

Laboratory Exercise: Introduction to Python

Overview

This assignment asks you to write three simple programs in Python. We will provide step by step guidelines for writing the first program, and some tips for writing the other two programs.

As before, you can do this assignment either in the computer lab or on your own computer, if you have one. In the latter case, you will need to install Python. To do this, go to the web page <http://www.python.org/download>, download the appropriate installer, and run it.

Starting up Python

On the lab computers, go to the Start menu, choose “All Programs”, then “Class Software”, then “Python 2.5”, then “IDLE (Python GUI)”. If you are on another computer, you may have to look elsewhere, depending on where you installed it. Look for the IDLE application, and execute it (by double-clicking).

What you see is the Python interpreter. Try typing ‘print “hello world”’ as follows (note: “>>>” is the prompt that the computer prints):

```
>>> print "hello world"
hello world
>>>
```

You just wrote a very short program to print something.

We would like to store longer programs in a file so that we do not have re-type them each time we want to use them.

Program #1: Converting Decimal to Binary

This program should print out your name, the date, and then computer and print out a binary representation of an integer between 0 and 31 inclusive. For this exercise, you can print the binary digits in reverse order. The name of file should be “binary.py”. What follows are step-by-step instructions to help you write this program.

1. Creating a new file.
 - a. Go to “File” on the Python GUI
 - b. Select “New Window”.
 - c. Now you should see a new window pop up. This is the script window, where you write your collection of Python commands.

Important: At this point, you should work through all the things described in pages 8–11 of the book. Also, please read pages 28–50. This lab will be easier if you do the reading before you try it.

2. Printing your name and today’s date
 - a. In the script window, use two print statements to type out your name and today’s date, using the wording of the example after 2.3. For example, my first line would be: print “My name is Matt Bishop”

As you type your statements, “print” will be in a different color from the quoted parts. The Python GUI uses different colors (called “syntax highlighting”) to help programmers differentiate between reserved words of the Python language (such as “print”) and everything else.

 - b. When you are done writing the two statements, save the file on the Desktop as binary.py
 - c. Select the “Run” menu, then “Run Module” (or press F5) to run your program. If you did things right, your output should look like this (of course, your program will display your name, and today’s date):

```
>>>
My name is Matt Bishop
Today is 5/11/2007
>>>
```

If you made a mistake in typing (a syntax error), the interpreter will highlight the offending part of the line in red. Before continuing, we want to you to learn about two common error messages.

- d. Create an error in a reserved word by changing one of your print statements to “Print” (that is, capitalize the first letter of the word “print”). Then press F5 (and agree to save your program). The interpreter will notify

- you that you have a syntax error. Click “OK”, and look at the program. It will highlight the end of the line where you changed “print” to “Print”. This demonstrates that the interpreter can guess where the error is located, but may highlight an area a little after the actual error. Once you see this, please change the “Print” back to “print”, and then press F5. Be sure your program runs properly!
- e. Now, create an error in a non-reserved word by deleting the double quote in front of “My” on the line that prints your name. You should only have one quotation mark at the end of the line. Then press F5 (and agree to save your program). The interpreter will again report that you have a syntax error, and will highlight “name”. This again demonstrates that the interpreter can guess where the error is located, but may highlight an area a little after the actual error. Now, please add the missing double quote, and then make sure your program runs properly
3. The hard part of this assignment is to convert an integer to binary. First, think of a number between 0 and 31 inclusive. Your program will convert this number to binary, after printing it.
 - a. Create a variable named number and assign your chosen number to this variable.
 - b. Use a print statement to print out “My number is” followed by the number. Do this as follows:


```
print "My number is", number
```

Note that you can use one print statement to print several things on the same line. When printed, a blank space separates them. Also, put a comma “,” between them in the print statement.
 - c. Now print the message “This number, in binary with the digits in reverse order, is:”
 - d. Create a variable called remainder and assign the result of number % 2 to this variable. The “%” operator means to calculate the remainder of number / 2. This would look like:


```
remainder = number % 2
```
 - e. Now assign the result of number / 2 to the variable number. It may seem a little odd that result will appear on both sides of the equals. The interpreter has no problem with this statement because it will do the division by 2 before it assigns the new value to result.
 - f. Print the value of remainder
 - g. Repeat the sequence of steps d, e, and f four more times.
 4. To run your program, press F5 or go to Run->Run Module. Here is a sample output:


```
>>>
My name is Matt Bishop
Today is 5/11/2007
My number is 28
0
0
1
1
1
1
>>>
```

Program #2. Capitalizing The Vowels

This program asks the user to type some text. It then prints what the user typed in quotation marks, followed by what the user typed with all vowels (the letters “a”, “e”, “i”, “o”, and “u”) capitalized¹ and in quotation marks. The name of your program file should be capvowels.py. Your prompts and output should match the example below. We will not give you step by step direction for this program, but will provide some hints.

