IT209 – LAB ASSIGNMENT - 4 (LA4)
Cart CLASS DEFINITION

In this assignment you are to create a class called **Cart** that will model the operation of an on-line store shopping cart. A **Cart** object, once created, will hold items selected by an on-line shopper. When the shopper is finished, the **Cart** object will be used to perform a checkout for the shopper.

The **Cart** class is to be created with the following methods and attributes:

Method Name	Purpose	Input	Output/returns
__init__	Initializer - sets the customer name for the cart and maintains a cart sequence number. The first cart is #1, the second is #2, etc.	The name of the shopper or customer (type string)	One Cart class object
add_item	If the object passed is not type Item, return False and 'not Item type' If the item is not in the Cart, add the object and the quantity of 1, return True and 'item added'. If the item is already in the Cart, add 1 to the quantity, return True and 'item added'	One item object	True plus 'item added' or False plus 'not Item type'
checkout	Print a heading line for the receipt, including the customer name and cart number. Iterate through the Cart contents, print each item, the quantity, and extended price (qty * unit price). Print a total line at the end with the total price for all items in the cart.	None	Prints a checkout receipt for the Cart, including customer name, cart number, each item qty, and price, and total price

Attribute	Type	Definition (note: double underscore protection is not needed)
cust_name	str	The name of the customer, passed to the Cart initializer when the Cart object is created
seq_no	int	Sequence number for the Cart object – starts as 1 for cart #1, then 2, 3, etc. This is not an input, but is maintained by the Cart class.
contents	dict	Holds the contents of the cart. The key is the name of the Item object added, which will be supplied by the Item object (i.e. item_object.name). The corresponding value is a list with two things: (1) the quantity of that particular item in the cart (integer), and (2) the object id of the Item being added.

A simple **Item** class definition is provided with the assignment for you to use to hold items. It only has an initializer method and two attributes: name (str) and price (float). **name** is used as the dictionary key inside the **Cart**. **price** is the unit price for one of the items (i.e. 1.99 is the cost of one carton of eggs).

The most difficult part of this lab is creating and using the **self.contents** dictionary inside the **Cart** class. The initializer only creates a blank dictionary when the Cart object is created. The **add_item** and **checkout** methods need to maintain and use it.

As an example, when item objects **i1** ('milk', 3.99), **i2** ('eggs', 1.99), and **i3** ('wine', 9.50) are added, the dictionary contents will be:

self.contents = { 'milk': [1, <i1 object id>], 'eggs': [1, <i2 object id>], 'wine': [1, <i3 object id>] }

The **add_item** method checks to see if the item name is in the dictionary. If not, it adds each of the above to the dictionary.

If Item object **i4** (‘eggs’, 1.99) is then added, the dictionary becomes:

self.contents = { 'milk': [1, <i1 object id>], 'eggs': [**2**, <i2 object id>], 'wine': [1, <i3 object id>] }

In this case the **add_item** method found that 'eggs' was already a key in the dictionary and only updated the quantity by adding one. The object ID did not need to be updated since it is a duplicate of the one previously added. Note that items can only be added to the Cart one at a time. There is no provision for adding 2, 3, or more of a given item at once.

The **checkout** method prints a receipt for the cart contents. It prints a title line showing the customer name and cart number, then iterates through the cart dictionary and prints a line for each item (name, quantity, and extended price), and a total line at the bottom showing the total cost for all items in the cart. Here’s an example:

```
Checkout for Mary Smith    Cart#  1

     milk                 1      $  3.99
     eggs                 2      $  3.98
     salmon               1      $  9.50

     Total                      $  17.47
```

A test script is included with the lab that you can copy/paste into your code that creates some Cart and Item objects and tests the **add_item** and **checkout** methods. However, please be aware that the script is not an exhaustive test, so please perform some testing beyond what’s in the script. It’s an unfortunate reality of software development that testing can only expose errors and not prove the lack of errors. Even the most exhaustive testing cannot guarantee defect-free software.

This lab should require between 25 and 30 lines of Python code to complete, not including comments and the provided code (Item class and testscript).

What and where to submit:

1. Submit by uploading a single file with the Python code to Blackboard.
2. Submit the program using the file template provided with the Lab assignment specification.
3. Input a short set of comments as the first lines that identify the program, its purpose, and its author. Also include other comments where you think necessary.
4. Include a screen shot of the program’s output using, at minimum, the display output resulting from running the testscript.

How the assignment will be assessed:

The Python code will be visually inspected and executed via command line ("python <program name>.py"). The GTA will run it with the test script, and possibly some additional test cases of their own, to ensure the class is defined, the attributes initialized, and the methods work correctly. The GTA will assess each of the following and assign a point value for each.

Item	Assessment Description	Max Value
Python code	A complete executable program (without errors) is submitted, is named correctly, includes the required attributes and functions, and includes identifying comments.	2
initializer method	A correctly coded Cart class initializer method is included that creates the required attributes and successfully creates Cart objects	2
add_item method	A correctly coded Cart class **add_item** method is included that successfully adds single items, duplicate items, and rejects wrong object types submitted. The method returns the correct values.	3
checkout method	A correctly coded **checkout** method is included that displays a cart receipt conforming with the requirements described in the text above	3
Total		**10**