Debugging using Loop Exercises String Delimiters and Formats

CS 8: Introduction to Computer Science, Spring 2019
Lecture #10

Ziad Matni, Ph.D.
Dept. of Computer Science, UCSB

Administrative

- Homework #5 issued – due in a week
- Lab04 – due on Sunday by midnight (11:59 pm) on **Gradescope**!
- Midterm Exam #1 is graded
  - Grades will be released on Thursday morning
  - **Average** grade is **83** and **Median** grade is **87**

A Chance for Extra Credit!!!

- I will put up a link to a survey on Piazza (https://ucsb.co1.qualtrics.com/jfe/form/SV_b7ndMYpqi4hNVKR)
  - It's to get some midterm feedback from y'all on the class…
  - It's a very short survey (1-2 minutes) ***and*** it's anonymous!
  - Must be completed by Wednesday at 11:59 PM!

- If I get ***at least 80% of you*** to take it,
  I will give *everyone in the class* **+2 points** on their midterm #1 grade!!

- Elif I get ***at least 70% of you*** to take it,
  I will give *everyone in the class* +1 points on their midterm #1 grade!!

- Elif…
  - I got nothin'…

Lecture Outline

- Exercises with Loops
- String Formats

Exercise with Nested Loops

```
def drawRectangle(width, height):
""" print a rectangle with given width
    and height using the character *
    (instead of turtle)

    For example drawRectangle(5,3)
    should print
    *****
    *****
    *****
"""
```

Let's try it out!

Exercise with Nested Loops

```
def drawRectangle(height, width):
    for w in range(height):
        for h in range(width):
            print("*", end="")
        print("")
```

PLEASE NOTE THE INDENTATIONS!!!!

Accumulation Exercise 1

- Useful for "accumulating" something while going through a collection.

- Finish this function:

```
def countOddNumbers(lst):
    """ returns the number of odd numbers in lst """
```

Let's try it out!

Accumulation Exercise 1

- Finish this function:

```
def countOddNumbers(lst):
    """ returns the number of odd numbers in lst """
    oddItems = 0
    for item in MyL:
        if item % 2 == 1:
            oddItems += 1
    return oddItems
```

Accumulation Exercise 2

- Finish this function:

```
def countWords(sentence):
    """ returns the number of words in the string sentence """
```

Let's try it out!

Accumulation Exercise 2

```
def countWords(sentence):
    """ returns the number of words in the string sentence """

    wordCount = 1
    for c in sentence:
        if c == ' ':
            wordCount += 1
    return wordCount

# is there a case where this won't work?
```

The **.append** Function for Lists

- You can add items into a list by *appending* them to the end of the list

- Example: To grow `l = [1, 2]` into `l = [1, 2, 3]`
  you can do:

```
l.append(3)
```

- It's not the only way to "grow" a list, but it's easy and intuitive…

Accumulation Exercise 3

- Finish this function:

```
def createListOfOdd(lst):
    """ returns a new list that contains all """
    """ the odd numbers in lst """
```

Let's try it out!

Accumulation Exercise 3

- Finish this function:

```
def createListOfOdd(lst):
    """ returns a new list that contains all """
    """ the odd numbers in lst """
    newList = []
    for item in lst:
        if item % 2 != 0:
            newList.append(item)
    return newList
```

String Delimiters

- Recall that:

  “hello” and ‘hello’

  are the same thing

  (Python lets you use either single or double quote marks for string delimiters)

- They can even be used together, like this:

  s = “hello, I’m Joe” or

  s = ‘So I said, “Who are you?”’

- Otherwise, we’d have to use the \ (called “escape sequence”), like this:

  s = “So I said, \”Who are you?\””

Newlines in Python

- The most straight-forward way is to use the "**\n**" character
- Example:

```
>>> s = "How I wish you were here.\nWe're just two lost souls
swimming in a fishbowl,\nYear after year"


>>> print(s)
How I wish you were here.
We're just two lost souls swimming in a fishbowl,
Year after year
```

*Note: there's no need for a third **\n** here, because the print() function always puts one there, BY DEFAULT (it can be over-ridden)*

Alternative Way to Make Newlines

- You can ALSO define a string with triple double-quotes ("""), like this:

```
>>> s = """
How I wish you were here.
We're just two lost souls swimming in a fishbowl,
Year after year
"""

>>> print(s)
How I wish you were here.
We're just two lost souls swimming in a fishbowl,
Year after year
```

Recall: String Indexing & Slicing

- If `s = "hello"`
- Then `s[0] = "h"` , etc…

- The last character in any string is…
  `s[len(s) - 1]`

- In the example above, `s[0:3] = "hel"`
  - In other words, it goes from index 0 to index 2 (*one-before-3*)
- Also, `s[2:] = "llo"` (from 2 to the end)
- And, `s[:4] = "hell"` (from the beg. to 3)

Recall: Negative Indices in Strings

- If `s = "hello"`
- Then `s[-1] = "o"`
  `s[-2] = "l"` , etc…

- In the example above, `s[-2:] = "lo"`

  etc…

Slicing Works on Lists Too!

Example:

```
ThisList = [3, 4, "spaghetti", -5]
```

```
ThisList[0:2] = [3, 4]
ThisList[-2:] = ["spaghetti", -5]
```

The **.split()** Method for Strings

- You can **split a string into its component words** and then **place them in a list**
  - With ONE instruction!!

Example:

```
>>> s = "What about Bob?"
>>> l = s.split()
>>> print(l)
   ["What", "about", "Bob?"]
```

Note: the split is done on SPACE characters and these are NOT part of the collected sub-strings in the list!

The **.split()** Method for Strings

- The **default split** is on space characters (" ")
- You can over-ride that default and split on ANY string

Example:

```
>>> s = "What about Bob?"
>>> l = s.split('a')
>>> print(l)
    ["Wh", "t ", "bout Bob?"]
```

*Note: NOW the split is done on the '**a**' characters and these are NOT part of the collected sub-strings in the list!*

LET'S REDO THIS EXERCISE!!!

- Finish this function:

```
def countWords(sentence):
    """ returns the number of words in the string sentence """
    sum = 0
    MyNiceList = sentence.split()
    return len(MyNiceList)

    # SOOOO much easier!!!
```

YOUR TO-DOs

- ❑ **Homework** #5 due **Tuesday, 5/14**
- ❑ Finish **Lab4** (turn it in by **Sunday**)

- ❑ Ensure *(smiles / frowns) > 5.7*

</LECTURE>