1. Open a new file, and save it as capvowels.py
2. Before writing the actual program, place your name in a comment at the top of the file. In Python, comments start with a #. For example, for me, I would write

```
# Author: Matt Bishop
```

1. Yes, “y” and “w” can sometimes be vowels, and sometimes consonants. Writing a program to figure out when those letters are vowels is hard, so we’ll just treat them as consonants.

3. Use the Python built-in function `raw_input` to write a prompt and read input from the user. That function prints out the prompt string you give it, and returns the data the user types after the prompt. Store the returned values in a variable with a name of your own choosing.
 - a. For example, I chose `line` as the name of my variable. So my input line would be:


```
line = raw_input("Please type your text: ")
```
 - b. Note that I put a space after the colon in the prompt so that what the user types is not run up against the prompt. Try leaving out the space and see how it looks.
4. Now we're going to put quotation marks around the input. This requires *concatenating* three strings: one containing the opening quotation mark, one containing the input string, and one containing the closing quotation mark.
 - a. Putting a quotation mark into a string seems easy: just surround it by quotation marks. But that would put three quotation marks in a row, and the interpreter would think the second one terminated the string (which would be read as an empty string). So, you *escape* the inner quotation mark with a backslash "\". The string containing a single quotation mark is "\"".
 - b. String concatenation means putting two strings together. In Python, you use the "+" sign to do this. So, to put a quotation mark in front of the string stored in the variable `line`, do this:


```
line = "\"" + line
```

 There is no space between the first string and the second. So the quotation mark will be followed by the first character in the string stored in `line`.
 - c. Now append a quotation mark to the string you read in.
5. Next, print the string that was read in, as follows. Let `instr` be the string you read in. Then the output should be


```
The original line was "instr"
```
6. Now replace the vowels, one at a time. Do this using 5 assignment statements.
 - a. For the first statement, replace every "a" with "A". Use the "replace" method described on p. 39 of the text. The assignment looks like this in my version of the program:


```
line = line.replace("a", "A")
```
 - b. Do this for "e", "i", "o", and "u".
7. Finally, print the resulting string with the words "The transformed string is".
8. Press F5 to save and run your program. Your prompt should match the one in the example below. The input was "the quick brown fox jumped over the lazy dog".


```
>>>
Please type your text: the quick brown fox jumped over the lazy dog
The original line was "the quick brown fox jumped over the lazy dog"
The changed line is "thE qUIck brOwN fOx jUmPEd OvEr thE lAzY dOg"
>>>
```

Program #3: Computing A Tip

Challenge 3 on p. 50 of the Python text says: "Write a Tipper program where the user enters a restaurant bill total. The program should display two amounts: a 15-percent tip and a 20-percent tip." The name of your program file should be `tipper.py`. Here are additional specifications that your program must follow:

1. Place your name in a comment at the top of the file.
2. Read the user input into a variable named `price`.
3. To compute the 15-percent tip, multiply that variable by 0.15.
4. To compute the 20-percent tip, multiply that variable by 0.20.
5. Your prompts and output should match mine exactly.
6. Test your program with the numbers shown in the example below.

You will need to enclose your `raw_input` function within a float conversion function. There is an example of how to do this using an integer conversion function on p. 45 of the text:

```
rent = int(raw_input("Manhattan Apartment: "))
```

Look in the book to find the float conversion function.

Here are some sample runs. Remember, when given the inputs following the word “Price: “, your program must give exactly the same output as this example.

```
>>>
Price: 100
For 100.0 a 15% tip is 15.0
For 100.0 a 20% tip is 20.0
>>> ===== RESTART =====
>>>
Price: 39.99
For 39.99 a 15% tip is 5.9985
For 39.99 a 20% tip is 7.998
>>> ===== RESTART =====
>>>
Price: 23.86
For 23.86 a 15% tip is 3.579
For 23.86 a 20% tip is 4.772
>>>
```

Turning This In

To hand everything in, create a ZIP file named lab7.zip containing the three python files. In the labs, you can use the FilZip utility to create the ZIP file. Go to Start, then to All Programs, then to Utilities, then to FilZip. In FilZip, go to File, then to New Archive, and create lab7.zip on the Desktop. Click the Add (+) button and select the files. Note that you must click two Add buttons before they are added to the archive. Save your zip file in MySpace, and also submit it to MyUCDavis.