CSC500

Module 6 Portfolio Milestone

GIT Repository: <https://github.com/focomapper/CSC500>

Introduction:

The module 6 Portfolio Milestone assignment involves creating the shopping cart class and prompting the user to add item to purchase objects to the shopping cart via a print\_menu method in the main body of code. Methods for adding items and printing cart contents were implemented while method to remove and modify items were stubbed out for a later time. Additionally, an item to purchase attribute was added to the item to purchase class. As a problem solver, I found this milestone rewarding.

Pseudo code-

1. Import datetime library.
2. Upate ItemToPurchase class to include item description.
3. Create ShoppingCart class.
   1. Implement constructor, add\_item, get\_num\_items\_in\_cart, get\_cost\_of\_cart, print\_total and print\_descriptions methods.
4. Create and implement print\_menu method.
5. Prompt user to create shopping cart and add items to cart.
6. Print results.

Source code-

from datetime import datetime  
  
# Create Item To Purchase class  
class ItemToPurchase:  
 def \_\_init\_\_(self):  
 # Initialize variables  
 self.item\_name = "none"  
 self.item\_price = float(0)  
 self.item\_quantity = 0  
 self.item\_description = "none" # Assumed this is necessary but not in instructions  
  
 # Print item method  
 def print\_item\_cost(self):  
 total\_price = self.item\_price \* self.item\_quantity  
 print('{:^50}'.format(self.item\_name + ' ' + str(self.item\_quantity) + ' @ ${:,.2f}'.format(self.item\_price) + ' = ${:,.2f}'.format(total\_price)))  
  
  
# Create Shopping Cart class  
class ShoppingCart:  
  
 cart\_contents = []  
  
 def \_\_init\_\_(self, name='none', date="January 1, 2020"):  
 self.customer\_name = name  
 self.current\_date = date  
  
 def add\_item(self, item): # item to purchase  
 self.cart\_contents.append(item)  
  
 def remove\_item(self, item\_name): # item name  
 print('Option to remove not available, try another option')  
 # Not complete, stubbed out, needs loop to find item in cart  
 # if item not in self.cart\_contents:  
 # print("Item not found in cart")  
 # else:  
 # self.cart\_contents.remove(item)  
  
 def modify\_item(self, item): # item to purchase  
 print('Option to modify not available, try another option')  
 # Not complete,stubbed out, needs loop  
 # if item.item\_name in self.cart\_contents:  
 # business logic needed  
  
 def get\_num\_items\_in\_cart(self):  
 total\_quantity = 0  
 for item in self.cart\_contents:  
 total\_quantity += item.item\_quantity  
 return total\_quantity  
   
 def get\_cost\_of\_cart(self):  
 cost = 0  
 for item in self.cart\_contents:  
 cost += item.item\_price \* item.item\_quantity  
 return cost  
   
 def print\_total(self):  
 print('{:^50}'.format(self.customer\_name + '\'s Shopping Cart - ' + self.current\_date))  
  
 for item in self.cart\_contents:  
 item.print\_item\_cost()  
  
 print('{:^50}'.format('Number of Items: ' + str(self.get\_num\_items\_in\_cart())))  
 print('{:^50}'.format('${:,.2f}'.format(self.get\_cost\_of\_cart())))  
  
 def print\_description(self):  
 print('{:^50}'.format(self.customer\_name + '\'s Shopping Cart - ' + self.current\_date))  
 print('{:^50}'.format("Item Descriptions"))  
 if self.get\_num\_items\_in\_cart() == 0:  
 print('{:^50}'.format('No items in cart'))  
 else:  
 for item in self.cart\_contents:  
 print('{:^50}'.format(item.item\_name + ': ' + item.item\_description))  
  
  
def print\_menu(cart):  
 item = ItemToPurchase()  
 print('a - Add item to cart')  
 print('r - Remove item from cart')  
 print('c - Change item quantity')  
 print('i - Output items\' descriptions')  
 print('o - Output shopping cart')  
 print('q - Quit')  
 option = str(input('Choose an option: '))  
 if option == 'q':  
 del item  
 return option  
 elif option == 'a':  
 item.item\_name = str(input("Enter the item name: "))  
 item.item\_price = float(input("Enter the item price: "))  
 while not type(item.item\_price) is float:  
 item.item\_price = float(input("Enter the item price in dollar and cents: "))  
 item.item\_quantity = int(input("Enter the item quantity: "))  
 while not type(item.item\_quantity) is int:  
 item.item\_quantity = int(input("Enter the item quantity in whole numbers: "))  
 item.item\_description = str(input("Enter the item description: "))  
 cart.add\_item(item)  
 elif option == 'r':  
 cart.remove\_item(item.item\_name)  
 elif option == 'c':  
 cart.modify\_item(item)  
 elif option == 'i':  
 cart.print\_description()  
 elif option == 'o':  
 cart.print\_total()  
  
  
customer\_name = str(input("Enter customer name: ")).title()  
while True:  
 cart\_date = input("Enter date: month day, year (ex. January 1, 2024): ")  
 try:  
 valid\_date = datetime.strptime(cart\_date, "%B %d, %Y")  
 formatted\_date = str(valid\_date.strftime("%B %d, %Y"))  
 break  
 except ValueError:  
 print('Date is invalid, please enter valid date')  
  
customer\_cart = ShoppingCart(customer\_name, formatted\_date)  
edit\_cart = ""  
while edit\_cart != 'q':  
 edit\_cart = print\_menu(customer\_cart)  
  
customer\_cart.print\_total()  
customer\_cart.print\_description()

Screenshot of code executing-

A screenshot of a computer

Description automatically generated