# Source Code

# Create the shopping\_cart class

class shopping\_cart:

def \_\_init\_\_(self):

# Initialize cart as empty list

self.cart\_items = []

self.item = self.ItemToPurchase()

self.customer\_name = "none"

self.current\_date = "January 1, 2020"

def \_\_init\_\_(self, customer\_name, current\_date):

# Initialize cart as empty list

self.cart\_items = []

self.item = self.ItemToPurchase()

self.customer\_name = customer\_name

self.current\_date = current\_date

# Step 1: Build the ItemToPurchase class with the following specifications:

# - item\_name (string)

# - item\_price (float)

# - item\_quantity (int)

# Default constructor Initializes item's

# - name = "none"

# - item's price = 0

# - item's quantity = 0

# Method

# - print\_item\_cost()

# Create the ItemToPurchase class

class ItemToPurchase:

# Default constructor no parameters per assignment prompt

def \_\_init\_\_(self):

self.item\_name = "none" # string

self.item\_price = 0.0 # float (prompt said 0, but made float)

self.item\_quantity = 0 # int

self.item\_description = "none" # string

# print total cost of line item per assignment prompt

def print\_item\_cost(self):

self.price\_times\_qty = self.item\_price \* self.item\_quantity

print("%s %d @ $%0.2f = $%0.2f" % (self.item\_name,

self.item\_quantity,

self.item\_price,

self.price\_times\_qty))

# print descriptions

def print\_item\_description(self):

self.price\_times\_qty = self.item\_price \* self.item\_quantity

print("%s: %s" % (self.item\_name, self.item\_description))

# Add an item to the cart by creating an ItemToPurchase object and appending it to the cart list

def add\_item(self, ItemToPurchase):

self.cart\_items.append(ItemToPurchase)

# Remove item method by popping cart index

def remove\_item(self, item\_name):

item\_found = False

for i in range(len(self.cart\_items)):

if item\_name in self.cart\_items[i].item\_name:

item\_found = True

print("\t%d %s removed from cart." % (self.cart\_items[i].item\_quantity, self.cart\_items[i].item\_name))

self.cart\_items.pop(i)

break

if not item\_found:

print("\tItem not found in cart. Nothing removed.")

# Modifies an item's description, price, and/or quantity. Has parameter ItemToPurchase. Does not return anything.

def modify\_item(self, ItemToPurchase):

item\_found = False

for i in range(len(self.cart\_items)):

if ItemToPurchase.item\_name in self.cart\_items[i].item\_name:

item\_found = True

print("\nItem %s found in cart...Updating..." % ItemToPurchase.item\_name)

self.cart\_items[i].print\_item\_cost()

print("\tHas been updated to:")

self.cart\_items[i].item\_quantity = ItemToPurchase.item\_quantity

self.cart\_items[i].print\_item\_cost()

break

if not item\_found:

print("\nItem not found in cart. Nothing modified.")

# Show cart contents method

def print\_cart\_contents(self, what):

print("-------------------------------------")

for x in range(len(self.cart\_items)):

print("Item %d: " % (x+1),end="")

if what == "items":

self.cart\_items[x].print\_item\_cost()

elif what == "description":

self.cart\_items[x].print\_item\_description()

print("-------------------------------------")

# Returns quantity of all items in cart. Has no parameters.

def get\_num\_items\_in\_cart(self):

return sum([item.item\_quantity for item in self.cart\_items])

# Determines and returns the total cost of items in cart. Has no parameters.

def get\_cost\_of\_cart(self):

return sum([items.price\_times\_qty for items in self.cart\_items])

# Outputs total of objects in cart.

def print\_total(self):

# Check if there are items in the cart

if 0 < len(self.cart\_items):

print("%s's Shopping Cart - %s" % (self.customer\_name, self.current\_date))

print("Number of Items: %d" % self.get\_num\_items\_in\_cart())

# print items in cart

self.print\_cart\_contents("items")

print("Total: $%.2f" % self.get\_cost\_of\_cart())

else:

print("\nSHOPPING CART IS EMPTY")

# Outputs each item's description.

def print\_descriptions(self):

