**An Experimental Lab**

**Due: October 14, 11:59 p.m.**

**Objective:** This lab is designed to help you learn about strings through experimentation and also give you the confidence that you can learn new programming features on your own. Since this is a “lab”, and not a programming assignment, you may talk to other people in the class about the assignment but **each** person is to write and turn in their own version of the program on their moodle directory.

**The Lab:** Strings are actually a special type of variable in Python called an ***object***. First, let’s consider a variable that holds a simple type of information, such as an integer. There are operations that you can apply to integers, like addition, subtraction etc., and you can use an integer value/variable when you call a function, like when you call the randint function to generate a random number.

Objects such as Strings share many of these same characteristics, but they also have some extra features. Like a variable that holds an integer, for instance, a variable that holds a String object can be used in an assignment statement (we have been assigning values to strings and printing out those values all along), and can be used in functions (for instance, you provide a string to be displayed as part of an input statement). For objects, however, there are also special pieces of code (like functions) that are ONLY associated with objects of type a certain type. These pieces of code are called **methods**. Thus, there are methods associated with Strings that can only be used with Strings. The use of these methods is specified by the variable name that is holding a string, followed by a period, followed by the method name and any parameters it might take. The general formant is

object\_variable.method\_name( optional parameters)

If the method takes no parameters, the parenthesis still need to be there, but they will have nothing inside of them. It’s just the rules of the language.

Actually, we have already been using objects and calling their methods. Surprise! For instance, a file variable is actually an object. So, if we had the statement

students = open (‘class\_list.txt’,’r’)

we would then actually use the close method:

students.close()

You can see that we have the method *close* (which can only be used with file variables) invoked by specifying the file variable (students), followed by a period, followed by the method name (close), and empty (), since there are no parameters for the close method. So its not hard…we’ve already been doing it.

Now let’s get back to Strings. A string is a sequence of characters, and that is how it is represented in the computer. Each character in the string can be referenced by its ***index***, which is its location in the string. Python starts numbering these locations at zero. When I trace code involving a String, I will be drawing it with a circle pointing to the String, to reinforce the idea that it is an object.

As stated above, individual characters of a String can be accessed by *indexing* their location, which is done by having the index value placed between square brackets after the String variable name:

String\_variable[index]

For example, the following code will generate the following output:

name = ‘Rose’

print name[0]

ind = 2

print name[2]

Output:

R

S

Be careful, the one thing you can not do with a String is use the index to assign a new value to an individual location of a String. For example, you can NOT do the following

Name[1] = ‘a’

The correct technical term is to say the Strings are *immutable* (individual parts of them can’t be changed).

**The len Function**

The length function will take a string as a parameter and return the length of the string as an integer. The format for a call is:

len(Sting\_variable)

But what exactly is meant by the length? Write and run a small Python program to see for yourself.

*Write a small Python program that will use the len function to return the next to the last letter of a String. What would you do if you wanted to make sure that the string had a “next to the last letter”, before trying to output it?*

**For loop Variation**

Python provides a for-loop variation that will access every character of a string in turn. Its format is

for loop\_variable in String\_variable:

this will place each character in turn into the loop\_variable. This loop has the same general concept as the for-loop variation we looked at in class to read each line of a file in turn.

***Write a small python program to print out each letter of a String on its own line.***

**Slicing**

Python provides a way to specify a portion of a string variable. This technique is called slicing and it is accomplished by specifying a start and end indices as follows:

String\_variable[start:end]

But what do the start and end really mean?

***Write a small Python program starting with the assignment***

***Letters = ‘abcdefgh’***

***Write a small test program that will try different values for start and end to see how they really work. What happens if start is the same as end? What about greater than end? What about if you specify an end that is longer than the string?***

Python also allows some “short hand” notation with slicing. If the start index is omitted, it is assumed the start point is the beginning of the String:

String\_variable[:end]

For example, Letters[:2]. If the end index is omitted, it is assumed that the end point is the end of the String:

String\_variable[start:]

For example, Letters[3:].

If a third parameter is added, it acts as a “step value” and will move ahead by the step value, skipping over parts of the string:

String\_variable[start:end:step\_value]

For example, letters[0:7:2].

***Write a Python program to test each of these variations. Text values to see what would happen in what you may consider to be error choices of indices. How does Python handle these cases?***

Finally, if a negative value is specified as a start or end value, it indicates a location *relative to the back* of the string (positive values indicating a location relative to the front of the string).

For example, Letters[-3:]

***Write a small Python program to test this variation.***

***Write a small Python program that will ask the user to input the name of a file, with an extension, and your program should output “This is a Python file” or “This is not a Python file” based on the whether or not the file name entered ended in “.py”extension . Be sure that your program will not think that a file named “happy” is a Python program.***

**in and not in Operators**

We are already familiar with the concatenation operator (+) which is used with Strings. Two additional operators are the **in** and **not in** operators. These are boolean operators and they take two Strings as operands. The format is

String1 in String 2

**in** will return true if String1 is contained in String2, and false otherwise. Conversely, **not in** will return true if String1 is not contained in String2, and false otherwise.

The following code example, shows how in could be used:

Line = ‘catch the dog’

if ‘cat’ in Line:

print ‘cat appears ’

else:

print ‘cat does not appear’

**String Methods**

Strings have a number of methods that can be used to:

* Test their values
* Perform modifications
* Search for substring/replace characters

As specified above, these methods are invoked follows:

String\_variable.method\_name(optional parameters)

A listing of python string methods can be found at:

http://docs.python.org/release/2.5.2/lib/string-methods.html

Please note that if the description says it “returns” some value, such returns a Boolean, you would use the method as you would any expression of that value, such as:

trythis = ‘abc123’

if trythis.isdigit():

print ‘string only contains all digits’

else:

print ‘string does not contain only digits’

If the description indicates that the method “returns a copy of a string”, it does returns a string as a result, which you can then use as any other string. Be careful, it is not changing the actual string that the method was invoked with. That string will remain unchanged unless you assign the result back to the original string.

For example:

Letters = ‘abc’

Nletters = Letters.upper()

This will assign ‘ABC’ to Nletters, but Letters will remain unchanged, still containing ‘abc’. If you wanted to change the value of Letters to all upper case, you would use the statement(s):

Letters = Letters.upper()

or, using two steps:

Temp = Letters.upper()

Letters=Temp

***Write a python program that will prompt the user for a file name. Make sure that you have already created a file by this name in the same directory as your program with several lines of text in it. After reading in the file name from the user, create a new file with the same name, but extension .out. Your program should to write each line of the specified input to the new output file, but all of the contents of the file should be in uppercase letters.***

For example, if the contents of the original file, named **labtest**, were:

See the dog

Run Down the Street

The file named **labtest.out** should contain

SEE THE DOG

RUN DOWN THE STREET

Make sure that this program is checked by the instructor or TA.

Drop a copy of this file in the StringsLab folder on moodle. Make sure it is properly documented.