Files

Python uses file objects to interact with external files on your computer. These file objects can be any sort of file you have on your computer, whether it be an audio file, a text file, emails, Excel documents, etc. Note: You will probably need to install certain libraries or modules to interact with those various file types, but they are easily available. (We will cover downloading modules later on in the course).

Python has a built-in open function that allows us to open and play with basic file types. First we will need a file though. We're going to use some iPython magic to create a text file!

iPython Writing a File

In [6]:
```
%%writefile test.txt
Hello, this is a quick test file
```

```
Overwriting test.txt
```

Python Opening a file

We can open a file with the open() function. The open function also takes in arguments (also called parameters). Lets see how this is used:

In [14]:
```
# Open the text.txt we made earlier
my_file = open('test.txt')
```

In [15]:
```
# We can now read the file
my_file.read()
```

Out[15]:
```
'Hello, this is a quick test file'
```

In [16]:
```
# But what happens if we try to read it again?
my_file.read()
```

Out[16]:
```
''
```

This happens because you can imagine the reading "cursor" is at the end of the file after having read it. So there is nothing left to read. We can reset the "cursor" like this:

In [42]:
```
# Seek to the start of file (index 0)
my_file.seek(0)
```

In [19]:
```
# Now read again
my_file.read()
```

Out[19]:
```
'Hello, this is a quick test file'
```

In order to not have to reset every time, we can also use the readlines method. Use caution with large files, since everything will be held in memory. We will learn how to iterate over large files later in the course.

In [26]:
```
# Readlines returns a list of the lines in the file.
my_file.readlines()
```

Out[26]:
```
['Hello, this is a quick test file']
```

Writing to a File

By default, using the open() function will only allow us to read the file, we need to pass the argument 'w' to write over the file. For example:

In [39]:
```
# Add a second argument to the function, 'w' which stands for write
my_file = open('test.txt','w+')
```

In [40]:
```
# Write to the file
my_file.write('This is a new line')
```

In [43]:
```
# Read the file
my_file.read()
```

Out[43]:
```
'This is a new line'
```

Iterating through a File

Lets get a quick preview of a for loop by iterating over a text file. First let's make a new text file with some iPython Magic:

In [44]:
```
%%writefile test.txt
First Line
Second Line
```

```
Overwriting test.txt
```

Now we can use a little bit of flow to tell the program to for through every line of the file and do something:

In [45]:
```
for line in open('test.txt'):
    print line
```

```
First Line

Second Line
```

Don't worry about fully understanding this yet, for loops are coming up soon. But we'll break down what we did above. We said that for every line in this text file, go ahead and print that line. Its important to note a few things here:

1.) We could have called the 'line' object anything (see example below).
2.) By not calling .read() on the file, the whole text file was not stored in memory.
3.) Notice the indent on the second line for print. This whitespace is required in Python.

We'll learn a lot more about this later, but up next: Sets and Booleans!

In [46]:
```
# Pertaining to the first point above
for asdf in open('test.txt'):
    print asdf
```

```
First Line

Second Line
```

In [ ]:
```

```