# Check if there are items in the cart

if 0 < len(self.cart\_items):

print("%s's Shopping Cart - %s" % (self.customer\_name, self.current\_date))

print("Item Descriptions")

# print items in cart

self.print\_cart\_contents("description")

else:

print("\nSHOPPING CART IS EMPTY")

# Checkout method to print goodbye msg

def checkout(self):

if (self.get\_cost\_of\_cart() > 0):

print("\nThank you for shopping with us!")

else:

print("\nNothing in your cart, have a nice day!\n")

# Get user input

def print\_input():

print("\nMenu options:")

print("\ta - Add Item")

print("\tr - Remove Item")

print("\tc - Change item quantity")

print("\ti - Output items' descriptions")

print("\to - Output shopping cart")

print("\tq - Quit")

print("\t=====================================")

menu\_selection = input("\tPlease enter a menu selection and press enter: ")

# input validation

try:

# make sure inputs are from menu list

if (menu\_selection in ["a","r","c","i","o","q"]):

return menu\_selection

else:

return print\_input()

except:

return print\_input()

# Init shopping cart object

my\_cart = shopping\_cart(input("Enter customer's name:\n"), input("Enter today's date:\n"))

print("Customer's name: " + my\_cart.customer\_name)

print("Today's date: " + my\_cart.current\_date)

# Shopping loop

checkout = False # loop control variable

while not checkout:

# Present user with menu until checkout is true

user\_selection = print\_input()

# add or remove items from cart based on user selection, or checkout

match user\_selection:

case "a":

new\_item = my\_cart.ItemToPurchase()

try:

print("\nItem", len(my\_cart.cart\_items) + 1)

new\_item.item\_name = str(input("Enter the item name:\n"))

new\_item.item\_description = input("Enter the item description:\n")

new\_item.item\_price = float(input("Enter the item price:\n"))

new\_item.item\_quantity = int(input("Enter the item quantity:\n"))

print("\nAdded to cart: ",end="")

new\_item.print\_item\_cost()

my\_cart.add\_item(new\_item)

except:

print("Sorry, item name, price or quantity was not valid, nothing added.")

case "r":

print("\nREMOVE ITEM FROM CART")

# Print curret cart contents to select from

print("Cart Currently Contains:")

my\_cart.print\_cart\_contents("items")

# get user input of name of item to remove

item\_to\_remove = input("\nEnter name of item to remove: ")

my\_cart.remove\_item(item\_to\_remove)

case "c":

print("\nCHANGE ITEM QUANTITY")

# Print curret cart contents to select from

print("Cart Currently Contains:")

my\_cart.print\_cart\_contents("items")

# initialize an item to pass to modify\_item()

modified\_item = my\_cart.ItemToPurchase()

# get user input of name of item to modify

print("\nWhich item do you want to modify?")

modified\_item.item\_name = str(input("Enter the item name:\n"))

modified\_item.item\_quantity = int(input("Enter the new quantity:\n"))

# Send item update data to modify\_item to update cart

my\_cart.modify\_item(modified\_item)

case "i":

print("\nOUTPUT ITEMS' DESCRIPTIONS")

my\_cart.print\_descriptions()

case "o":

print("\nOUTPUT SHOPPING CART")

my\_cart.print\_total()

case "q":

# end shopping loop

checkout = True

# print good bye msg

my\_cart.checkout()

