**Reading From a File (File Input)**

**Reading a File**

We can also read a file. Here’s how:

f = open('blah.txt', **'r'**) # opens blah.txt to read it

Here we are using the ***read*** mode – note the letter ‘r’ at the end. To read a line:

f = open('blah.txt', 'r') # opens blah.txt to read it

x = f.readline() # reads a line

When you run this, nothing happens. The computer has read the line, and stored what it read into the variable x, but not displayed what it read. You have to tell the computer to *print* what it read:

f = open('blah.txt', 'r') # open blah.txt to read it

x = f.readline() # read a line

print(x) # print it

And of course, you should close the file when you are done:

f = open('blah.txt', 'r') # open blah.txt to read it

x = f.readline() # read a line

print(x) # print it

f.close()

Sometimes you don’t want to remember what is read, just print it:

f = open('blah.txt', 'r') # opens blah.txt to read it

print(f.readline()) # reads a line and print it

f.close()

What happens if you try to read what isn't there?

f = open('blah.txt', 'r')

a = f.readline()

b = f.readline()

c = f.readline()

d = f.readline()

f.close()

It works - or does it?

At the bottom of your IDE, in the console, enter each variable name:

>>>a

blah blah blah

>>> b

' '

>>>c

' '

Some variables are empty because there was nothing to read. This is a useful fact that we will use later.

**Reading Files of Unknown Length**

Most files will have thousands of lines to read. If you know what type of data you are reading, and how many lines of it you expect to read, you can put your read statement in a loop:

f = open('blah.txt', 'r') # opens blah.txt to read it

for i in range (10): # do this 10 times

print ("line: ", end="")

print (f.readline()) # reads a line and then prints it

f.close()

Notice that it printed out a lot of empty lines.

Most of the time you don’t know exactly how many lines are in a file. In Python, the readline() will return an empty string (quotes with nothing in them) when it gets to the end of a file. So we can use a while loop to continue until we hit empty:

f = open('blah.txt', 'r') # opens blah.txt to read it

print("\nStart of file")

while True: # loop forever

x = f.readline() # read a line

if x=="": # if the line is empty

print ("end of file")

break # quit the loop (stop reading)

else: # if the line isn't empty

print("line: ", x) # print it out

f.close() # close file when the loop is finished

You can **read an entire text file** into a single variable using the read() function:

f = open('blah.txt', 'r') # opens blah.txt to read it

stuff = f.read() # reads the whole text file

print (stuff) # prints it out

f.close()

This isn't usually that helpful when you have to separate the contents afterwards. But if you just need to read something in and spit it out on the screen, this is an easy way to do it.

Key Terms: **input, output, open, close, readline(), read(), strip()**

**Exercise**

1. Write a program that opens the text file you created in the previous lesson. Have it read the first name into a variable called *firstname*, the second name into a variable called *secondname*, the address into a variable named *address* and the phone number into a variable named *phone*. Put this in a loop that goes until the end of the file. Print the information out neatly as shown. Close the file.

Name: John Cleese

Address: 34 Funny Walk Way

Phone Number: 234-2342

Name: Donald Duck

Address: 99 Disney Drive

Phone Number: 333-4442

etc.

**Helpful Note:** When you get working on this, you will notice a funny problem (not haha funny). When you print things out there will be extra blank lines. This is because a line of text always ends with the newline character (\n). The computer reads them in as an actual character and then prints them out.

What the text file looks like to you:

John

Cleese

34 Funny Walk Way

What the text file looks to the computer:

John\n

Cleese\n

34 Funny Walk Way\n

Fortunately, to get rid of these annoying "Enter keystrokes", there is a handy string function called **strip**(). You use it to strip the end off the string, like this:

firstname = f.readline().strip() # take the newline character off