CS 8: Introduction to Computer Programming with Python
Summer 2016

Project 1

Assigned: Thursday, May 25 **Due:** Thursday, June 15 11:59 PM

Overview

In this assignment, you will program a price calculator that adds up a series of prices, including sales tax, something like a cash register.

First, the program will prompt the user for the tax rate. If the entered rate is valid, the program will then prompt the user to enter the prices of a number of items. Some items may be taxable, while others are non-taxable. After all prices are entered, the program will print the subtotal (the total of all of the prices before tax), the total amount of tax charged (the total of all of the taxable items, times the tax rate), and the total amount of the transaction (the sum of the previous two lines).

Rather than write this program all at once, you will program one task at a time, building up to the fully working end result. You will use problem-solving techniques, such as *reducing the problem*, that we have discussed in lecture. If you have trouble programming the control structure that connects all of the various parts of this program, you may wish to draw a flowchart as described in the textbook and in lecture.

You will need to show your work to your TA as you complete each activity; you will be graded on **both** the final code submission (65%) **and** the completion of your activities in recitation (35%).

You should try hard not to fall behind on the project. If you do not complete a session's activities during the session, you should complete them on your own before the next session and/or attend office hours for help. **No late assignments will be accepted.** This applies to both activities and final code submissions. By Thursday, June 15, you must:

- Show the result of each activity to the TA.
- Upload your code to Pitt Box, to the **provided** folder named `cs8-proj1-abc123`, where `abc123` is your Pitt username.

Note that you may submit to Box multiple times; you will be graded only on the last submission made before the deadline. You are encouraged to submit early and often, even if you are not finished, rather than risk getting a 0 by failing to submit on time.

Lab Session 1, May 25

Activity 1

In this activity, you will tackle a simplified version of the sales tax calculator that handles only a single item at a time. You will use *input*, *calculations*, and *output* to determine the total price of a single item, including sales tax. This experience will be helpful in solving the larger problem of calculating tax on many items.

1. Create a new Python program. Write the code to prompt the user for the tax rate. Store the entered **floating point** value in a variable named `tax_rate`.
2. Write the code to prompt the user for the item's price. Store the entered floating point value in a variable named `taxable_total`.
3. Write the code to determine the sales tax owed on `taxable_total` based on the tax rate stored in `tax_rate`. Store the resulting value in a variable named `tax`.

   Hint: The tax rate entered will represent a percentage (e.g., `8` for 8%), but will need to be divided before being multiplied by `taxable_total` (e.g., `0.08` should be multiplied for 8% tax).
4. Print the subtotal (since you have only one item, this is just the price of that item); the sales tax (`tax`); and the total amount owed (the sum of `taxable_total` and `tax`). Use the `format` function to ensure the output correctly shows 2 decimal places for all monetary values.

Here is the output from several example runs of the desired program:

```
Tax rate (%): 5
Price ($): 15.20

Subtotal: $15.20
Tax: $0.76
Total: $15.96
```

```
Tax rate (%): 7.5
Price ($): 12

Subtotal: $12.00
Tax: $0.90
Total: $12.90
```

```
Tax rate (%): 0
Price ($): 75.99

Subtotal: $75.99
Tax: $0.00
Total: $75.99
```

Once you have completed these steps, show your TA your progress so that you get credit for completing the activity.

End Lab Session 1

Lab Session 2, June 1

In this lab session, you will add several features to your program, using the progress you've made on the reduced problem. You will use *decision structures* to check for errors, and handle untaxable items.

Activity 2

In this activity, you will ensure that the tax rate is within a reasonable range. The highest sales tax rate in the United States is about 14%. Your program should print an error if a tax rate is entered that is above 20% or below 0%, as either of these conditions likely indicates that the user made an error.

Add an `if-else` construction that checks the value of `tax_rate` after it is input by the user. If it is less than 0 or greater than 20, you should print an error. Otherwise, you should continue with the calculations as before. Part of this exercise will be determining *where* in your code (i.e., *when* during execution) you should perform this check.

Here is the output from several example runs of the desired program:

```
Tax rate (%): -5
Tax rate is outside of range.
```

```
Tax rate (%): 54
Tax rate is outside of range.
```

Once you have completed these steps, show your TA your progress so that you get credit for completing the activity.

Activity 3

In this activity, you will add an additional prompt for the user to enter whether the item is taxable. If it is, you should continue as before. If it is not, you should continue without adding tax to the item's price.

1. After accepting the price of the item from the user, add an input prompt that asks the user whether the item should be taxable. An entry of `y` should be considered a yes (taxable), and anything else can be considered a no (non-taxable).

2. Alter you program so that it takes into account the value input by the user in the previous step. Use another `if-else` construction.

   Hint: You may want to copy your program that *always* adds tax, then modify it so that it *never* adds tax. Then, consider the differences between these two versions, and use decision structure to switch between these two types of behavior.

Here is the output from several example runs of the desired program:

```
Tax rate (%): 8
Price ($): 24
Taxable? (y/n): y

Subtotal: $24.00
Tax: $1.92
Total: $25.92
```

```
Tax rate (%): 7.5
Price ($): 75
Taxable? (y/n): n

Subtotal: $75.00
Tax: $0.00
Total: $75.00
```

Once you have completed these steps, show your TA your progress so that you get credit for completing the activity.

End Lab Session 2

Lab Session 3, June 8

In this lab session, you will identify the portions of your code that should be repeated in order to consider multiple items. You will use *repetition structures* to allow the user to enter multiple items, allowing for each individual item to be taxable or non-taxable.

Activity 4

In this activity, modify your program to prompt the user for how many items will be input, then ask for the prices (and taxable/non-taxable status) for each one. Calculate the subtotal, tax, and total accordingly.

1. After accepting the tax rate from the user, prompt the user for the number of items that they would like to enter. Store this value in `num_items`.
2. Use repetition structure to repeat the per-item prompts to the user. This includes the price prompt as well as the inquiry to determine whether the item is taxable. Your price prompt **must** specify the item number each time, so that the user can keep track of which item is being entered.
3. Adjust the calculations to handle the repetition. Separately add up the taxable items and the non-taxable items. Use **accumulator** variables named `taxable_subtotal` and `nontaxable_subtotal`. Be sure to initialize these variables, and increase their values in the right spot. After the two subtotals are calculated, print out the full subtotal, full amount of tax (on the taxable subtotal only), and the total amount due.

Here is the output from an example run of the desired program:

```
Tax rate (%): 8
Number of items: 3
Price of item #1 ($): 10
Taxable? (y/n): y
Price of item #2 ($): 15
Taxable? (y/n): n
Price of item #3 ($): 75.50
Taxable? (y/n): y

Subtotal: $100.50
Tax: $6.84
Total: $107.34
```

Once you have completed these steps, show your TA your progress so that you get credit for completing the activity.

Bonus Activity

This optional activity is less guided. Completing it will result in up to 8 bonus points on your submitted code. In this activity, you will use a *sentinel value*, so that the user does not have to know how many items he/she will be entering in ahead of time.

Review your notes and textbook regarding sentinels. Remove the prompt to the user that requires them to know how many items will be entered. Instead, allow the user to type the sentinel price `-1` when there are no more items. (You should not count the value `-1` toward any totals.)

You do not need to show your TA the result of completing this activity. Instead, simply submit your code with this method of repetition rather than a count-based repetition.

End Lab Session 3