Files

Files are used to store data. Whenever you are saving anything on a computer, it is being saved as a file. When I type .txt at the end of a file, the program reading it will know it is a *text file*. If I write .py at the end, it will know it is a *Python file*. If I write .cpp at the end, it will know it is a *C++ file*.

Writing a file

The best way to understand file writing is to show an example. There isn't much math or logic to it unless you know some programming knowledge. Luckily we just finished talking about lists and loops so we should be fine.

Let's pretend we want to create a file called "names.txt" and add some names to it.

```
names = ["Michael", "Jill", "Rebecca", "Thomas"]
ourFile = open("names.txt","w") # 'w' for 'writing'
for name in names:
  ourFile.write(name)
ourFile.close()
```

No matter what you are doing with files, you must **open** the file and you must **close** the file. If you do not close the file, the file is possibly not saved or not saved correctly. Like when you are told my your computer to eject the USB stick before removing it, except it matters this time.

For opening a file, you have the *name* of the file and then the *mode* that it is opened in. Our name is `names.txt` and the mode is `w` which stands for "writing".

Then we go through every `name` in `names` and `write` it to `ourFile`. Every time we use the `write()` function, the function takes the argument (in this case, `name`) and adds it to the end of the file and concatenates a newline character at the end.

Our file, names.txt, will look like this:

```
Michael
Jill
Rebecca
Thomas
```

Writing and appending

In (w)riting mode, Python looks for the file in the folder you are in. If the file is not there, it creates it. If the file is there, all the data in the file *is erased.* In writing mode, you always start with a clean slate.

In (a)ppending mode, Python looks for the file in the folder you are in. If the file is not there, it creates it. If the file is there, all the date *stays the same*. In appending more, you continue where the file was left off.

Appending a file

Oh no! We forgot to add David. We don't want to recreate the list and add *everyone* **again**. That's a lot of work.

```
name = "David"
theFile = open("names.txt","a")
theFile.write(name)
theFile.close()
```

That wasn't too bad. David needs to get his stuff together next time.

Reading a file

There are a couple ways to read a file. Let's say we want to read a file called names.txt that contains the following information:

```
Michael
Jill
Rebecca
Thomas
David
Claira
Storm
```

Here are three different ways to read the file and put these names into a list.

```
'''Method 1'''
theFile = open("names.txt","r")
names = list()
line = theFile.readline()
while (len(line) != 0):
    names.append(line)
    line = theFile.readline()
theFile.close()


'''Method 2'''
theFile = open("names.txt","r+")
theText = theFile.read()
names = theText.split()
theFile.close()
```

```
'''Method 3'''
theFile = open("names.txt")
names = list()
for line in theFile:
  names.append(line)
theFile.close()
```

All three methods work.

The first opens the file using the mode `r` for *read*. The function `readline()` takes the first line and saves it to line. The function also moves to the next line to be read when it is called again. Then while the line is not empty, we append the line to names. After we close the file.

The second opens the file using the mode `r` for *read*. The function `read()` is used and it transfers the complete file to a string and saves it to the variable `theText`. The string member function `split()` is then used. This function takes in a string and returns a list of every word. If it is a single word, then it returns a list of letters.

The third opens the file using the mode `r` for *read*. We use a `for` loop, which saves each line in the file to the variable `line` and inside the `for` loop we are appending the variable to the list `names`.

r, r+, and nothing

When you open a file to read, you can either do `r`, `r+`, or just leave it blank. Leaving it blank and writing `r` allow the program to **only** read the file. If you use `r+`, you allow the program to **read and write** to the file.