

CMSC201

Computer Science I for Majors

Lecture 13 – File I/O

Prof. Katherine Gibson

Prof. Jeremy Dixon

Last Class We Covered

- Functions
 - Returning values
 - Returning multiple values at once
- Modifying parameters
 - Mutable
 - Immutable
- Modular programming

Any Questions from Last Time?

Today's Objectives

- To learn about escape sequences
 - What they are
 - Why we need them
 - How to use them
- To be able to
 - Open a file
 - Read in its data

Escape Sequences

“Misbehaving” `print()` Function

- There are times when the `print()` function doesn't output exactly what we want

```
>>> print("I am 5 feet, 4 inches")
```

```
I am 5 feet, 4 inches
```

```
>>> print("I am 5'4'")
```

```
File "<stdin>", line 1
```

```
    print("I am 5'4'")
```

```
        ^
```

```
SyntaxError: EOL while scanning string literal
```

Special Characters

- Just like Python has special keywords...
 - `for`, `int`, `True`, etc.
- It also has special characters
 - Single quote (`'`), double quote (`"`), etc.

Backslash: Escape Sequences

- The backslash character (\) is used to “*escape*” a special character in Python
 - Tells Python not to treat it as special
- The backslash character goes in front of the character we want to “escape”

```
>>> print("I am 5'4\"")
```

```
I am 5'4"
```


Using Escape Sequences

- There are three ways to solve the problem of printing out our height using quotes

```
>>> print("I am 5'4\"")
```

```
I am 5'4"
```

```
>>> print('I am 5\'4"')
```

```
I am 5'4"
```

```
>>> print("I am 5\'4\"")
```

```
I am 5'4"
```

Using Escape Sequences

- There are three ways to solve the problem of printing out our height using quotes

```
>>> print("I am 5'4\"")
```

```
I am 5'4"
```

escape double quotes
(using " for the string)

```
>>> print('I am 5\'4"')
```

```
I am 5'4"
```

escape single quotes
(using ' for the string)

```
>>> print("I am 5\'4\"")
```

```
I am 5'4"
```

escape both single and
double quotes (works
for both ' and ")

Common Escape Sequences

Escape Sequence	Purpose
<code>\'</code>	Print a single quote
<code>\"</code>	Print a double quote
<code>\\</code>	Print a backslash
<code>\t</code>	Print a tab
<code>\n</code>	Print a new line (“enter”)
<code>"""</code>	Allows multiple lines of text

""" is not really an escape sequence, but is useful for printing quotes

Escape Sequences Example

```
tabby_cat = "\tI'm tabbed in."
```

```
print(tabby_cat)
```

```
I'm tabbed in.
```

\t adds a tab

```
persian_cat = "I'm split\non a line."
```

```
print(persian_cat)
```

```
I'm split
```

```
on a line.
```

\n adds a newline

```
backslash_cat = "I'm \\ a \\ cat."
```

```
print(backslash_cat)
```

```
I'm \ a \ cat.
```

\\ adds a single backslash

Escape Sequences Example

```
fat_cat = """  
I like to eat:  
\t* Cat food  
\t* Fishies  
\t* Catnip\n\t* Grass  
"""  
print(fat_cat)
```

Escape Sequences Example

```
fat_cat = """
```

```
I like to eat:
```

```
\t* Cat food
```

```
\t* Fishies
```

```
\t* Catnip\n\t* Grass
```

```
"""
```

```
print(fat_cat)
```

```
I like to eat:
```

```
    * Cat food
```

```
    * Fishies
```

```
    * Catnip
```

```
    * Grass
```

`\t` puts in a tab

`\n` adds a newline

Escape Sequences Example

```
fat_cat = """  
I like to eat:  
\t* Cat food  
\t* Fishies  
\t* Catnip\n\t* Grass  
"""
```

```
print(fat_cat)
```

```
I like to eat:  
    * Cat food  
    * Fishies  
    * Catnip  
    * Grass
```

