CMSC201 Computer Science I for Majors

Lecture 16 – File I/O

Last Class We Covered

- Using **`for`** loops
  - Syntax
  - Using it to iterate over a list
  - Using it for “counting” the number of actions
- The **`range()`** function
  - Syntax
  - Three forms: one, two, or three numbers

Any Questions from Last Time?

Today’s Objectives

- To learn about escape sequences
  - What they are and why we need them
  - How to use them
- To learn how to use the **`split()`** function
  - To break a string into tokens
- 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 <u>in front</u> 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 when using ' or ")

Common Escape Sequences

Escape Sequence	Purpose
`\'`	Print a single quote
`\"`	Print a double quote
`\\`	Print a backslash
`\t`	Print a tab
`\n`	Print a new line ("enter")

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

How Python Handles Escape Sequences

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

```
example1 = "dog\n"
example2 = "\tcat"
```

0	1	2	3
d	o	g	\n

0	1	2	3
\t	c	a	t

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 split on 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)

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

these character(s) are called the delimiter

```
>>> 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")
```

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

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

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

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???

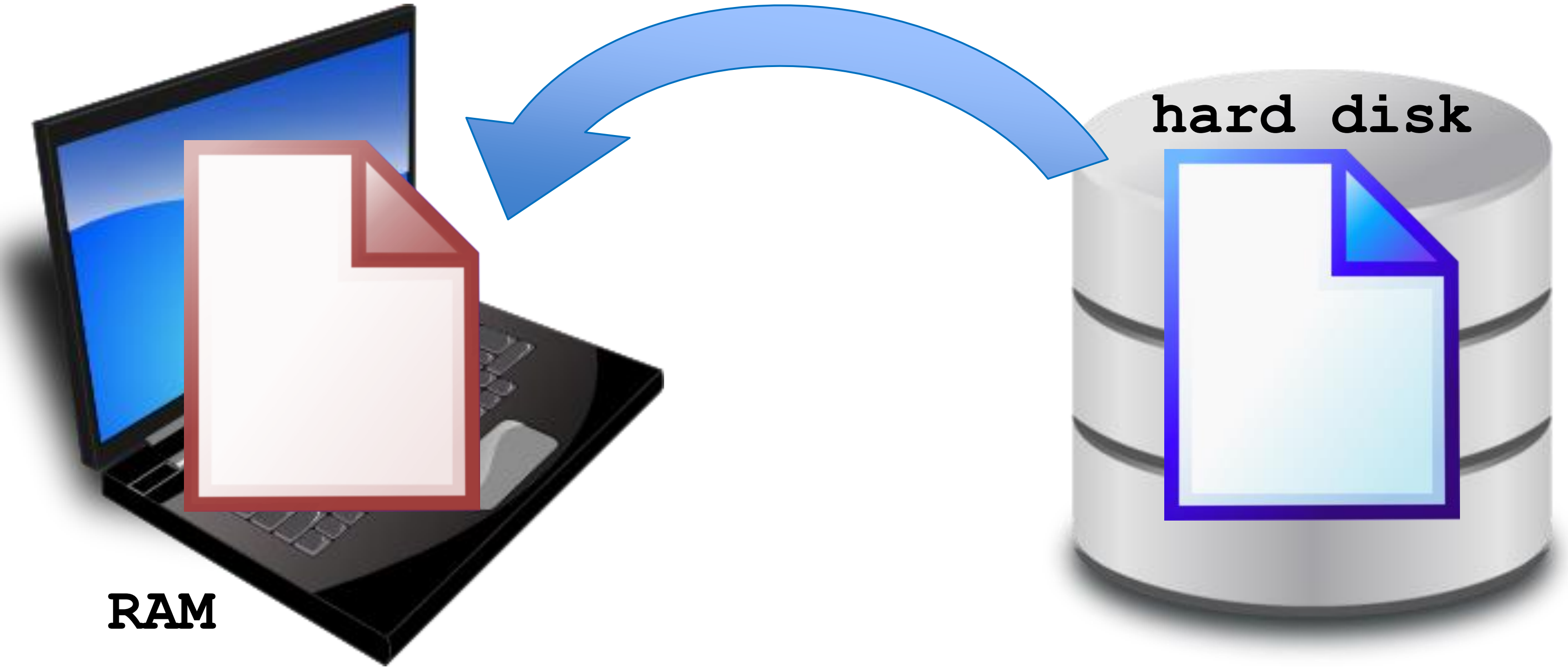
What is File I/O?

