**CSC 110  
Lab Activity -- Working with Strings**

Objectives:

* work with individual characters in a string
* optional: gain more practice working with files

For this lab exercise, you will manipulate a string in several ways.  You should create and upload one program that does all these things.  Develop your program progressively, adding one operation at a time and testing as you go.  Remember that your program needs to work with any phrase the user enters, not just the example shown.

1. Start by writing code that asks the user to enter a short phrase. (Let's assume the user enters "happy birthday".)
2. **Use a loop** to print the user's phrase, one character per line. Test this to make sure it works.
3. Modify your program so that it also prints the phrase all on one line in reverse order.  Our test phrase above would be printed like this:   
     
   yadhtrib yppah  
     
   Do this in two ways.
   1. First, use a **slice** to get the reversed string, and print that.
   2. Then, working with the original string again (without using a slice), use a **loop** to print one character at a time, all on the same line, but in reverse order.  (Hint: try using a COUNTING loop that counts through all the legal index values in reverse order.  If you are using a slice, you are missing the point.)  Remember that you can use the 'end' parameter when you call the 'print' function to keep each the characters all on the same line.  If you prefer, you could use a string accumulator -- see Note 1 below.
4. Modify your program so that it also prints the phrase one word per line, like this:  
     
   happy  
   birthday   
     
   Use an if statement inside a loop.

Test to see if the character is a space (just a string comparison to ' '), and when it is, start printing on a new line.

1. **Optional**: Add code to your program that asks the user to enter the name of a file. Then, print the contents of the file one **word** per line, but do not print any vowels ('aeiouAEIOU').  Assume that the file may have more than one word on each line, but your output should have just one word per line.
2. **Optional**: Working once again with the phrase the user entered in Part 1, add code that replaces any word beginning with the letter 'b' with the word "bee". To do complete word replacement like this, you will need to use a string accumulator to hold the word, as described in Note 1 below. You know a word is completed if you encounter a space OR the end of the message.  (Tip: add one space at the end of the user's phrase to ensure that every word is followed by a space.)  Here are a few examples:  
       
   -- "happy birthday" would be printed out as "happy bee"  
   -- "a bear likes honey" would be printed out as "a bee likes honey"  
   -- "bake better bread" would be printed out as "bee bee bee"

Write all your code in one .py file and upload the file before the deadline to get lab credit.

Note:

1. Here's how to use a string variable as an accumulator. First, be sure to initialize the accumulator before the loop to an empty string (two quote marks with nothing between them, not even a space).  Then, instead of printing each character one at a time, add each character to the accumulator. When you reach the end of a word, print out the word (or a substitute), and then reset the accumulator to an empty string.  (In other applications, you may choose to add a newline character ('\n') to the accumulator to go to a new line, and then just print out the string accumulator's value one time after the loop is finished.)

**Lab Activity -- More Work with Strings**

**Objectives:**

* continue practice working with **string objects**

**Instructions:**  
  
Visit the [CodingBat web site](http://codingbat.com/python), **log in to your account**, and complete the following exercises: **Warmup-2**: string\_times, front\_times, string\_bits, string\_splosion  
  
**String-1**: [do all that you have not already done]: make\_out\_word, extra\_end, first\_two, first\_half, without\_end, combo\_string, non\_start, left2  
  
**String-2**: double\_char, count\_hi (notice the hint), cat\_dog  
  
Remember that these activities require you to write function definitions that "return" a value. That might mean using a string variable as an accumulator.  
  
Remember that "Warmup" exercises have solutions available to check if you need a hint. The sample solutions illustrate the use of loops. Also, some regular exercises have a hint available.  
  
Note: You may have read about the Python "repetition operator" (\*) that can be used with strings. To get the most value from these practice activities, write solutions that do NOT use this operator.  
  
Feel free to post questions/comments about these exercises in the Discussion Forum.

## CodingBat Score -- Second Half

This is the place where you will get "quiz credit" for the CodingBat problems you have completed during the second half of the course (those related to lists and strings). Please check to make sure that you have completed all assigned problems:

* Python **Warmup-2** section: **7 exercises**(string\_times, front\_times, string\_bits, string\_splosion, array\_count9, array\_front9, array123)
* Python **String-1** section: **8 exercises** (make\_out\_word, extra\_end, first\_two, first\_half, without\_end, combo\_string, non\_start, left2)
* Python **String-2** section: **3 exercises** (double\_char, count\_hi, cat\_dog)
* Python **List-1** section: **4 exercises** (same\_first\_last, rotate\_left3, max\_end3, has23)
* Python **List-2** section: **6 exercises** (all of them)

Exercises are worth 1 point each except that "Warmup" exercises are worth half a point each.  
  
Exercises are due completed by 11pm Monday, 6/13 -- but complete them**before Test #3 so you can get maximum benefit** from them.