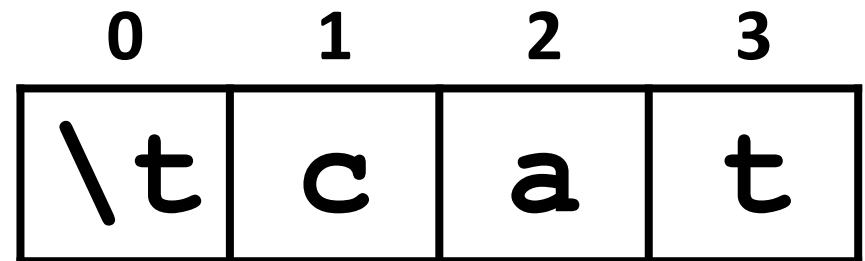
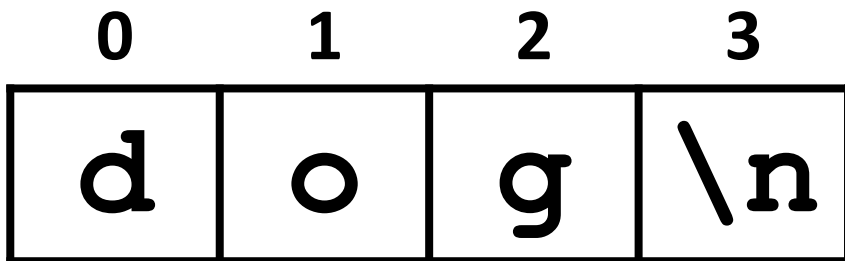
when using triple quotes ("\""), the times you hit "enter" inside the string will print as newlines

How Python Handles Escape Sequences

- Escape sequences look like two characters to us
- Python treats them as a single character

```
example1 = "dog\n"
```

```
example2 = "\tcat"
```



File Input and Output

Why Use Files?

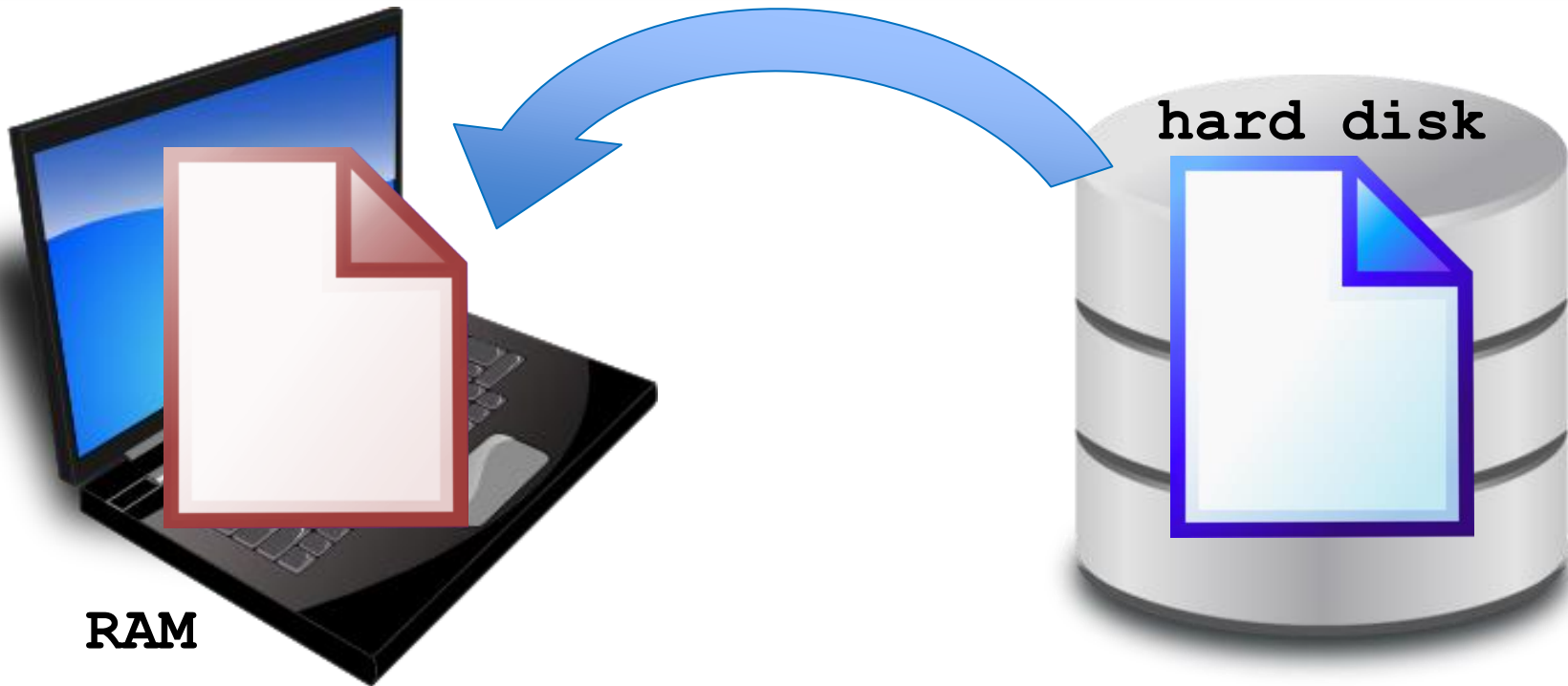
- Until now, the Python programs you've been writing are pretty simple for input/output
 - User types input at the keyboard
 - Results (output) are displayed in the console
- This is fine for short and simple input...
 - But what if we want to average 50 numbers, and mess up when entering the 37th one?
 - 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, not 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 program object
 - Opening the file
 - Manipulate the file object
 - Reading from or writing to the file object
 - Close the file
 - Making sure the object and file match at the end