- 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

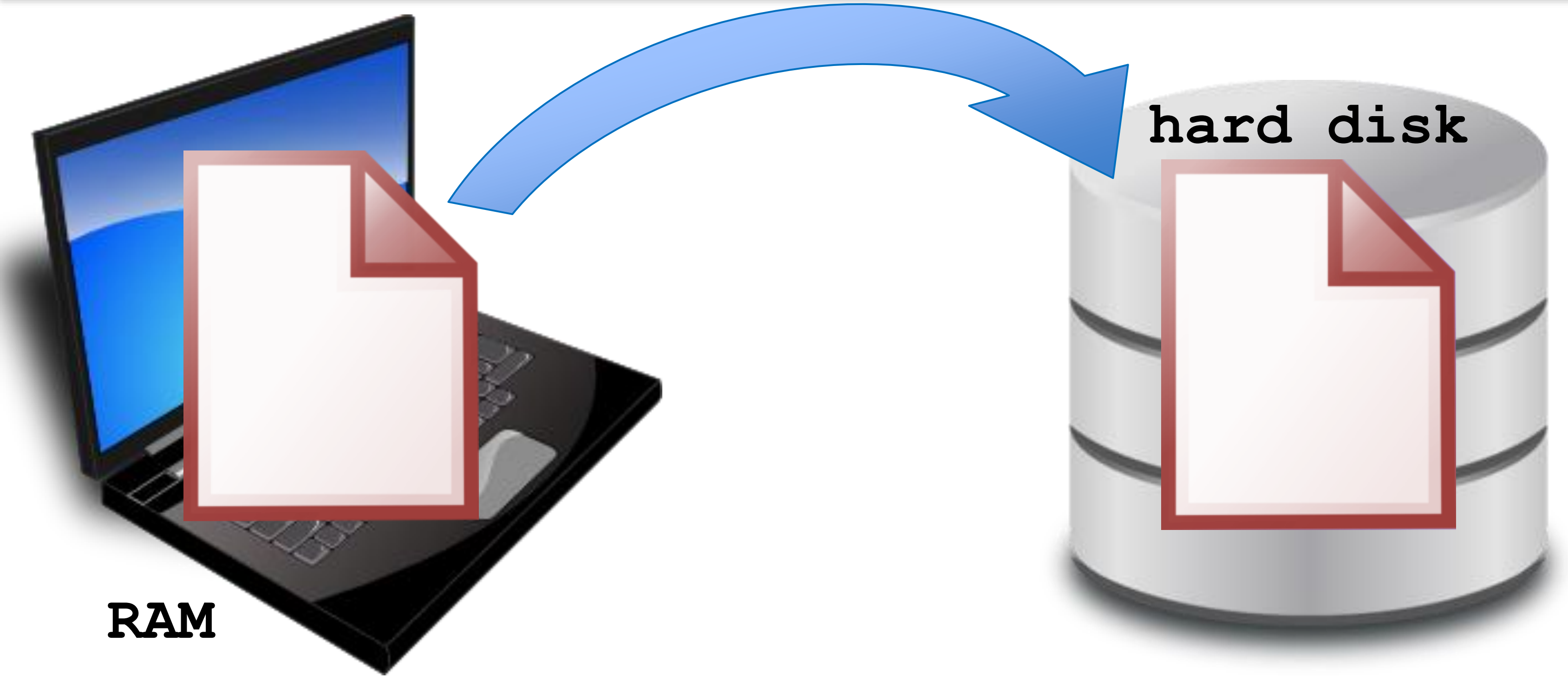
- "Reading" in a file using a word processor
  - File opened from hard disk
  - Contents read into memory (RAM)
  - File closed on hard disk
  - IMPORTANT: Changes to the file are made to the copy stored in memory, <u>not</u> the original file on the disk



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 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

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"`
  - `"numbers.dat"`
  - `"roster.txt"`

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("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
```

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 3.2\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!

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 3.2\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 3.2 3.1\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 3.2 3.1\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 **`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 Suzy 9.5 8.1 7.6 3.2 3.1

456 Brad 7.0 9.6 6.5 4.9 8.8

789 Jenn 8.0 8.0 8.0 8.0 7.5
```