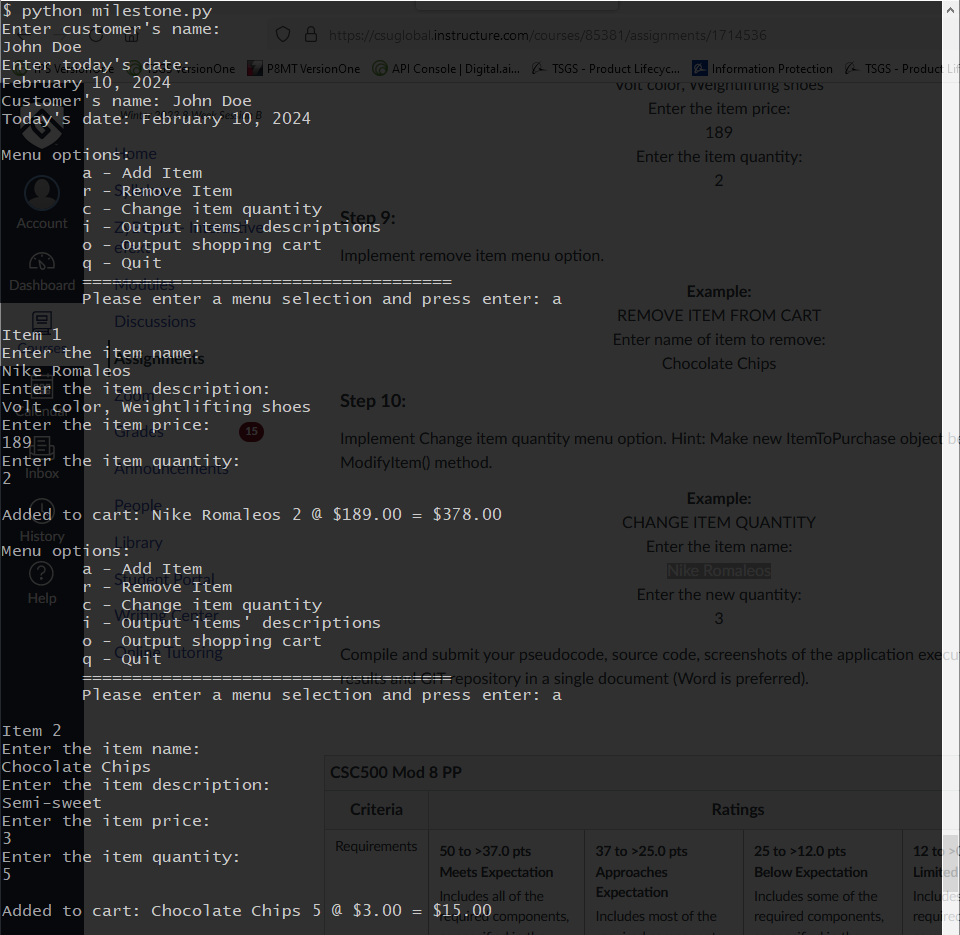
case \_:

