UMBC

AN HONORS UNIVERSITY IN MARYLAND

CMSC201 Computer Science I for Majors

Lecture 14 – File I/O (Continued)

Prof. Katherine Gibson

Prof. Jeremy Dixon

Last Class We Covered

- Escape sequences
  - Uses a backslash (\)
- File I/O
  - Input/Output
  - How to open a file
    - For reading or writing
  - How to read lines from a file

Any Questions from Last Time?

Survey #1

- Available now on Blackboard
- Due by Friday, April 1st at midnight
  - Check completion under "My Grades"
- Some statistics (from 500 students):
  - If they had taken the surveys…
    - 9 students would have gotten an A instead of a B
    - 4 students would have gotten a B instead of a C
    - 9 students would have gotten a C instead of a D

Today’s Objectives

- To review how to open and read from a file
- To learn how to use the **`split()`** function
  - To break a string into tokens
  - And to learn the **`join()`** 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")
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")
✓3. wr                      n(fi
✗4. "file"   = open("test.dat", "R")
✓5. theFile = open("file.dat", "a")
```

not a valid filename

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

Three Ways to Read a File

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

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 **`aFile`**

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

```
bigString  = aFile.read()
```

2. Read the first line of the file

```
firstLine  = aFile.readline()
```

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

```
stringList = aFile.readlines()
```

Whitespace

- There are two ways we know of to remove whitespace from a string

- Slicing can be used to remove just the newline at the end of a line that we have read in from a file:

```
myLineWithoutNewline = myLine[:-1]
```

- The **`strip()`** function removes all leading and trailing whitespace (tabs, spaces, newlines) from a string

```
withoutWhitespace = myLine.strip()
```

Using `for` Loops to Read in Files

- Remember, `for` loops are great for iterating!
- With a list, the `for` loop iterates over…
  - Each element of the list (in order)
- Using a `range()`, the `for` loop iterates over…
  - Each number generated by the range (in order)
- And with a file, the `for` loop iterates over…
  - Each line of the file (in order)

String Splitting

String Splitting

- We can break a string into individual pieces
  - That you can then loop over!

- The function is called **`split()`**, and it has two ways it can be used:
  - Break the string up by its whitespace
  - Break the string up by a specific character

Splitting by Whitespace

- Calling **`split()`** with no arguments will remove all of the whitespace in a string
  - Even the “interior” whitespace

```
>>> line = "hello world this is my song\n"
>>> line.split()
['hello', 'world', 'this', 'is', 'my', 'song']

>>> whiteCat = "\t\nI   love\t\t\nwhitespace\n  "
>>> whiteCat.split()
['I', 'love', 'whitespace']
```

Splitting by Specific Character

- Calling **`split()`** with a string in it, we can remove a specific character (or more than one)

these character(s) are called the delimiter

```
>>> commas = "once,twice,thrice"
>>> commas.split(",")
['once', 'twice', 'thrice']

>>> double = "hello how ill are all of your llamas?"
>>> double.split("ll")
['he', 'o how i', ' are a', ' of your ', 'amas?']
```

Splitting by Specific Character

- Calling **`split()`** with a string in it, we can remove a specific character (or more than one)

these character(s) are called the delimiter

```
>>> commas = "once,twice,thrice"
>>> commas.split(",")
['once', 'twice', 'thrice']

>>> double = "hello how ill are all of your llamas?"
>>> double.split("ll")
['he', 'o how i', ' are a', ' of your ', 'amas?']
```

notice that it didn't remove the whitespace

Practice: Splitting

- Use **`split()`** to solve the following problems

- Split this string on all of its whitespace:

```
daft = "around the \nworld"
```

- Split this string on the double t’s (**`tt`**):

```
doubleT = "nutty otters making lattes"
```

Practice: Splitting

- Use **`split()`** to solve the following problems

- Split this string on all of its whitespace:

```
daft = "around the \nworld"
daft.split()
```

- Split this string on the double t's (**`tt`**):