UMBC

AN HONORS UNIVERSITY IN MARYLAND

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 Suzy 9.5 8.1 7.6 3.2 3.1

456 Brad 7.0 9.6 6.5 4.9 8.8

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

More About Whitespace

Whitespace

- Whitespace is any “blank” character, that represents space between other characters
- For example: tabs, newlines, and spaces

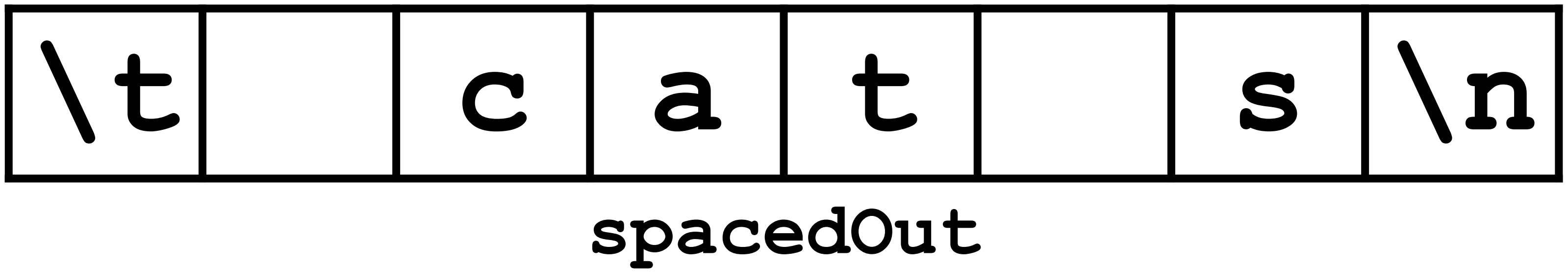
  `"\t"` `"\n"` `" "`

- When we read in a file, we can get whitespace
  - Sometimes, we don’t want to keep it

