CMSC201 Computer Science I for Majors

Lecture 18 – File Input/Output

Last Class We Covered

- Recursion
  - Recursion
    - Recursion
- Fibonacci Sequences
- Recursion vs Iteration

Any Questions from Last Time?

Today’s Objectives

- To learn all about file input and output

- Including how to:
  - Open a file
  - Read in its data
  - Write to a file
  - Close a file

File Input and Output

Why Use Files?

- Until now, the Python programs you've been writing use pretty simple input and output
  - User types input at the keyboard
  - Results (output) are displayed in the console
- This is fine for short and simple input…
  - But what if we want to average 50 numbers, and mess up when entering the 37th one?
  - Start all over???

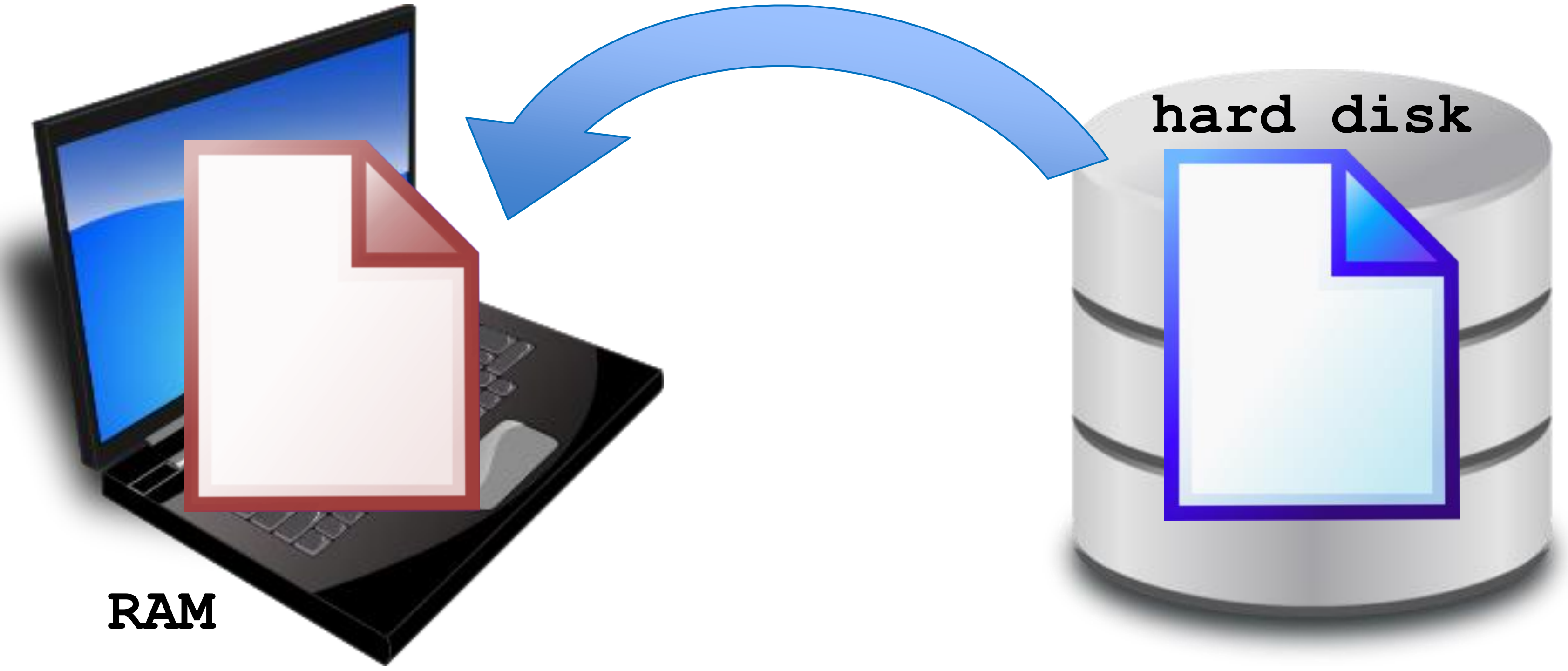
What is File I/O?

- One solution is to read the information in from a file on your computer
  - You can even write information to a file

- This process is called ***File I/O***
  - "I/O" stands for "input/output“
  - Python has built-in functions that make this easy

