

# CMSC201

## Computer Science I for Majors

### Lecture 17 – File I/O (Continued)

# Last Class We Covered

- Escape sequences
  - Uses a backslash (\)
- File I/O
  - How to open a file
    - For reading or writing
  - How to read lines from a file
- The `split()` function
  - To break a string into tokens

# Any Questions from Last Time?

# Today's Objectives

- To review how to open and read from a file
- To learn the `join()` function
  - “Opposite” of the `split()` function
- To get more practice with File I/O
- To cover the different ways to write to a file
- To learn how to close a file

# Review from Last Class

# 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 ()` ?

**x** 1. `myFile = open (12, "r")`

not a valid string

**✓** 2. `fileObj = open ("HELLO.txt")`

**✓** 3. `writeFile ("file.dat", "R")`

not a valid variable name

uppercase "R" is not a valid access mode

**x** 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()**

# Whitespace

- There are two ways we know of to remove whitespace from a string
- The `strip()` function removes all leading and trailing whitespace (tabs, spaces, newlines) from a string  
`withoutWhitespace = myLine.strip()`
- The `split()` function can be used to remove all whitespace (including interior), creating a list of strings  
`tokens = myLine.split()`

# Splitting and Joining

# Review: String Splitting

- The `split()` function can be used in two ways:
  - Break the string up by its whitespace
  - Break the string up by a specific character
- Both methods take in a single string, and return a list of one or more strings

```
>>> names = "Avery Ben Chetra Emily"
```

```
>>> names.split()
```

```
['Avery', 'Ben', 'Chetra', 'Emily']
```

# Joining Strings

- We can also join a list of strings back together!
  - The syntax is very different from `split()`
  - And it only works on a list of strings

```
"X".join(list_of_strings)
```

function  
name

the list of strings we want to join together

the delimiter (what we will use to join the strings)

# Example: Joining Strings

```
>>> names = ['Alice', 'Bob', 'Carl', 'Dana', 'Eve']
>>> "_".join(names)
'Alice_Bob_Carl_Dana_Eve'
```

- We can also use more than one character as our delimiter if we want

```
>>> " <3 ".join(names)
'Alice <3 Bob <3 Carl <3 Dana <3 Eve'
```

# Practice: Joining

- Use `join()` to solve the following problems
- From `grades`, create a string that looks like  
`"A+, A, A-, B+, B, B-, C+, C, C-, D, F"`
- From `acronyms`, create a string that looks like  
`"OMG BRB LOL BBQ IDK BFF JK OMW FYI"`

## Splitting into Variables



# Known (Formatted) Input

- ***Known input*** means that we know how the data inside a file will be formatted (laid out)
- For example, in `workerHours.txt`, we have:
  - The employee ID number
  - The employee's name
  - The hours worked over five days

**workerHours.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
```

# Splitting into Variables

- If we know what the input will look like, we can `split()` them directly into different variables

```
var1, var2, var3 = threePartString.split()
```


all of the variables we want  
to split the string into

the string whose input we  
know, and are splitting on

we can have as many different  
variables as we want

# Example: Splitting into Variables

```
>>> s = "Jessica 31 647.28"  
>>> name, age, money = s.split()  
>>> name  
'Jessica'  
>>> int(age)  
31  
>>> float(money)  
647.28
```



we may want to convert some of them to something that's not a string

# Writing to Files

# Opening a File for Writing

- Use `open ()` just like we do for reading
  - Provide the filename and the access mode

```
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 function 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...

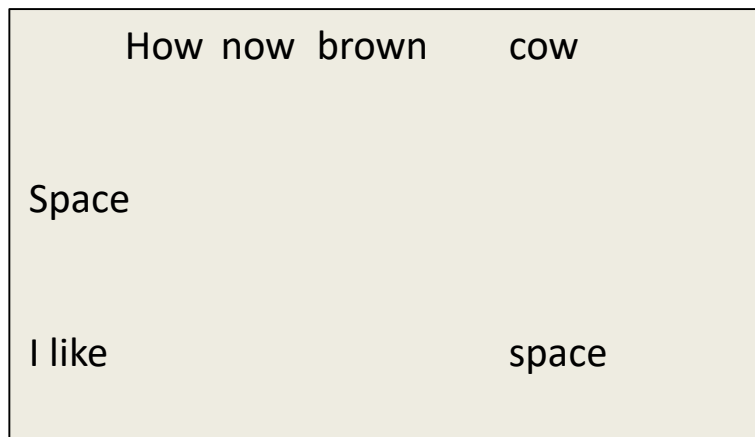
**LIVECODING!!!**

# deSpacing

- Write a function that
  - Reads in from a file called “spaced.txt”
  - Counts how many whitespace (`\n`, `\t`, and `' '`) characters it has
  - 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/k/k38/pub/cs201/spaced.txt`  
– Lots of tabs and spaces



```
How now brown cow
Space
I like space
```

- Output:

```
bash-4.1$ python spaced.py
```

```
There were 44 spacing characters in the file
```

# Announcements

- Project 2 out on Blackboard
  - Design due Friday, April 14th @ 8:59:59 PM
  - Project due Friday, April 21st @ 8:59:59 PM
  - Uses 3D lists and file I/O
- Final exam is when?
  - Friday, May 19th from 6 to 8 PM