Input from and Output to Files

CMSC 104 Section 02
October 2, 2025

Administrative Notes

Classwork 4 today

Using files

Until now all input has come from the keyboard and all output has gone to the screen

That’s not useful if you want to preserve & re-use data

- Don’t make the user type the data in every time
- Don’t make the user rely on screenshots to record the results of the program
- Make your program part of a “pipeline” that accomplishes work

File input

The biggest issues:

Knowing the name of the file, including the directory path

Knowing the format of the data in the file

Steps

First, open the file:

- If you know the name and path ahead of time
- If you ask the user to enter the name

Next, read the contents

- readline() - read one line of the file as a string
- readlines() - read the rest of the file, one line at a time, as a list of strings
- read() - read the rest of the file, as a single string including all lines

Then, close the file

- The way we do it, this will be handled automatically

Opening the file

This is unique to Windows

If you know the name when you write the program

```
fname = "C:\\users\\aarsenault\\Documents\\Spring24.csv"
with open(fname, "r") as infile:
      data = infile.readline()
# now you have one line read in, and you can process it
      new_data = infile.readlines()
# now you have the rest of the file as a list of strings
```

Backslashes and escaping characters

If you want to encode “non-printable” characters in Python, you use a backslash

\ is a backslash; / is a slash

\n is a newline character

\t is a tab character

And so on.

Mac, Linux, etc. use forward slashes in their directory paths - so no problem

```
/alfredarsenault/PythonProjects/Data/Spring24.csv
```

But Windows uses backslashes - which can cause Python to misinterpret your path name as an escape character

- Spoiler alert: your program crashes, and you have to figure out why

Working through an example

See the file “Fall_2025.csv” on GitHub