Syntax: Opening a File

Syntax for `open()` Function

```
myFile = open(FILE_NAME [, ACCESS_MODE])
```

FILE_NAME

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

Syntax for `open()` Function

```
myFile = open(FILE_NAME [, ACCESS_MODE])
```

ACCESS_MODE (optional 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)

Examples of Using `open ()`

- In general, we will use commands like:

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

```
dataIn = open("stats.dat", "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

File Processing: Reading

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
```

it's literally one
giant string!

```
'123 Susan 12.5 8.1 7.6 3.2\n456 Brad 4.0
11.6 6.5 2.7 12\n789 Jenn 8.0 8.0 8.0 8.0
7.5\n'
```

our input file

hours.txt

```
123 Susan 12.5 8.1 7.6 3.2
456 Brad 4.0 11.6 6.5 2.7 12
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
```

it's literally one
giant string!

```
'123 Susan 12.5 8.1 7.6 3.2\n456 Brad 4.0
11.6 6.5 2.7 12\n789 Jenn 8.0 8.0 8.0 8.0
7.5\n'
```

notice the escape sequence
(`\n`) is being printed, instead of
the text starting on a new line

our input file

hours.txt

```
123 Susan 12.5 8.1 7.6 3.2
456 Brad 4.0 11.6 6.5 2.7 12
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 Susan 12.5 8.1 7.6 3.2\n'
>>> lineTwo = info.readline()
'456 Brad 4.0 11.6 6.5 2.7 12\n'
```

there's actually an easier way to do this... can you guess what it is?

(we'll show you soon)

our input file

hours.txt

```
123 Susan 12.5 8.1 7.6 3.2
456 Brad 4.0 11.6 6.5 2.7 12
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 Susan 12.5 8.1 7.6 3.2\n',
 '456 Brad 4.0 11.6 6.5 2.7 12\n',
 '789 Jenn 8.0 8.0 8.0 8.0 7.5\n']
```

our input file

hours.txt

```
123 Susan 12.5 8.1 7.6 3.2
456 Brad 4.0 11.6 6.5 2.7 12
789 Jenn 8.0 8.0 8.0 8.0 7.5
```

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)

A Better Way to Read One Line at a Time

- Instead of reading them manually, use a **for** loop to iterate through the file line by line

```
>>> info = open("hours.txt")
>>> for eachLine in info:
...     print(eachLine)
...
123 Susan 12.5 8.1 7.6 3.2

456 Brad 4.0 11.6 6.5 2.7 12

789 Jenn 8.0 8.0 8.0 8.0 7.5
```

A Better Way to Read One Line at a Time

- Instead of reading them manually, use a **for** loop to iterate through the file line by line

```
>>> info = open("hours.txt")
>>> for eachLine in info:
...     print(eachLine)
...
123 Susan 12.5 8.1 7.6 3.2
456 Brad 4.0 11.6 6.5 2.7 12
789 Jenn 8.0 8.0 8.0 8.0 7.5
```

why are there all these empty lines???

now that we're calling **print()**, the **\n** is printing out as a second new line

Whitespace

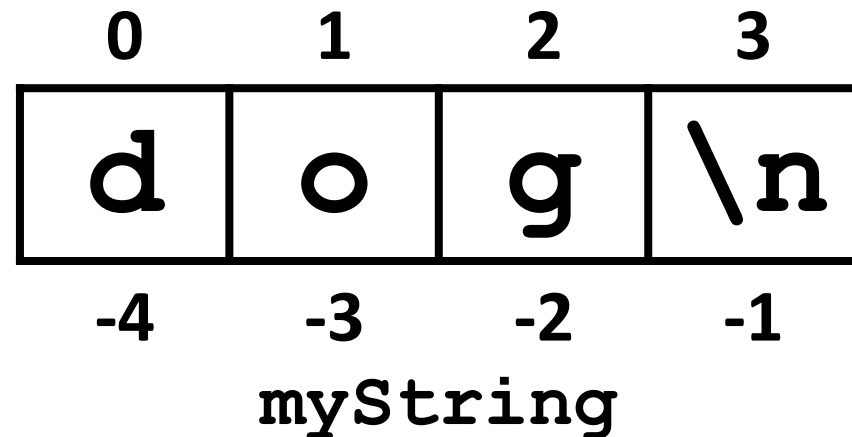
Whitespace

- Whitespace is any “blank” character, that represents space between other characters
- For example: tabs, newlines, and spaces
 "`\t`" "`\n`" " "
- When we read in a file, we can get whitespace
 - Sometimes, we don't want to keep it

