CS 172 – Lab 2: Using Classes

Lab Overview

In this lab, we will experiment with creating custom classes. Ultimately, we will create a receipt for a purchase. Since we may want to have several receipts, and each receipt contains items, this seems like a good opportunity for object-oriented design and programming (receipt are objects, each of which contain item objects).

Supplemental Information

String Formatting: Creating nicely formatting strings is built into Python 3. You can read about the syntax here: (https://www.w3schools.com/python/ref_string_format.asp). Blank spaces that you want filled in are given with `{}`. Within the brackets, you can describe how the data is shown.

```
>>> "I just bought ${:.2f} in goods and services.".format(19.7863)
'I just bought $19.79 in goods and services.'
>>> "{:+<25}".format("Oranges")
'Oranges++++++++++++++++++'
>>> "{:+>25}".format("Oranges")
'++++++++++++++++++Oranges'
```

In the above examples, the **{}** denote a blank that should be filled. The **:** says we are going to format that value. The **.2** says show two decimals. The **f** says the number is a float. `{:<25}` Left aligns Oranges and uses a printing field of 25 characters. `{:+<25}` Left aligns Oranges, uses a printing field of 25 characters and fills any the unused space with the `+` character.

Getting the Date: Python 3 has a `datetime` library for working with dates. To get a timestamp for the current date, you can use the following code.

```
>>> import datetime
>>> str(datetime.datetime.now())
>>> '2017-10-17 13:30:28.983682'
```

Milestone 1: The `Item` Class

The first thing we'll need to do is design and implement an items class! We can then instantiate it to create the item objects that are part of the receipt. Define the `Item` class in the file named **item.py**. Please see template file provided.

The `Item` class should have three attributes.

- `__name`: A string with the item name.
- `__price`: A float with the item price in dollars.
- `__taxable`: A `Boolean` that is `True` if the item is taxed.

The `Item` class should have the following methods.

- `__init__`: The constructor. Takes data for all three attributes as parameters. These parameters should be in same type as their associated attributes.
- `itemToString`: returns the item as a string.
- `getPrice`: return the price of the item.
- `getTax`: Takes the tax rate as a floating-point parameter. Returns the tax charged on the item.

Once you have implemented the `Item` class, go to the bottom of the file. You will find some hardcoded tests for the class. Run **item.py** in Thonny and see if these tests pass. If so, this is a good point to submit your code for up to 40 points! You will get partial lab credit for completing this part, even if you do not complete the rest of the lab.

Milestone 2: The `Receipt` Class

Next, we want to design and implement a `Receipt` class! Define the `Receipt` class in the file named **receipt.py**. Please see template file provided.

The `Receipt` class should have two attributes.

- `__tax_rate`: The tax rate for taxable items (a float).
- `__purchases`: A `list` of `Item` objects.

The `Receipt` class should have the following methods.

- `__init__`: The constructor. Takes the tax rate as a floating-point number. If no tax rate is provided at instantiation, then the tax rate should default to 7%.

- `receiptToString`: returns the full receipt as a string.

  The string must have the following contents:

  - Each item listed with its price.
  - The total cost of the items.
  - The total tax charged on all items.
  - The grand total with tax added.
  - The current date when the receipt we generated.
  - All values must be shown to two decimal places.

  Hint: Format the numbers before you try to align them. Use two columns of 20 characters to make the lines look nice.

- `addItem`: Takes an `Item` object as a parameter and adds it to the `Receipt` object (e.g. to the `__purchases` data member).

Once you have implemented the `Receipt` class, go to the bottom of the file. You will find some hardcoded tests for the class. Run **receipt.py** in Thonny and see if these tests pass. If so, this is a good point to submit your code for up to 30 more points! You will get partial lab credit for completing this part, even if you do not complete the rest of the lab.

Milestone 3: The Main Script

Open the file named **main.py** that has the main script (we have provided a template for you). Your main script should create an empty `Receipt` object, then asks the users for `Items`. For each `Item` object read in the name, price, and if the item is taxable. When the user says that do not want to add any more items, print out the full receipt.

Make sure you use good style (good variable names, appropriate commenting, and a header comment)

Once you reach this point you are ready to make your final submission for up to 30 more points.

Example Execution Trace

```
Welcome to Receipt Creator
```

```
Enter Item name: Hot Dog
Enter Item Price: 5.15
Is the item taxable (yes/no): no
Add another item (yes/no): yes

Enter Item name: Soda
Enter Item Price: 2.50
Is the item taxable (yes/no): yes
Add another item (yes/no): yes

Enter Item name: Pretzel
Enter Item Price: 0.50
Is the item taxable (yes/no): no
Add another item (yes/no): yes

Enter Item name: Candy Bar
Enter Item Price: 1.25
Is the item taxable (yes/no): yes
Add another item (yes/no): no

----- Receipt 2023-01-02 16:21:49.515170 -----
Hot Dog_____________________________5.15
Soda________________________________2.50
Pretzel_____________________________0.50
Candy Bar___________________________1.25

Sub Total___________________________9.40
Tax_________________________________0.26
Total_______________________________9.66
```

Scoring

The score for the assignment is determined as follows:

- 10 points – `Item` Class Constructor
- 10 points – `Item` Class `itemToString` Method
- 10 points – `Item` Class `getPrice` Method
- 10 points – `Item` Class `getTax` Method
- 10 points – `Receipt` Class Constructor
- 10 points – `Receipt` Class `receiptToString` Method
- 10 points – `Receipt` Class `addItem` Method
- 15 points – Completed Program (main script)

- 15 points – Compliance and style: program meets the lab requirements, good variable names are used, the program is appropriately commented (there is at the very least a header comment in each file).

Note: Please make sure you put write both lab partners' names and userIDs (in the form abc123) in the header comment of the file you submit for this assignment.