Removing Whitespace

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

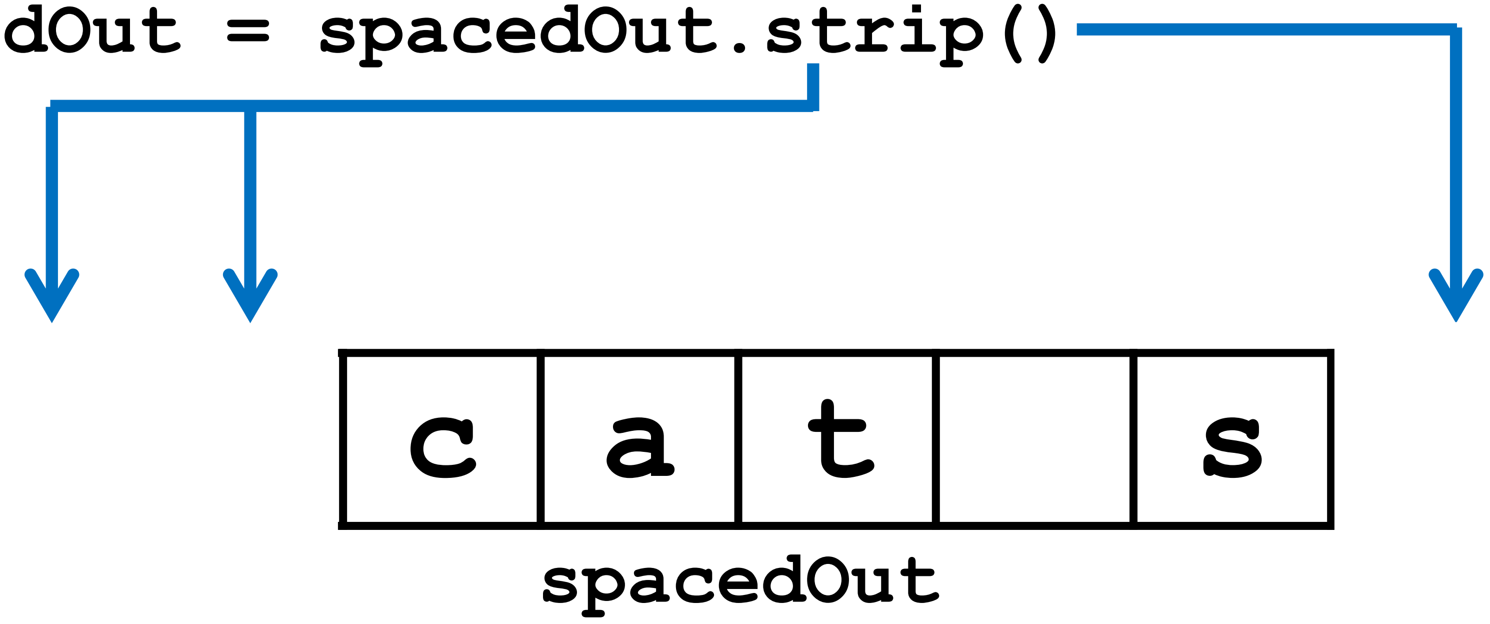
```
spacedOut = spacedOut.strip()
```



Removing Whitespace

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

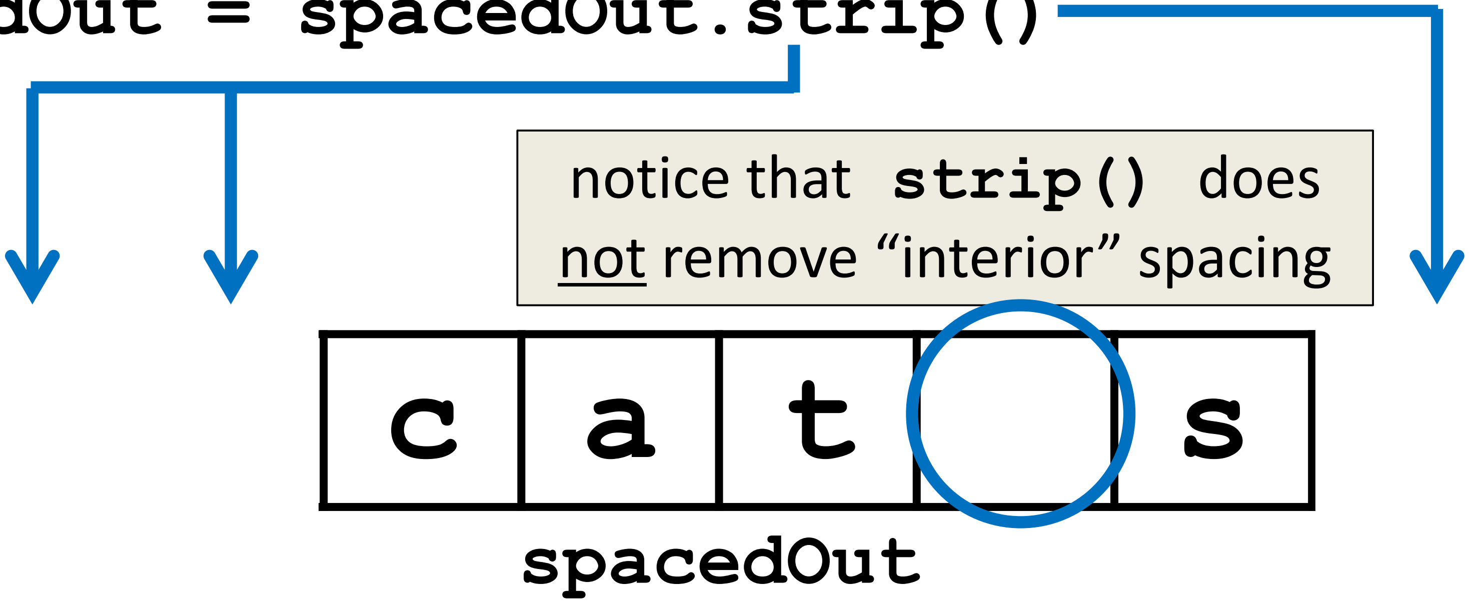
```
spacedOut = spacedOut.strip()
```



Removing Whitespace

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

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



Announcements

- HW 5 out on Blackboard
  - Due Friday, April 7th @ 8:59:59 PM
- Project 2 comes out Saturday
  - Uses 3D lists and file I/O
- Final exam is when?
  - Friday, May 19th from 6 to 8 PM