Removing the Newline from the End

- To remove the escaped newline sequence (`\n`) from a string we read in, we can use slicing

```
myString = myString[:-1]
```



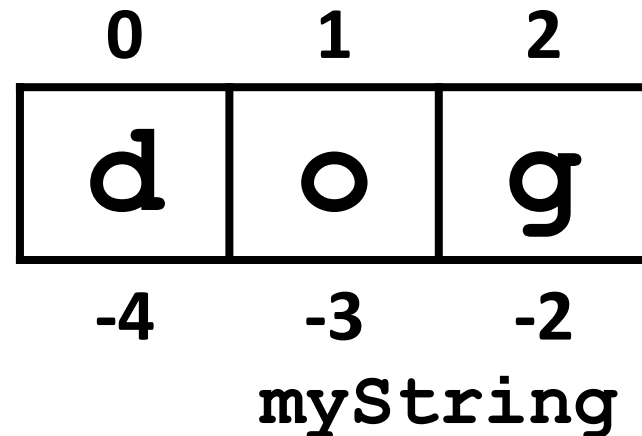
Removing the Newline from the End

- To remove the escaped newline sequence (`\n`) from a string we read in, we can use slicing

```
myString = myString[:-1]
```

don't remove anything from the beginning

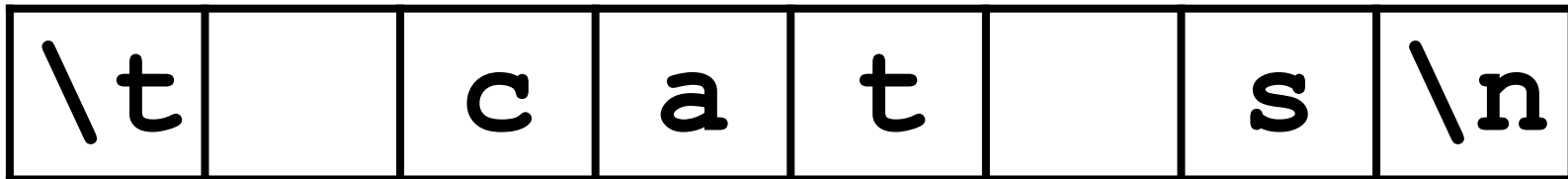
just remove the very last character



Removing Whitespace

- To remove all whitespace from the start and end of a string, we can use `strip()`

```
spacedOut = spacedOut.strip()
```

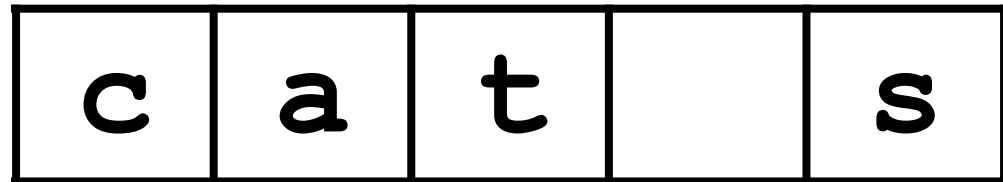


spacedOut

Removing Whitespace

- To remove all whitespace from the start and end of a string, we can use `strip()`

```
spacedOut = spacedOut.strip()
```



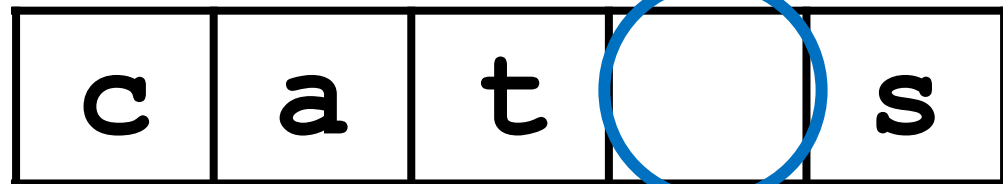
spacedOut

Removing Whitespace

- To remove all whitespace from the start and end of a string, we can use `strip()`

```
spacedOut = spacedOut.strip()
```

notice that `strip()` does not remove “interior” spacing



spacedOut

Miscellaneous (and Exercises!)

