut charts to show the proportions of categorical data.



images/Donut

Creating Donut Charts

Because donut charts are essentially pie chart, in order to create them we are going to use the same function we used to create pie charts but with just some small changes to it. What we are going to do is create a pie chart then we are going to add a small circle in the middle of it. The following code is used to create our pie chart.

```
groups2= ['freshman','sophomore','junior','senior']
total=[40,31,15,14]
# Creating plot

plt.pie(total, labels = groups2)
# show plot
plt.show()
```

The way we are going to tackle this is by adding an extra argument to the `plt.pie()` function `wedgeprops=dict(width=0.6)`. Where width is how large the circle is in the middle.

```
plt.pie(total, labels = groups2,wedgeprops=dict(width=0.3))
```

We can further add the percentages using the following code:

```
plt.pie(total, labels =
        groups2,wedgeprops=dict(width=0.3),autopct='%1.1f%%')
```

Another method

Another way to tackle it is using the following method. Where we don't give it the extra argument.

```
plt.pie(total, labels = groups2)
```

Now we are going to literally add a circle to the middle of our current pie chart to transform into a donut chart before we show our plot. The `plt.show()` function anything has it that the command will create a new figure.

```
# adding circle in the middle
my_circle=plt.Circle( (0,0), 0.7, color='white')
p=plt.gcf()
p.gca().add_artist(my_circle)
```

Multiple Charts

Previously, we worked with multiple line charts and donut charts. For both of those we use the idea, of simply adding to the current graph that we are currently using. Let's suppose we want to present the following data:

```
groups2= ['freshman','sophomore','junior','senior']
total=[140,31,25,10]
total2=[150,20,16,12]
```

We use the same techniques it will plot the results one over the other. Therefore we will be missing part of the data we want to present. Try the following

```
plt.pie(total, labels = groups2, autopct='%1.1f%%')
plt.pie(total2, labels = groups2, autopct='%1.1f%%')
plt.show()
```

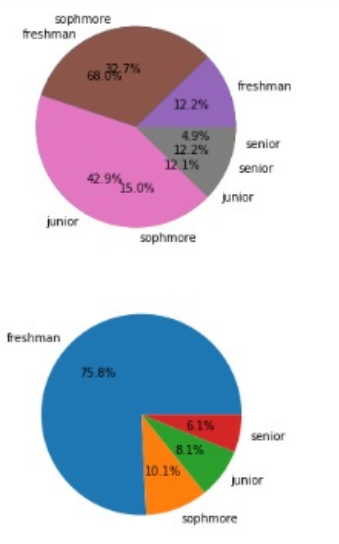
Our solution to this issue will be using the `.figure()` function. This function let separate the different plots base on our need. Here is an application using `.figure()`.

```
plt.figure(0)
plt.pie(total, labels = groups2, autopct='%1.1f%%')
plt.figure(2)
plt.pie(total2, labels = groups2, autopct='%1.1f%%')
plt.show()
```

- Please note that without putting the number as argument for figures, it will still separate the figures. However, it will be good habit to label them so you can have more control and cleaner code.

Assuming you wanted to merge to figure it is possible by simply giving them the same number as argument.

```
groups2= ['freshman','sophomore','junior','senior']
total=[140,31,25,10]
total2=[150,20,16,12]
total3=[12,32,42,12]
plt.figure(1)
plt.pie(total, labels = groups2, autopct='%1.1f%%')
plt.figure(2)
plt.pie(total2, labels = groups2, autopct='%1.1f%%')
plt.figure(1)
plt.pie(total3, labels = groups2, autopct='%1.1f%%')
plt.show()
```



images/Capture2