pass

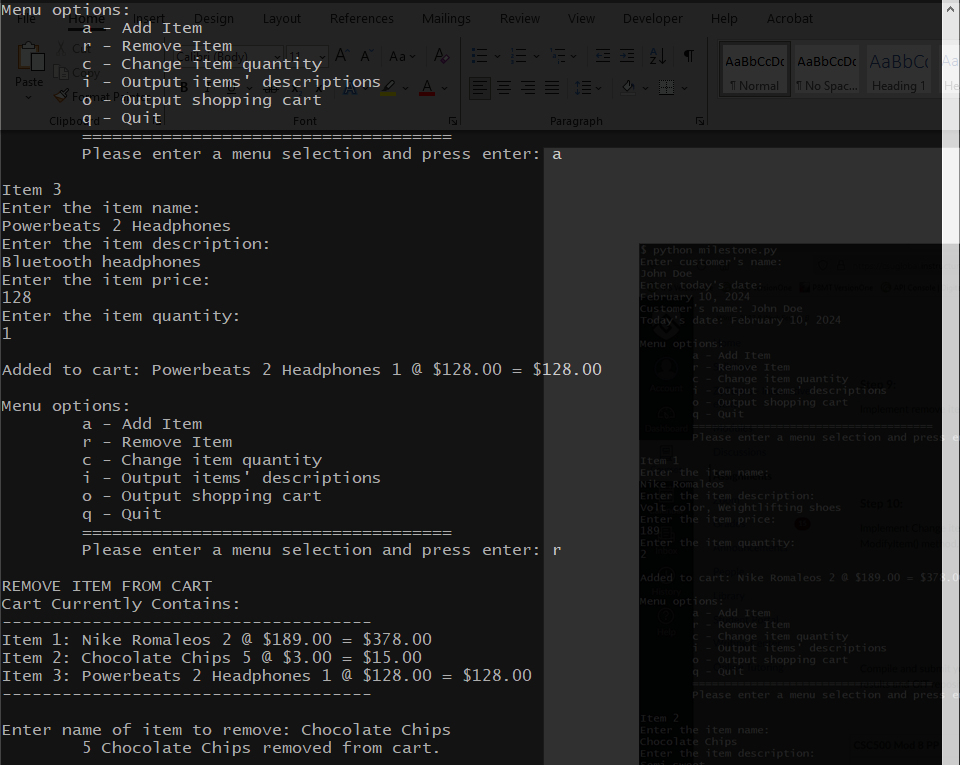
# Screenshots

## Step 7 Added user input for customer name and date and

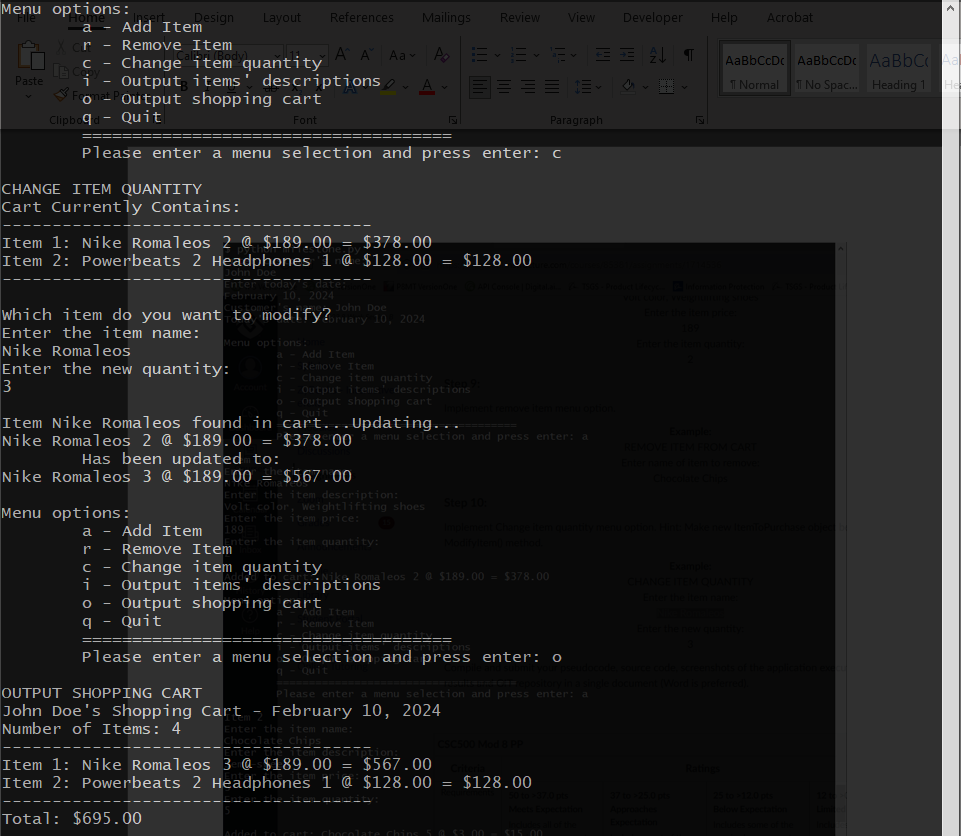
## Step 8 Demonstration of Add Item

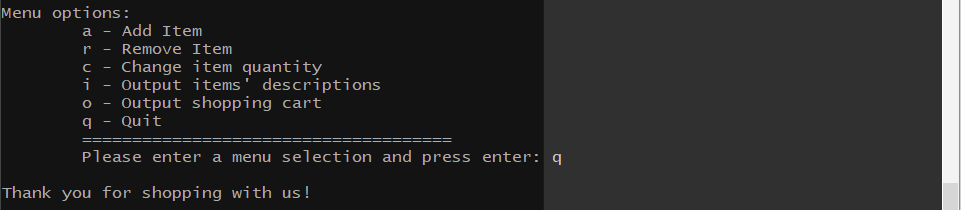


## Step 9 Demonstration of Remove Item



## Step 10 Demonstration of Change Item Quantity





# Github Repository

<https://github.com/wheyluhai/csuglobal/tree/main/CSC500-1/Module_8>