File I/O Example: Word Processor

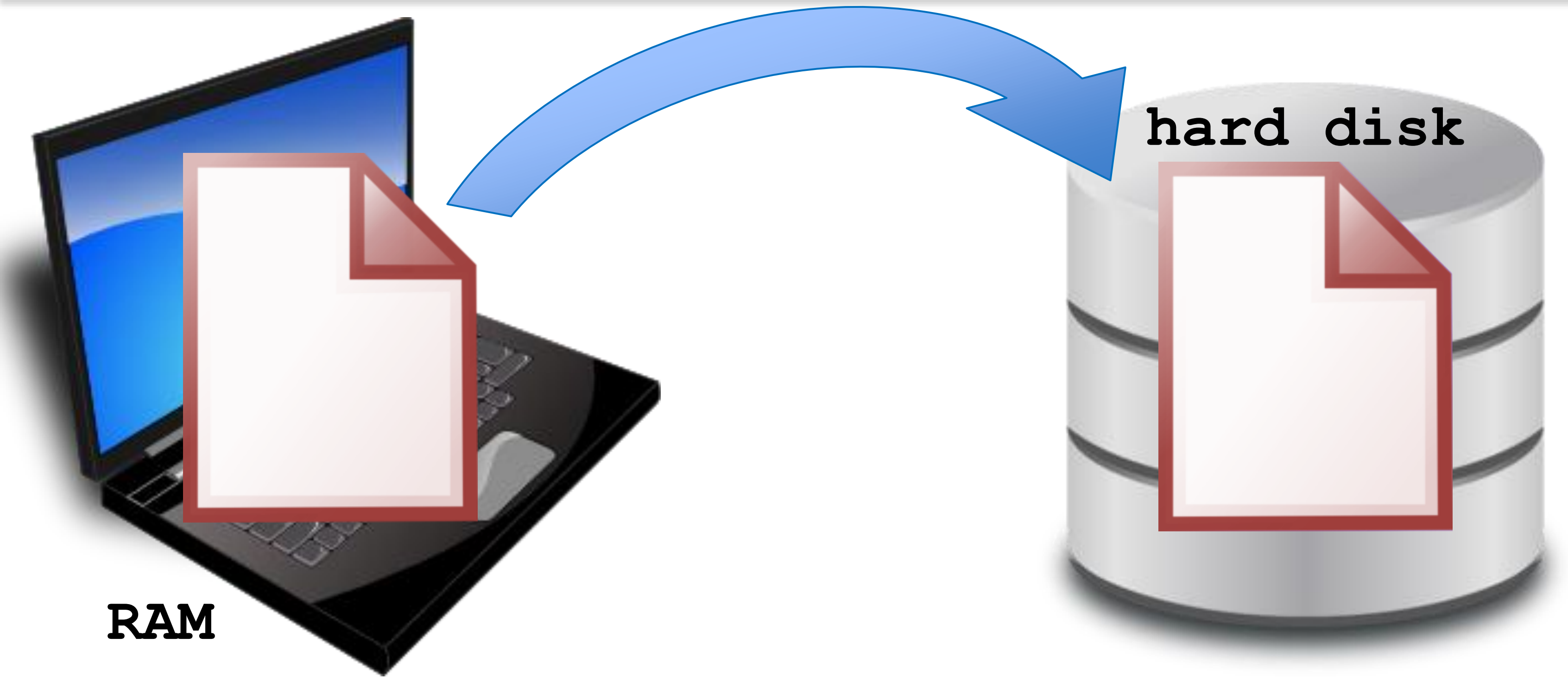
- "Reading" in a file using a word processor
  - File opened from hard disk
  - Contents read into memory (RAM)
  - File closed on hard disk
  - IMPORTANT: Changes to the file are made to the copy stored in memory, <u>not</u> the original file on the disk



1. File opened from hard disk
2. Contents read into memory (RAM)
3. File closed from hard disk
4. Changes are saved to the copy in memory

File I/O Example: Word Processor

- “Writing” a file using a word processor
  - (Saving a word processing file)
  - Original file on the disk is reopened in a mode that will allow writing
    - This actually erases the old contents!
  - Copy the version of the document stored in memory to the original file on disk
  - File is closed



1. File opened on hard disk for writing
2. (Old contents are erased!)
3. Copy version in memory to hard disk
4. Close file on hard disk

File Processing

- In order to do interesting things with files, we need to be able to perform certain operations:
  - Associate an external file with a variable
    - Opening the file
  - Manipulate the variable (the file object)
    - Reading from or writing to the file object
  - Close the file
    - Making sure the object and file match at the end

Opening a File

Syntax for `open()` Function

```
myFile = open(file_name [, access_mode])
```

`file_name`

- This argument is a string the contains the name of the file you want to access
  - `"input.txt"`
  - `fileName`
  - `"roster.dat"`

Syntax for `open()` Function

```
myFile = open(file_name [, access_mode])
```

`access_mode` (<u>optional</u> argument)

- This argument is a string that determines which of the modes the file is to be opened in
  - **`"r"`** (open for reading)
  - **`"w"`** (open for writing)
  - **`"a"`** (open for appending)

File being opened must be in the same folder as the Python file

Examples of Using **open()**

- In general, we will use commands like:

```
myFile  = open("scores.txt")
dataIn  = open(statsFile,  "r")
dataOut = open("stats2.dat", "w")
```

an example input file

```
scores.txt
2.5   8.1 7.6 3.2  3.2
3.0  11.6 6.5 2.7 12.4
8.0   8.0 8.0 8.0  7.5
```

