PROG12583: Assignment 1 (IPO)

Part A (70 %)

Create a program for a retailer that creates an invoice after a dialog with a customer.

Your program should:

1. Greet the user, tell them what store they are at, and ask for their name, phone number, and postal code.

2. Present the user with a list of 5 different items for sale in your store one at a time, For each item tell them the price and ask them how many they want (they can say 0 if they don't want any).

3. Ask them for the amount of the discount they get (from 0% to 100%). We'll just assume they will be honest about this :-)

4. Then present the user with a receipt. The receipt should have the following information on it:
   a. The name, phone number, and postal code of the customer
   b. The name, address, and phone number of the store.
   c. A line for each item with the name of the item, the quantity the customer wants, the price of each item, and the total price. All prices must be rounded to two decimal places.
   d. A subtotal that adds up all the prices.
   e. The amount of HST in dollars (HST is currently 13%).
   f. The total with HST on a new line.
   g. The amount of the discount in dollars.
   h. The final price.

Structuring Your Program

1. Your code should have an clear input section, a processing section and output section. You should use comments to clearly separate each phase of the program.

2. You should structure your code so that you could change the names of the products, the prices of the products and the value of the HST easily. The best way to do this is to create variables for each of these pieces of information and assign values to them at the top of your program (e.g. product1, price1, product2, price2, etc.). Then whenever you need to reference those quantities, you can just use the variable name…

```
print("How many",product2,"do you want? They are
$",price2,"each. ",end="")
```

Sample Run

```
Hello and welcome to the online fruit stand. Please
tell us your name, phone number, and postal code
  Sara Taylor
 905-555-1234
 L8M 1P9
 First we have Apples for $0.50.
 How many would you like?  23
 First we have Oranges for $0.65.
 How many would you like?  47
 First we have Bananas for $0.23.
 How many would you like?  0
 First we have Grapefruit for $0.99.
 How many would you like?  124
 First we have Dragon Fruit for $1.99.
 How many would you like?  10
 What is your discount? (0 - 100 percent) 15

 =================================================================
 |        The Fruit Stand   | Customer:              Sara Taylor |
 | www.davesfruitstand.com |                       905-555-1234 |
 |                          |                           L8M 1P9 |
 |===============================================================  |
 |                   PRODUCT | QUANTITY | UNIT PRICE | TOTAL PRICE|
 |                    Apples |    23    |       0.50 |      11.50 |
 |                   Oranges |    47    |       0.65 |      30.55 |
 |                   Bananas |     0    |       0.23 |       0.00 |
 |                Grapefruit |   124    |       0.99 |     122.76 |
 |              Dragon Fruit |    10    |       1.99 |      19.90 |
 |---------------------------------------------------|-----------|
 |                                       Sub Total 1 |     184.71 |
 |                                             H.S.T |      24.01 |
 |                                       Sub Total 2 |     208.72 |
 |                                    Discount (10%) |      20.87 |
 |                                        Amount Due |     187.85 |
 |===============================================================  |
```

Formatting Your Output

Getting everything to line up properly as shown above can be tricky. There is a function built into Python called format which can help you out here if you use it cleverly.

The format function requires two arguments. The first is the value to format, and the second is the "format specification string" which tells the function how you want it formatted. The result will be a string.

String Formatting

```
prod1 = "GrapeFruit"
format(prod1,'40s')
→ 'GrapeFruit                              '
```

(`string` has been padded with spaces to 40 characters)

```
format(prod1,'>20s')
→ '          GrapeFruit'
```

(`string` has been padded on the left with spaces to 20 characters)

Floating Point Formatting

```
price1 = 1.90
format(price1,'.2f')
→ '1.90'
```

(`price` has been rounded to 2 decimal places for display)

```
format(price1,'10.3f')
→ '     1.900'
```

(`price` has been rounded to 3 decimal places and padded on the left to 10 characters)

quantity1 = 124

Integer Formatting

```
format(quantity,'10d')
→ '       124'
```

(`quantity` has been padded on the left to 10 characters)

Other Format Specification Strings

https://docs.python.org/release/3.2.5/library/string.html#formatspec

Commenting Standards

Comments: You must use `#` comments to mark the Input, Processing and Output sections of your program. Use a # comment to give a brief explanation whenever it might not be completely clear to a readerwhat you are doing or why.

Names: Every variable name should be descriptive and meaningful (for example use names like `customerAge` or `circleRadius` rather than `a` or `b`). By convention, variable names should start with a lower-case letter. If you have multi-word names you can use camel case (e.g. `myNewVariable`) or underscores (e.g. `my_new_variable`).

Literals vs. "Constants": Limit the use of literals that might need to be changed later. Use variables declared at the top of the program to store values like this (the value of the HST, the names and prices of the products, etc.). Doing this will make your code more maintainable and more readable. Some languages support a special type of variable called a "constant" for this purpose. A constant is a named value that can't be changed. Python doesn't support constants so we just use regular variables and then try to remember not to change them.

Naming Constants: You can use the convention of `ALL_CAPS` for constant names (e.g. `HST`, `PRICE_1`, etc.) if you like. It's a common naming convention that signals to the reader of the code that this variable should be assigned only once and then never again. It's up to you whether or not you decide to use this convention.

Part B (10%)

Analyze the source code. Divide the program into keywords, declarations, statements and blocks.

1. Number of **keywords** used and actual keywords. Research the keyword using your textbooks.
2. Number of **symbols** used and actual symbols. Research the keyword using your textbooks.
3. Number of predefined Python program elements and the actual elements themselves. Identify what kind of element it is (e.g. variable, function/method, class, and module).

Part C (20%)

Attempt to cause syntax, runtime and logical / semantic errors.

1. Change code in the source file with the intent of causing errors. **Explain / show what you have changed**. Note and identify any changes that happen right away in VSCode editor.
2. After each change, attempt to run the program and note the error produced if any.
3. Each time, **record / show the resulting error** displayed by the Python interpreter such that it is clear what change caused what error.
4. Identify the type of error produced and explain why you think it is of that type.
5. Attempt as many different errors as possible. The minimum number of errors is 5. It should also be at least one error for each error type, syntax, runtime and logical.

You are welcome to document the changes and the errors they produced using screenshots as long as the screenshots are inserted in the Word document.

Evaluation

Your program will be evaluated for performance, structure, and internal documentation. See the rubric on SLATE for more information.

Handing In

Part A – Submit the python code as Ass1<Your Stud-id>.py file and copy it into a word document and submit the word document with name Ass1_<your stud-id>.doc

Part B – In the same work document label as Part B and submit the answers.

Part C - In the same work document label as Part C and submit the answers/screenshots.