```
doubleT = "nutty otters making lattes"
doubleT.split("tt")
```

Looping over Split Strings

- Splitting a string creates a list of smaller strings
- Using a **`for`** loop with a split string, we can iterate over each word (or token) in the string
- Syntax:

```
for piece in myString.split():
    # do something with each piece
```

Example: Looping over Split Strings

```
double = "hello how ill are all of your llamas?“
for token in double.split("ll"):
    print("y" + token + "y")
```

```
yhey
yo how iy
y are ay
y of your y
yamas?y
```

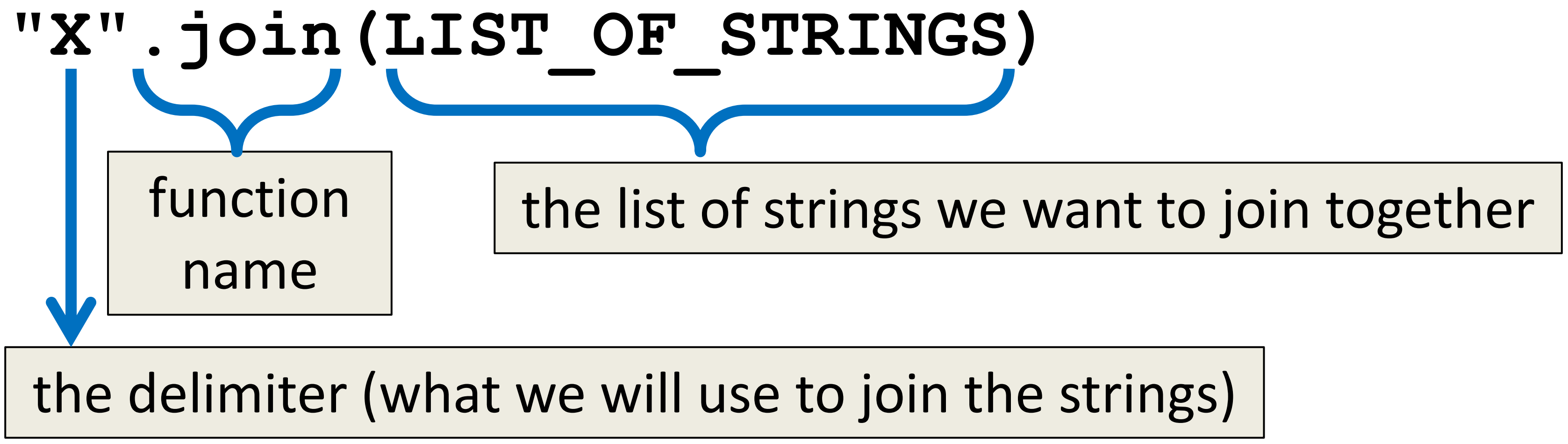
append a “y” to the front and end of each list element, then print

remember, **`double.split("ll")`** makes the list `['he', 'o how i', ' are a', ' of your ', 'amas?']`

String Joining

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 <u>strings</u>



```
"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'
```

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 <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

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

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

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

Word of Caution

- Write can only take <u>one string</u> 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 <u>one</u> string

- But this will:

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

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, and appending!

```
myFileObject.close()
```

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

Announcements

- Homework 7 will be out soon
  - Due Monday, April 4
- Complete the survey on Blackboard!
  - Do not lose an easy 1% of your total grade!

Exercise: Writing to a File

- Remember our grocery list program?
- At the end of our program, the user has added all of their items to the list **`grocery_list`**

- Write the contents of **`grocery_list`** to a file
  - Don’t forget to open and close the file!

Solution: Writing to a File

```
# code above this populates grocery_list

# open file for writing
gFile = open("groceries.txt", "w")

for g in grocery_list:
    # print each item, plus a newline
    gFile.write(g + "\n")

# close file
gFile.close()
```

Writing to a File: Newlines

- Why did we need a newline in our example?
- Without it, our file looks like this:

```
durianscoconutlimecoke
```

- But with it, each item is on a separate line:

```
durians
coconut
lime
coke
```