Lab 5: File I/O

CSE/IT 107

NMT Computer Science

"If you don't think carefully, you might believe that programming is just typing statements in a programming language."

— W. Cunningham

"Only ugly languages become popular. Python is the exception."

— Donald Knuth

---

1 Introduction

2 File I/O

Knowing how to work with files is important, since it lets us store and retrieve data beyond the scope of a single execution of a program. To open a file for reading or writing we will use the `open` function. The following example opens a file and writes "Hello World" to it.

```
1 output_file = open("hello.txt", "w")
2
3 print("Hello World", file=output_file)
4 output_file.close()
```

The arguments to the `open` function are, in order, the name of the file to open and the mode in which to open the file. "w" means that the file is to be opened in write mode. If the file does not exist, this will create the file. If it does exist, then the contents of the file will be cleared in preparation for the new ones.

Other options include "a", which is similar to "w" but will not clear the contents of an existing file and will instead append the new data to the end, and "r" which will read the file instead. If "r" is used and the file does not exist, then an error will occur. The following code takes a filename as user input, then prints out the entire contents of that file.

```
1 filename = input("What file should be read? ")
2
3 input_file = open(filename, "r")
4 for line in input_file:
5     print(line, end="")
6
7 input_file.close()
```

The additional concepts introduced in these examples are:

- The `print` function can have an additional "file" parameter passed to it to allow writing to a file. This causes it to send its output to the file rather than the screen, though otherwise it performs identically.
- The `print` function has an additional optional "end" parameter. This allows you to specify what should be printed after the main string given to it. This is important because it defaults to `"\n"`, which causes a newline after every print statement. By changing "end" to `""` we prevent a newline from being added after every line of the file is printed. This is because each line in the file already has a newline at the end of it, so we don't need `print` to add its own.
- `.close()` is used to properly tell Python that you are done with a file and close your connection to it. This isn't *strictly* required, but without it you risk the file being corrupted or other programs being unable to access that file.
- When reading from a file, Python can use a `for` loop to go through each line in sequence. This identically to if you think of the file as a list with every line being a different element of the list.

2.1 Error Handling

3 Program Boilerplate

4 Exercises

save.py

load.py

5 Submitting

Files to submit:

- hailstone.py

You may submit your code as either a tarball (instructions below) or as a .zip file. Either one should contain all files used in the exercises for this lab. The submitted file should be named either `cse107_firstname_lastname_lab5.zip` or `cse107_firstname_lastname_lab5.tar.gz` depending on which method you used.

For Windows, use a tool you like to create a `.zip` file. The TCC computers should have `7z` installed. For Linux, look at lab 1 for instructions on how to create a tarball or use the "Archive Manager" graphical tool.

Upload your tarball or .zip file to Canvas.