Getting a Filename from a User

- Instead of putting the filename straight in the code, we can ask the user for the filename
- Save their response in a variable, and call the `open ()` function with it

```
# printfile.py
#     Prints a file to the screen.

def main():
    fname = input("Enter filename: ")
    infile = open(fname, 'r')
    data = infile.read()
    print(data)

main()
```

Exercise: Jabberwocky

- Write a program that goes through a file and reports the longest line in the file

Example Input File:

caroll.txt

```
Beware the Jabberwock, my son,  
the jaws that bite, the claws that catch,  
Beware the JubJub bird and shun  
the frumious bandersnatch.
```

Example Output:

```
>>> longest.py  
longest line = 42 characters  
the jaws that bite, the claws that catch,
```


Jabberwocky Solution Pseudocode

inside main:

open the file "carroll.txt" (for reading)

create a variable to store the "longest" line

we'll refer to this variable as "record"

what should this variable be initialized to?

for each line of the input

if the current line is longer than the record

update the record to the current line

print the length of the longest line

print the longest line


call main

Jabberwocky Solution Code

```
def main():
    inputFile = open("carroll.txt")
    longest = ""
    for line in inputFile:
        if len(line) > len(longest):
            longest = line

    print("Longest line =", len(longest))
    print(longest)
main()
```

Jabberwocky Solution Walkthrough

```
line =  Beware the Jabberwock, my son,  
the jaws that bite, the claws that catch,  
Beware the JubJub bird and shun  
the frumious bandersnatch.
```

```
line = "Beware the Jabberwock, my son,"
```


```
longest = ""
```

```
longest = "Beware the Jabberwock, my son,"
```

```
len(line) > len(longest)
```

```
31 > 0
```



```
 for line in inputFile:  
    if len(line) > len(longest):  
        longest = line
```

Jabberwocky Solution Walkthrough

`line =` →

```
Beware the Jabberwock, my son,
the jaws that bite, the claws that catch,
Beware the JubJub bird and shun
the frumious bandersnatch.
```

```
line = "the jaws that bite, the claws that catch,"
```

```
longest = ""
```

```
longest = "Beware the Jabberwock, my son,"
```

```
longest = "the jaws that bite, the claws that catch,"
```


```
len(line) > len(longest)
```

```
42 > 31
```



```
→ for line in inputFile:
    if len(line) > len(longest):
        longest = line
```

Jabberwocky Solution Walkthrough

`line =` 

```
Beware the Jabberwock, my son,
the jaws that bite, the claws that catch,
Beware the JubJub bird and shun
the frumious bandersnatch.
```

```
line = "Beware the JubJub bird and shun"
```

```
longest = ""
```

```
longest = "Beware the Jabberwock, my son,"
```

```
longest = "the jaws that bite, the claws that catch,"
```

```
len(line) > len(longest)
```

```
32 > 42
```


x



```
for line in inputFile:
    if len(line) > len(longest):
        longest = line
```

Jabberwocky Solution Walkthrough

```
Beware the Jabberwock, my son,
the jaws that bite, the claws that catch,
Beware the JubJub bird and shun
the frumious bandersnatch.
```

`line =` 

```
line = "the frumious bandersnatch."
```

```
longest = ""
```

```
longest = "Beware the Jabberwock, my son,"
```

```
longest = "the jaws that bite, the claws that catch,"
```

```
len(line) > len(longest)
```

```
27 > 42
```

x

```
→ for line in inputFile:
    if len(line) > len(longest):
        longest = line
```

Announcements

- Homework 6 is out
 - Due by Monday (March 28th) at 8:59:59 PM
- Survey #1 will come out in the next week
 - Announcement will be made via Blackboard
 - Will be available on Blackboard

Practice Problem

- Write a function that takes in a string and counts how many special (`\n`, `\t`, and `\\`) characters it contains.
 - Remove all of those characters from the string.
 - Return the total count and the updated, “unspecial” string.

```
specialStr = "\t\n I like this and\\or \tthat.\n"  
count, unspecialStr = unspecial(specialStr)  
# count equals 5  
# unspecialStr is " I like this andor that."
```


More Practice Problems

- Update the Jabberwocky code to find the *shortest* line instead – think carefully about what you should initialize “**shortest**” to be.
- Write code that opens a file and prints out *every other* line, starting with the first line.
 - Think carefully about what method you use for reading in the lines of the file.