# Port Folio Milestone

**Online Shopping Cart**

This application completes the Online Shopping Cart. The Online Shopping Cart classes and menu support code is separated into its own module file and imported into the main application file. This is done to facilitate the re-use of the Online Shopping Cart functionality in additional applications.

The submitted Python code demonstrates functionality by creating a ShoppingCart object and then prompting the user for the customer name and the current date which are echoed back to the user. Next, the print\_menu function from the Online Shopping Cart module is called. The user can then choose from the menu or enter a single q character to quit.

## Python Code – Main Application

#Import OnlineShoppingCart\_module, provides for items and shopping cart class support and user menu

import OnlineShoppingCart\_module as olsc

#Online Shopping Cart Main Application

if \_\_name\_\_ == "\_\_main\_\_":

#Create ShoppingCart

shoppingCart = olsc.ShoppingCart()

#Prompt user for customer information

shoppingCart.customer\_name = input('Input customer name:')

shoppingCart.current\_date = input('Input current date:')

print('Customer name: {}'.format(shoppingCart.customer\_name))

print('Today\'s date: {}'.format(shoppingCart.current\_date))

#Create ItemsToPurchase

item = olsc.ItemsToPurchase()

num\_item = 1

#Call main menu

olsc.print\_menu(shoppingCart)

## Python Code – Online Shopping Cart Module

#Define ItemsToPurchase class

class ItemsToPurchase:

item\_name = ''

item\_price = 0.0

item\_quantity = 0

def \_\_init\_\_(self,item\_name = 'none',item\_description = 'none',item\_price = 0,item\_quantity = 0):

self.item\_name = item\_name

self.item\_description = item\_description

self.item\_price = item\_price

self.item\_quantity = item\_quantity

def print\_item\_cost(self):

print('{} {} @${:.2f} = ${:.2f}'.format(self.item\_name,self.item\_quantity,self.item\_price,self.item\_price\*self.item\_quantity))

#Define ShoppingCart class

class ShoppingCart: #Parameterized constructor, which takes the customer name and date as parameters

customer\_name = 'none'

current\_date = 'January 1, 2020'

cart\_items = []

def \_\_init\_\_(self,customer\_name = 'none',current\_date = 'January 1, 2020'):

self.customer\_name = customer\_name #customer\_name (string) - Initialized in default constructor to "none"

self.current\_date = current\_date #current\_date (string) - Initialized in default constructor to "January 1, 2020"

def add\_item(self,ItemsToPurchase): #Adds an item to cart\_items list. Has parameter ItemToPurchase. Does not return anything.

self.cart\_items.append(ItemsToPurchase)

def remove\_item(self,item\_name): #Removes item from cart\_items list. Has a string (an item's name) parameter. Does not return anything.

index = 0

found = False

for cart\_item in self.cart\_items:

if item\_name == cart\_item.item\_name:

del self.cart\_items[index]

found = True

break

index += 1

if not found:

print('Item not found in cart. Nothing removed.') #If item name cannot be found, output this message: Item not found in cart. Nothing removed.

def modify\_item(self,ItemToPurchase): #Modifies an item's description, price, and/or quantity. Has parameter ItemToPurchase. Does not return anything.

index = 0

found = False

temp = ItemsToPurchase()

for cart\_item in self.cart\_items:

if ItemToPurchase.item\_name == cart\_item.item\_name: #If item can be found (by name) in cart,

if ItemToPurchase.item\_description != temp.item\_description: #check if parameter has default values for description, price, and quantity.

self.cart\_items[index].item\_description = ItemToPurchase.item\_description #If not, modify item in cart

if ItemToPurchase.item\_price != temp.item\_price: #check if parameter has default values for description, price, and quantity.

self.cart\_items[index].item\_price = ItemToPurchase.item\_price #If not, modify item in cart

if ItemToPurchase.item\_quantity != temp.item\_quantity: #check if parameter has default values for description, price, and quantity.

self.cart\_items[index].item\_quantity = ItemToPurchase.item\_quantity #If not, modify item in cart

found = True

break

index += 1

if not found: #If item cannot be found (by name) in cart

print('Item not found in cart. Nothing modified.') #output this message: Item not found in cart. Nothing modified.

