Introduction to Python

This assignment asks you to write three simple programs in Python. This handout provides 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 Python

On the lab computers, go to the Start menu, choose All Programs, then Class Software, then Python 3 (there may be numbers following the "3", depending on what version is installed; do *not* use "Python 2" if you see it), 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: Making Change

This program should print out your name, the date, and then the number of quarters, dimes,, nickels, and pennies required to make change for any amount of currency less than a dollar. Here is an example; the number "67", italicized on the third line, is what I typed as input:

```
My name is Matt Bishop
Today is 2/20/2011
Enter an amount between 0 and 99: 67
To give someone 67 cents in change, give them:
2 quarters
1 dimes
1 nickels
2 pennies
```

The name of your file should be "change.py". What follows are step-by-step instructions to help you write the program.

1. Create 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.
2. Print your name and today's date
   (a) In the script window, use two print statements to type out your name and today's date. For example, my first line would be: '`print("My name is Matt Bishop")`'

   As you type your statements, "print" may be in a different color from the quoted parts.[1] The Python GUI uses different colors (called "syntax highlighting") to help programmers distinguish functions and keywords in 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 "change.py".

[1] Some versions of the Windows interpreter may not do this. If not, don't worry.

(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 2/20/2011
>>>
```

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 by changing one of your print statements to "Print" (that is, capitalize the first letter of the word print). Then run your program (and agree to save your program). The interpreter will notify you that you have a syntax error, with a message like this:

```
>>>
My name is Matt Bishop
Traceback (most recent call last):
  File "/Users/bishop/change.py", line 2, in <module>
    Print("Today is 2/20/2011")
NameError: name 'Print' is not defined
>>>
```

This message tells you that line 2 of your file "/Users/bishop/change.py"[2] has an error. It displays the line, and then specifies the error. That's the last line in the above message. Look at what follows the colon ":" on that line[3] To fix the problem, go to line number 2 in your file. The error is that "Print" is not defined. At this point, you need to know that your program must invoke "print" (with a lower-case "p") to print something. Change "Print" back to "print" and rerun the program, saving your file when asked.

(e) Now, create a syntax error by deleting the double quote in front of "My" on the line that prints your name. You should only have one quotation mark, which is at the end of the line. Then run your program (and agree to save it). The interpreter will again report that you have a syntax error , and will highlight name. In general, when you make a syntax error, the interpreter will highlight the offending part of the line in red. Sometimes it highlights the exact location of the problem, but more often it highlights a location close to, but after, the problem (because it doesn't detect the problem until after the problem occurs). This is an example. The error actually occurs before "My", but the interpreter doesn't recognize it until it sees the next word, "name"—so it highlights "name". Now, please add the missing double quote, and then make sure your program runs properly.

3. The hard part of this assignment is to produce amounts of change. First, think of a number between 1 and 99 inclusive. Your program will convert this amount of money into quarters, dimes, nickels, and pennies, after printing it.

(a) Create a variable named "amount" and assign a number to this variable.

i. Use the Python built-in function `input()` to prompt the user for input, and then read the input.[4] That function prints out the prompt string you give it, and returns the data the user types after the prompt. This data is returned *as a string*. So you would say:

```
input("Enter an amount between 0 and 99: ")
```

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.

ii. Now you need to convert the input the user types into an integer and assign it to `amount`. To do this, use the built-in function `int()` to convert the string to an integer, and then simply assign the result:

```
amount = int(input("Enter an amount between 0 and 99: "))
```

Try typing in something that is not an integer, and see what happens.

(b) Use a "print" statement to print out "To give someone", your number, "cents in change, give them". Do this as follows:

[2] I saved my file in the directory "/Users/bishop", so your file name will undoubtedly be different.

[3] The name of the exception comes before the colon. Here it is "NameError". For now, ignore this. You will see how to use it soon.

[4] In Python 2, you would use the built-in function `raw_input()` instead of `input()`; everything else is the same.

