Assignment P3

Due Date: June 10

Purpose

The Python adventure continues! In this assignment, you will use Python's selection statements in an *almost* real kind of program. You will get your input from a file rather than the keyboard. You will also continue to use basic Python statements and work on your coding style.

Problem

The owner of Norton's Cumooberland Farmes convenience store and gas station just installed new phone readers to make payment for customers purchasing gas, snacks, and other stuff easier. The customer just shows their phone with the *MooApp* and the purchase price of the gas and/or other item(s) is deducted from their account. However, this new hardware requires new software. Because you know your way around sales (falingies and rebar), you have been hired to write a program that simulates the charges and the creation of a sales slip. This prototype can then be implemented on their hardware.

Input

As part of the simulation, a file will be used to simulate the cost of the items and the hardware that reads information from the customer's phone and the gas pump. Your program should only prompt for the name of the input file (a string). You should assume the input file is in the same folder/directory where the Python code is stored.

The file will contain the following items, one data item per line, in this **specific order**:

- The character code for the gas type (see section on **Specifics**, below). Because the pumps are made by different manufacturers, both upper- and lower-case input is acceptable. Note that one of the codes shows that no gas was pumped, so the customer is only purchasing other items.
- A string that represents whether the customer is in the *MooClub* (`"Moo"`) or not (`"No"`). A MooClub member gets a 5% discount on the total gas purchase.
- The number of gallons of fuel purchased (a float). Note this could be 0.
- The total price of additional items purchased (a float). Note this also could be 0.

Output

Suppose the input file contained:

```
R
Moo
11.0
5.49
```

The first item is the gas type, the second notes that the customer is a MooClub member, the third is the number of gallons pumped, and the last is the amount spent in the store. With this data, Your program should produce a neat, aligned sales slip similar to the the output at the top of the next page.

```
Sales Slip ~~~~~~~~~
11.0 Gallons Regular @ 2.279 ....... 25.07
MooClub Discount (5%)..............  1.25
Cost of additional items...........  5.49
Tax (6.25%)........................  0.34

TOTAL deducted from account......... 29.65

Thank you for shopping at Cumooberland Farmes
```

If the customer is not a MooClub member, then that line item should not be displayed. If no gas was purchased, then all of the items pertaining to gas should not be displayed. Similarly, if no additional items were purchased, then those lines should not be displayed. In other words, the sales slip should show only those items that contribute to the total amount purchased.

Although the form of the slip does not have to be identical to the above, you must display all of the labels shown (such as "Regular" for the gas type, above), print all floating point values with two decimal places aligned by the decimal point, and follow the **Specifics** below.

Specifics

- You must use both `if-else` and `elif` forms of selection.
- Both upper- and lower-case character input is acceptable for the gas code.
- You can assume that the input is correct; no error checking is required.
- Valid gas codes are as follows (with associated prices and type):

Code	Price/Gallon	Type
N or n	0.000	No gas was pumped
R or r	2.279	Regular
P or p	2.439	Plus
S or s	2.619	Super
U or u	2.689	Ultra

- The MooClub discount is 5%; the MA tax is 6.25% on items purchased in the store.[1]
- You **must** worry about round-off error. Every multiplication or division computation involving a float can give you a result with many decimal places. You must round off to the nearest penny **at each stage of your computations**. You may use the `round()` function or you can use your own code for this. Do not rely on output formatting, because that makes the output only *look* pretty; the computations will still have all of the decimal places and your final answer may not be correct.
- Use "named global constants" for relevant items.
- Be sure to follow all style guidelines.

[1] We are not taking non-taxable items into account.

Notes

- As with the last program, writing an algorithm before attempting to code the program will show you where some of the problems will arise (remember, however, that Python details are left out of an algorithm). Or you may like to try out the idea of a flow chart as shown in the text; this can help you see where the selection statements will fit into your program. I also suggest that you code one portion of the program at a time, making sure that the newest piece you just typed in works properly before continuing. Be especially careful about getting the input from the file correctly.
- Test adequately! Check your results!! A sample input file is available on the course web page to help you format your input.
- As usual, the name of your program file should be your last name + project number; e.g., `gousieP3.py`.
- Send your completed Python program as an email attachment to mgousie@wheatoncollege.edu. Submit by 11:59:59 PM on the due date for the project to be on time.
- Type and e-sign the Wheaton Honor Code Pledge in a comment in your code: "I have abided by the Wheaton College Honor Code in this work."

I love deadlines. I like the whooshing sound they make as they fly by.
– Douglas Adams (1952-2001)