def get\_num\_items\_in\_cart(self): #Returns quantity of all items in cart. Has no parameters

num\_items = 0

for cart\_item in self.cart\_items:

num\_items = num\_items + cart\_item.item\_quantity

return num\_items

def get\_cost\_of\_cart(self): #Determines and returns the total cost of items in cart. Has no parameters

total\_cost = 0

for cart\_item in self.cart\_items:

total\_cost = total\_cost + cart\_item.item\_price \* cart\_item.item\_quantity

return total\_cost

def print\_total(self): #Outputs total of objects in cart.

print('OUTPUT SHOPPING CART')

if self.get\_num\_items\_in\_cart() > 0:

print('{}\'s Shopping Cart - {}'.format(self.customer\_name,self.current\_date))

print('Number of Items: {}'.format(self.get\_num\_items\_in\_cart()))

for cart\_item in self.cart\_items:

cart\_item.print\_item\_cost()

print('Total: ${:.2f}'.format(self.get\_cost\_of\_cart()))

else:

print('SHOPPING CART IS EMPTY') #If cart is empty, output this message: SHOPPING CART IS EMPTY

def print\_descriptions(self): #Outputs each item's description

print('OUTPUT ITEMS\'S DESCRIPTIONS')

print('{}\'s Shopping Cart - {}'.format(self.customer\_name,self.current\_date))

for cart\_item in self.cart\_items:

print('{}: {}'.format(cart\_item.item\_name,cart\_item.item\_description))

#Define Shopping Cart Menu

def print\_menu(ShoppingCart):

menu\_choice = ''

while menu\_choice != 'q':

print('MENU')

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')

menu\_choice = input('Choose an option: ')

if menu\_choice == 'a':

item = ItemsToPurchase()

print('ADD ITEM TO CART')

try:

item.item\_name = input('Enter the item name:')

item.item\_description = input('Enter the item description:')

item.item\_price = float(input('Enter the item price:'))

item.item\_quantity = int(input('Enter the item quantity:'))

ShoppingCart.add\_item(item)

except:

print('Data entry error, try again.')

elif menu\_choice == 'r':

item\_to\_remove = ''

print('REMOVE ITEM FROM CART')

item\_to\_remove = input('Enter name of item to remove:')

ShoppingCart.remove\_item(item\_to\_remove)

elif menu\_choice == 'c':

item\_to\_modify = ItemsToPurchase()

print('CHANGE ITEM QUANTITY')

try:

item\_to\_modify.item\_name = input('Enter the item name:')

item\_to\_modify.item\_quantity = int(input('Enter the new quantity:'))

ShoppingCart.modify\_item(item\_to\_modify)

except:

print('Data entry error, try again.')

elif menu\_choice == 'i':

ShoppingCart.print\_descriptions()

elif menu\_choice == 'o':

ShoppingCart.print\_total()

elif menu\_choice == 'q':

pass

else:

print('Invalid menu choice, try again.')

## Output – Input Customer, add one item

Enter the customer name, “ABC Widgets,” and the date. One item (tinfoil) is added to the cart then the shopping description and total are printed out.

A screenshot of a computer

Description automatically generated

## Output – Add another item

Another item (snips) is added to the cart and then the descriptions and shopping cart total are printed out.

A screenshot of a computer

Description automatically generated

## Output – Change the quantity of an item

A backup pair of snips is needed so the quantity is updated then the descriptions and shopping cart total are printed out.

A screenshot of a computer

Description automatically generated

## Output – Add another item

The shopper needs another item (a ruler) so that is added, and the shopping cart description and total is printed out.

A screenshot of a computer

Description automatically generated

## Output – Remove item

That ruler was too expensive, so it was removed, and the shopping cart description and total is printed out.

A screenshot of a computer

Description automatically generated

## Output – Remove item

The shopper thought they had added an item (Sticky Notes), but they had not.

A screenshot of a computer

Description automatically generated

# Git-repo

<https://github.com/tlerunner/git-repo/tree/main/Module%208>