Aaron Eberly

CNIT 15501

Week 13 Honors Deliverable

Files

When a program has to deal with a lot of information, it may be best to use a file. To use a file, it must first be opened. Here is an example of opening a file:

my\_file = open("reminders.txt", "r")

So what does this mean? First of all, we are opening the file called reminders.txt. the “r” means that we are reading the file. Instead of “r”, we could have used “w” for writing, or “a” for append. The file will be referred to as my\_file in the program.

This will only successfully open the file if reminders.txt is in the same location as the program. If the file is not in the same location, you must instead type the entire path to the file.

It is important to close a file after you are done with it. Closing a file might look like this:

my\_file.close()

If there is a problem with opening the file, it will raise an IOError or FileNotFound exception. These can be caught like any other exception.

Writing a File

When you open a file, you can specify that you will be writing on the file, which you do by using “w” instead of “r”. If the file name does not exist, this will create a new file.

WARNING: If the file already exists, all the contents of the file will be deleted! This means that the file will be empty for writing. Be wary of erasing what is already on a file.

If you want to add to a file which already exists without overwriting what is already there, you can append instead of write. You do this by using “a” instead of “w”.

Reading a File

Remember that “r” is used to read a file. If a second parameter isn’t entered at all, read is the default.

You can use a for loop to iterate through each line in a file. Here is an example of printing every line in the file:

for line in my\_file:

    print(line)

You can also use the method readline(). readline() returns the next line in the file. Here is an example of using readline():

line = my\_file.readline()

while (line != ""):

    print(line)

    line = my\_file.readline()

In this code, the first line of a file is read into a variable. As long as the line isn’t blank, it will print the line, and then read the next line. The while loop is necessary to check if the line is blank. If so, the end of the file has been reached, and the loop should end.

You can use the method readlines() to put all of the lines of a file into a list.

allTheLines = my\_file.readlines()

print(allTheLines)

This will print out a list. Each element in the list will be a different line in the file. Printing out a list generally doesn’t look as good as the other printing methods.

The read() method returns the entire contents of a file as a string. This is different from readlines(), because read() returns a string, while readlines() returns a list of all the lines. You could use the following code to print out the contents of a file using read().

fullFile = my\_file.read()

print(fullFile)

The seek() method allows the position of the “cursor” in the file. This will allow you to read the file starting from some place in the middle of the file, rather than starting at the beginning. Here is an example:

my\_file.seek(40)

restOfFile = my\_file.read()

print(restOfFile)

The above code would print all of a file except for the first 40 characters. The first line moves the cursor 40 characters into the file. Then, when the file is read, it starts from the cursor instead of from the beginning of the file.

.rstrip()

This method will remove all trailing characters at the end of a string. This includes \t, \n, spaces, and more. The following code would be used to remove all trailing characters from the variable my\_str:

my\_str.rstrip()

Final Reminders

Always remember to open a file before using it. As soon as you are done with the file, remember to close it!