Reading in a File

Using File Objects to Read Files

```
myFile = open("myStuff.txt")
```

- This line of code does three things:
  1. Opens the file “myStuff.txt”
  2. In “reading” mode (which is the default)
  3. Assigns the opened file to the variable **`myFile`**
- Once the file is open and assigned to a variable, we can start reading it

Three Ways to Read a File

- There are three different ways to read in a file:

1. Read the whole file in as one big long string
   `myFile.read()`
2. Read the file in one line at a time
   `myFile.readline()`
3. Read the file in as a list of strings (each is one line)
   `myFile.readlines()`

Entire Contents into One String

```
>>> info = open("hours.txt")
>>> wholeThing = info.read()
>>> wholeThing
'123 Suzy 9.5 8.1 7.6 3.1 3.2\n456 Brad 7.0
9.6 6.5 4.9 8.8\n789 Jenn 8.0 8.0 8.0 8.0
7.5'
```

it's literally one giant string!

our input file

```
hours.txt
123 Suzy 9.5 8.1 7.6 3.1 3.2
456 Brad 7.0 9.6 6.5 4.9 8.8
789 Jenn 8.0 8.0 8.0 8.0 7.5
```

Entire Contents into One String

```
>>> info = open("hours.txt")
>>> wholeThing = info.read()
>>> wholeThing
'123 Suzy 9.5 8.1 7.6 3.1 3.2\n456 Brad 7.0
9.6 6.5 4.9 8.8\n789 Jenn 8.0 8.0 8.0 8.0
7.5\n
```

it's literally one giant string!

notice the escape sequence (**\n**) is read in as well

our input file

```
hours.txt
123 Suzy 9.5 8.1 7.6 3.1 3.2
456 Brad 7.0 9.6 6.5 4.9 8.8
789 Jenn 8.0 8.0 8.0 8.0 7.5
```

One Line at a Time

```
>>> info = open("hours.txt")
>>> lineOne = info.readline()
>>> lineOne
'123 Suzy 9.5 8.1 7.6 3.1 3.2\n'
>>> lineTwo = info.readline()
'456 Brad 7.0 9.6 6.5 4.9 8.8\n'
```

our input file

```
hours.txt
123 Suzy 9.5 8.1 7.6 3.1 3.2
456 Brad 7.0 9.6 6.5 4.9 8.8
789 Jenn 8.0 8.0 8.0 8.0 7.5
```

As a List of Strings

```
>>> info = open("hours.txt")
>>> listOfLines = info.readlines()
>>> listOfLines
['123 Suzy 9.5 8.1 7.6 3.1 3.2\n', '456 Brad
7.0 9.6 6.5 4.9 8.8\n', '789 Jenn 8.0 8.0
8.0 8.0 7.5\n']
```

our input file

```
hours.txt
123 Suzy 9.5 8.1 7.6 3.1 3.2
456 Brad 7.0 9.6 6.5 4.9 8.8
789 Jenn 8.0 8.0 8.0 8.0 7.5
```

Using `open()`

- Which of these are valid uses of `open()`?

```
1.myFile  = open(12, "r")
2.fileObj = open("HELLO.txt")
3.writeTo = open(fileName, "w")
4."file"  = open("test.dat", "R")
5.theFile = open("file.dat", "a")
```

Using `open()`

- Which of these are valid uses of `open()`?

not a valid string

✗ 1. `myFile   = open(12, "r")`

✓ 2. `fileObj = open("HELLO.txt")`

uppercase "**R**" is not a valid access mode

✓ 3. `w` not a valid variable name `(fi`

✗ 4. `"file"   = open("test.dat", "R")`

✓ 5. `theFile = open("file.dat", "a")`

Three Ways to Read a File

- Write the code that will perform each of these actions using a file object called **`fileIn`**

1. Read the whole file in as one big long string

2. Read the first line of the file

3. Read the file in as a list of strings (each is one line)

Three Ways to Read a File

- Write the code that will perform each of these actions using a file object called **`fileIn`**

1. Read the whole file in as one big long string

```
bigString  = fileIn.read()
```

2. Read the first line of the file

```
firstLine  = fileIn.readline()
```

3. Read the file in as a list of strings (each is one line)

```
stringList = fileIn.readlines()
```

Writing to Files

Opening a File for Writing

- Use **`open()`** just like we do for reading
  - Provide the filename <u>and the access mode</u>

```
fileObj = open("output.txt", "w")
```

  - Opens the file for writing
  - Wipes the contents!

```
fileObj = open("myNotes.txt", "a")
```

  - Opens the file for appending
  - Writes new data to the end of the file

Writing to a File

- Once a file has been opened, we can write to it
  - What do you think the method to write is called?

```
myFile.write( "hello world!" )
```

- We can also use a string variable in **`write()`**

```
myFile.write( writeString )
```

Details About `write()`

- `write()` only writes exactly what it's given!
  - This means whitespace (like `"\n"`) is up to you
  - Unlike `print()`, which adds a newline for you

```
myFile = open("greeting.dat", "w")
myFile.write("Hello\nWorld\n")
myFile.close()
```

Word of Caution

- Write can only take one string at a time!