```
print("To give someone", amount, "cents in change, give them:")
```
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 figure out the number of quarters in that much change. As a quarter is 25 cents, we need to figure out how many times 25 can go into the amount. So, print the value `amount // 25`, followed by the word "quarters". This looks like:
```
print(amount // 25, "quarters")
```
Note the *two* "/"s. If you use only one, this won't work![5]

(d) Next, as we've given change for part of the amount, we need to reduce the amount by the change we've given. To do this, we need how much is left over after deducting the change—which is the remainder of the above division. So assign the remainder to the variable `amount`:
```
amount = amount % 25
```
It may seem a little odd that `amount` appears on both sides of the equals. But remember that "=" is not *mathematical equality*; it is *assignment*. The interpreter has no problem with this statement because it does the right-hand side (here, take the remainder) before it assigns the new value to `amount`.

(e) Repeat the previous two steps for dimes (10), nickels (5), and pennies (1). Note you can omit the last step for pennies, as there will be no remainder.

4. Now run your program. Try giving it 67 as input and see if your output matches what is shown above.

That's it! You have finished your first Python program. Congratulations.

Program #2: Vowels and Consonants

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") capitalized[6], all consonants in lower case, and the resulting text in quotation marks. The name of your program file should be "capsmall.py". Your prompts and output should match the example below. We will not give you step by step directions for this program, but will provide some hints:

1. Open a new file, and save it as "capsmall.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
```

3. Use the Python built-in function "input" to write a prompt and read input from the user. Unlike the previous case, you want the string the user types, so you don't have to convert it to anything. Store the input in a variable with a name of your own choosing. For example, I chose `line` as the name of my variable, so my input line would be:
```
line = input(Please type your text: ")
```

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
```

[5] This is new to Python 3. In Python 2, you would use one "/" only.

[6] Yes, "y" and "w" can sometimes be vowels and sometimes be consonants. Writing a program to figure out when those letters are vowels is hard, so we'll just treat them as consonants for this exercise.

There is no space between the first sting 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. Suppose the user typed "InstRing". The output should be:

```
The original line is "InstRing"
```

6. Now we make the string all lower-case. This is easier than changing the case of each consonant, one at a time! To do this, we use the *string method* `lower()` described on p. 38 of the text. A method is like a function, but has a slightly different syntax:

```
line = line.lower()
```

Notice that, unlike "print", which prints what follows it in parentheses, the method follows what it operates on, with a period "." between them.

7. Now replace the lower-case vowels with capitals, one at a time. Do this using 5 assignment statements.

(a) Use the string method `replace()` described on p. 38 of the text. In my version of the program, this looks like:

```
line = line.replace("a", "A")
```

(b) Do this for "e", "i", "o", and "u".

8. Finally, print the resulting string with the words "The modified line is".

9. Save and run your program. As an example, the input "THE quick BROWN fox JUMPED over THE LaZy DoG" should produce (what the user typed is shown in slanted type, below):

```
Please type your text: THE quick BROWN fox JUMPED over THE LaZy DoG
The original line is "THE quick BROWN fox JUMPED over THE LaZy DoG"
The modified line is "thE qUIck brOwn fOx jUmpEd OvEr thE lAzy dOg"
```

Program #3: Sales Tax

On the UC Davis campus, sales tax is 8.25%. In the city of Davis, it is 8.75%. Write a "sales tax" program where the user enters a price. The program should display two amounts: the sales tax on the UC Davis campus and in the city of Davis. The name of your program file should be "salestax.py". Here are additional requirements your program must meet:

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 sales tax on the UC Davis campus, multiply that variable by 0.0825.
4. To compute the sales tax in the city of Davis, multiply that variable by 0.0875.
5. Your prompts and output should match the ones shown in the example below exactly.
6. Test your program with the numbers shown in the example below.

You will need to enclose your `input()` function within a float conversion function. You do this the same way you converted the input to an integer in the first exercise, except you use the float conversion function instead of `int()` (look on p. 43, in Table 2.4).

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
On the UC Davis campus, the sales tax for 100.0 is 8.25
In the city of Davis, the sales tax for 100.0 is 8.75
>>> ================================ RESTART ================================
>>>
Price: 39.99
```

```
On the UC Davis campus, the sales tax for 39.99 is 3.299175
In the city of Davis, the sales tax for 39.99 is 3.499125
>>> ================================ RESTART ================================
>>>
Price: 45.83
On the UC Davis campus, the sales tax for 45.83 is 3.780975
In the city of Davis, the sales tax for 45.83 is 4.010125
>>> ================================ RESTART ================================
>>>
Price: 0.50
On the UC Davis campus, the sales tax for 0.5 is 0.04125
In the city of Davis, the sales tax for 0.5 is 0.04375
>>>
```

What to Turn In

Please turn in all three programs to SmartSite. Then you are done. Please remember to use the Start button to log out from the lab computers!