# Port Folio Milestone

**Online Shopping Cart**

**Step 4:** Build the ShoppingCart class with the following data attributes and related methods. Note: Some can be method stubs (empty methods) initially, to be completed in later steps

* Parameterized constructor, which takes the customer name and date as parameters
* Attributes
* customer\_name (string) - Initialized in default constructor to "none"
* current\_date (string) - Initialized in default constructor to "January 1, 2020"
* cart\_items (list)
* Methods
* add\_item()
  + Adds an item to cart\_items list. Has parameter ItemToPurchase. Does not return anything.
* remove\_item()
  + Removes item from cart\_items list. Has a string (an item's name) parameter. Does not return anything.
  + If item name cannot be found, output this message: Item not found in cart. Nothing removed.
* modify\_item()
  + Modifies an item's description, price, and/or quantity. Has parameter ItemToPurchase. Does not return anything.
  + If item can be found (by name) in cart, check if parameter has default values for description, price, and quantity. If not, modify item in cart.
  + If item cannot be found (by name) in cart, output this message: Item not found in cart. Nothing modified.
* get\_num\_items\_in\_cart()
  + Returns quantity of all items in cart. Has no parameters.
* get\_cost\_of\_cart()
  + Determines and returns the total cost of items in cart. Has no parameters.
* print\_total()
  + Outputs total of objects in cart.
  + If cart is empty, output this message: SHOPPING CART IS EMPTY
* print\_descriptions()
  + Outputs each item's description.

**Example of print\_total() output:**  
John Doe's Shopping Cart - February 1, 2020  
Number of Items: 8  
Nike Romaleos 2 @ $189 = $378  
Chocolate Chips 5 @ $3 = $15  
Powerbeats 2 Headphones 1 @ $128 = $128  
Total: $521

**Example of print\_descriptions() output:**  
John Doe's Shopping Cart - February 1, 2020  
Item Descriptions  
Nike Romaleos: Volt color, Weightlifting shoes  
Chocolate Chips: Semi-sweet  
Powerbeats 2 Headphones: Bluetooth headphones

**Step 5:** In the main section of your code, implement the print\_menu() function. print\_menu() has a ShoppingCart parameter and outputs a menu of options to manipulate the shopping cart. Each option is represented by a single character. Build and output the menu within the function.  
  
If an invalid character is entered, continue to prompt for a valid choice. *Hint*: Implement Quit before implementing other options. Call print\_menu() in the main() function. Continue to execute the menu until the user enters q to Quit.

**Example:**  
MENU  
a - Add item to cart  
r - Remove item from cart  
c - Change item quantity  
i - Output items' descriptions  
o - Output shopping cart  
q - Quit  
Choose an option:

**Step 6:** Implement Output shopping cart menu option. Implement Output item's description menu option.

**Example of shopping cart menu option:**  
OUTPUT SHOPPING CART  
John Doe's Shopping Cart - February 1, 2020  
Number of Items: 8  
Nike Romaleos 2 @ $189 = $378  
Chocolate Chips 5 @ $3 = $15  
Powerbeats 2 Headphones 1 @ $128 = $128  
Total: $521

**Example of item description menu option.**  
OUTPUT ITEMS' DESCRIPTIONS  
John Doe's Shopping Cart - February 1, 2020  
Item Descriptions  
Nike Romaleos: Volt color, Weightlifting shoes  
Chocolate Chips: Semi-sweet  
Powerbeats 2 Headphones: Bluetooth headphones

## Python Code

# Define ItemsToPurchase class

class ItemsToPurchase:

item\_name = ''

item\_price = 0.0

item\_quantity = 0

def \_\_init\_\_(self,item\_name = 'none',item\_description = '',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\_description,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'

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"

self.cart\_items = [] #cart\_items (list)

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

self.cart\_items.append(ItemToPurchase)

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('\nOUTPUT 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('\nOUTPUT 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))

# In the main section of your code, implement the print\_menu() function. print\_menu() has a ShoppingCart parameter and outputs a

# menu of options to manipulate the shopping cart. Each option is represented by a single character. Build and output the menu within

# the function.

def print\_menu():

print('\nMENU')

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:\n')

return menu\_choice

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

# Initialize variables to same values in assignment

ShoppingCart = ShoppingCart()

ShoppingCart.customer\_name = 'John Doe'

ShoppingCart.current\_date = 'February 1, 2020'

ItemToPurchase\_01 = ItemsToPurchase()

ItemToPurchase\_01.item\_name = 'Nike Romaleos'

ItemToPurchase\_01.item\_description = 'Volt color, Weightlifting shoes'

ItemToPurchase\_01.item\_price = 189

ItemToPurchase\_01.item\_quantity = 2

ItemToPurchase\_02 = ItemsToPurchase()

ItemToPurchase\_02.item\_name = 'Chocolate Chips'

ItemToPurchase\_02.item\_description = 'Semi-sweet'

ItemToPurchase\_02.item\_price = 3

ItemToPurchase\_02.item\_quantity = 5

ItemToPurchase\_03 = ItemsToPurchase()

ItemToPurchase\_03.item\_name = 'Powerbeats 2 Headphones'

ItemToPurchase\_03.item\_description = 'Bluetooth headphones'

ItemToPurchase\_03.item\_price = 128

ItemToPurchase\_03.item\_quantity = 1

ShoppingCart.add\_item(ItemToPurchase\_01)

ShoppingCart.add\_item(ItemToPurchase\_02)

ShoppingCart.add\_item(ItemToPurchase\_03)

# If an invalid character is entered, continue to prompt for a valid choice. Hint: Implement Quit before implementing other options.

# Call print\_menu() in the main() function. Continue to execute the menu until the user enters q to Quit.

menu\_choice = ''

while menu\_choice != 'q':

menu\_choice = print\_menu()

if menu\_choice == 'a':

print(menu\_choice)

elif menu\_choice == 'r':

print(menu\_choice)

elif menu\_choice == 'c':

print(menu\_choice)

elif menu\_choice == 'i':

ShoppingCart.print\_descriptions() #Implement Output shopping cart menu option. Implement Output item's description menu option

elif menu\_choice == 'o':

ShoppingCart.print\_total() # Implement Output shopping cart menu option

## Output

A screenshot of a computer

Description automatically generated

# Git-repo

<https://github.com/tlerunner/git-repo/tree/main/Module%206>