Why don't these work?
the first is multiple strings
the second is an int, not a string

- These won't work:

```
fileObj.write("hello", "my", "name")
fileObj.write(17)
```

Why does this work?
concatenation creates one string
casting turns the int into a string

- But this will:

```
fileObj.write("hello" + " my " + "name")
fileObj.write(str(17))
```

Closing a File

- Once we are done with our file, we close it
  - We do this for all files – ones that we opened for writing, reading, or appending!

```
myFileObject.close()
```

- Properly closing the file is important – why?
  - It ensures that the file is saved correctly

Time for…

LIVECODING!!!

deSpacing

- Write a function that
  - Reads in from a file called “spaced.txt”
  - Counts how many whitespace characters it has
    (`\n`, `\t`, and)
  - Prints out the total count of whitespace characters
  - Creates a new file without any of the whitespace characters (called “unspaced.txt”)

deSpacing: Output

- File: Available in Dr. Gibson’s pub directory

  `/afs/umbc.edu/users/k/k38/pub/cs201/spaced.txt`

  – Lots of tabs and spaces

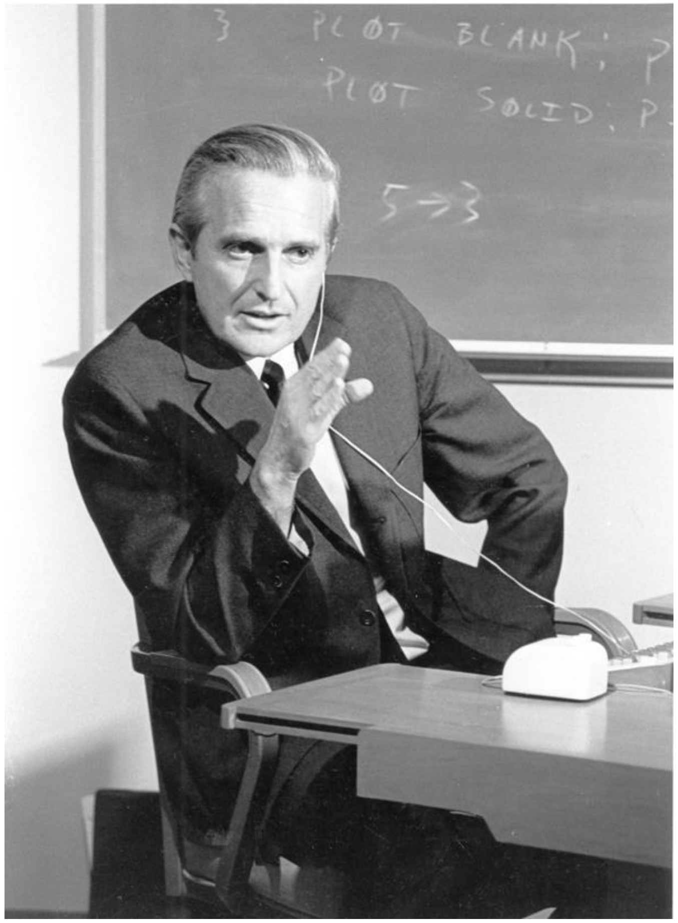
```
          How  now  brown        cow


Space


I like                           space
```

- Output:

```
linux1[37]% python3 spaced.py
There were 44 spacing characters in the file
```

Daily CS History

- Douglas Englebart
  - Invented the computer mouse, bitmapped screens, hypertext, and the precursor to the GUI
    - “Mother of All Demos”
  - Focused his career on “making the world a better place”
  - Believed the way to do this was by using technology to augment human intelligence



Announcements

- Homework 6 is due this Friday at 11:59:59 PM
- Midterm 2 grades will be returned <u>next</u> week
- Final exam is going to be on:
  - Friday, May 17th from 6 to 8 PM
  - Rooms will be assigned closer to the date
  - If you can't take the exam at that time, you need to let Dr. Gibson know via email NOW, not later

Image Sources

- Laptop:
  - https://pixabay.com/p-33521
- Database:
  - http://www.clipartkid.com/database-symbol-clip-art-at-clker-com-vector-clip-art-online-QMSKDE-clipart/
- Douglas Englebart:
  - https://commons.wikimedia.org/wiki/File:SRI_Douglas_Engelbart_1968.jpg