**Portfolio Milestone**

**Module 6**

Dayo Thompson

Colorado State University Global

CSC500-1

Douglas Mujeye

July 19, 2025

**Source code**

**Step 4:**

class Item:

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

self.item\_name = item\_name

self.item\_price = item\_price

self.item\_quantity = item\_quantity

self.item\_description = item\_description

@property

def item\_name(*self*):

*return* self.\_item\_name

@item\_name.setter

def item\_name(*self*, *value*):

*if* not isinstance(value, str) or value.strip() == "" or value.isdigit():

*raise* ValueError("Item name must be a non-empty string and not a number.")

self.\_item\_name = value

@property

def item\_price(*self*):

*return* self.\_item\_price

@item\_price.setter

def item\_price(*self*, *value*):

*if* not isinstance(value, (int, float)) or value < 0:

*raise* ValueError("Item price must be a non-negative number.")

self.\_item\_price = float(value)

@property

def item\_quantity(*self*):

*return* self.\_item\_quantity

@item\_quantity.setter

def item\_quantity(*self*, *value*):

*if* not isinstance(value, int) or value < 0:

*raise* ValueError("Item quantity must be a non-negative integer.")

self.\_item\_quantity = value

@property

def item\_description(*self*):

*return* self.\_item\_description

@item\_description.setter

def item\_description(*self*, *value*):

*if* not isinstance(value, str):

*raise* ValueError("Item description must be a string.")

self.\_item\_description = value

def print\_item\_cost(*self*):

total\_cost = self.item\_quantity \* self.item\_price

print(f"{self.item\_name} {self.item\_quantity} @ ${self.item\_price} = ${total\_cost:.2f}")

class ShoppingCart:

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

self.customer\_name = customer\_name

self.current\_date = current\_date

self.cart\_items = []

def add\_item(*self*, *ItemToPurchase*):

self.cart\_items.append(ItemToPurchase)

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

*for* i, item *in* enumerate(self.cart\_items):

*if* item.item\_name == item\_name:

*del* self.cart\_items[i]

*return*

print("Item not found in cart. Nothing removed.")

def modify\_item(*self*, *ItemToPurchase*):

*for* cart\_item *in* self.cart\_items:

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

*# Check if parameter has non-default values and modify accordingly*

*if* ItemToPurchase.item\_description != "":

cart\_item.item\_description = ItemToPurchase.item\_description

*if* ItemToPurchase.item\_price != 0:

cart\_item.item\_price = ItemToPurchase.item\_price

*if* ItemToPurchase.item\_quantity != 0:

cart\_item.item\_quantity = ItemToPurchase.item\_quantity

*return*

print("Item not found in cart. Nothing modified.")

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

total\_quantity = 0

*for* item *in* self.cart\_items:

total\_quantity += item.item\_quantity

*return* total\_quantity

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

total\_cost = 0

*for* item *in* self.cart\_items:

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

*return* total\_cost

def print\_total(*self*):

*if* len(self.cart\_items) == 0:

print("SHOPPING CART IS EMPTY")

*return*

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

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

*for* item *in* self.cart\_items:

item.print\_item\_cost()

print(f"\nTotal: ${self.get\_cost\_of\_cart():.2f}")

def print\_descriptions(*self*):

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

print("Item Descriptions")

*for* item *in* self.cart\_items:

print(f"{item.item\_name}: {item.item\_description}")

def main():

*# Get customer information*

customer\_name = input("Enter customer's name:\n")

current\_date = input("Enter today's date:\n")

*# Create shopping cart*

cart = ShoppingCart(customer\_name, current\_date)

*# Add multiple items to cart*

*while* True:

print("\nADD ITEM TO CART")

print("Enter the item name:")

name = input()

print("Enter the item price:")

price = float(input())

print("Enter the item quantity:")

quantity = int(input())

print("Enter the item description:")

description = input()

item = Item(name, price, quantity, description)

cart.add\_item(item)

print("Item added to cart.")

*# Ask if user wants to add another item*

add\_another = input("\nAdd another item? (y/n): ").lower().strip()

*if* add\_another != 'y':

*break*

*# Display cart total*

cart.print\_total()

*# Display item descriptions*

cart.print\_descriptions()

*# Main program*

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

main()

**Step 5:**

def print\_menu(*cart*):

"""

Outputs a menu of options to manipulate the shopping cart.

Each option is represented by a single character.

"""

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")

print("Choose an option:")

def main():

*# Get customer information*

customer\_name = input("Enter customer's name:\n")

current\_date = input("Enter today's date:\n")

*# Create shopping cart*

cart = ShoppingCart(customer\_name, current\_date)

*# Menu loop*

*while* True:

print\_menu(cart)

choice = input().lower().strip()

*if* choice == 'q':

print("Goodbye!\n")

*break*

*elif* choice == 'i':

*# Output items' descriptions*

print("OUTPUT ITEMS' DESCRIPTIONS")

cart.print\_descriptions()

*elif* choice == 'o':

*# Output shopping cart*

print("OUTPUT SHOPPING CART")

cart.print\_total()

*else*:

print("Enter the item name:")

name = input()

print("Enter the item price:")

price = float(input())

print("Enter the item quantity:")

quantity = int(input())

print("Enter the item description:")

description = input()

item = Item(name, price, quantity, description)

cart.add\_item(item)

print("Item added to cart.")

print() *# Add blank line for readability*

**Step 6**

class ItemToPurchase:

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

*self*.item\_name = *item\_name*

*self*.item\_price = *item\_price*

*self*.item\_quantity = *item\_quantity*

@property

def item\_name(*self*):

*return* *self*.\_item\_name

@item\_name.setter

def item\_name(*self*, *value*):

*if* not isinstance(*value*, str) or *value*.strip() == "" or *value*.isdigit():

*raise* ValueError("Item name must be a non-empty string and not a number.")

*self*.\_item\_name = *value*

@property

def item\_price(*self*):

*return* *self*.\_item\_price

@item\_price.setter

def item\_price(*self*, *value*):

*if* not isinstance(*value*, (int, float)) or *value* < 0:

*raise* ValueError("Item price must be a non-negative number.")

*self*.\_item\_price = float(*value*)

@property

def item\_quantity(*self*):

*return* *self*.\_item\_quantity

@item\_quantity.setter

def item\_quantity(*self*, *value*):

*if* not isinstance(*value*, int) or *value* < 0:

*raise* ValueError("Item quantity must be a non-negative integer.")

*self*.\_item\_quantity = *value*

def print\_item\_cost(*self*):

total\_cost = *self*.item\_quantity \* *self*.item\_price

print(f"{*self*.item\_name} {*self*.item\_quantity} @ ${*self*.item\_price} = ${total\_cost:.2f}")

def main():

*# Prompt user for details of the first item*

print("Item 1")

*while* True:

name1 = input("Enter the item name:\n")

*# Try to convert to integer; if successful, ask again*

*try*:

int(name1)

print("Item name cannot be an integer. Please enter a valid name.")

*except* ValueError:

*break* *# Input is not an integer, so accept it*

price1 = float(input("Enter the item price:\n"))

quantity1 = int(input("Enter the item quantity:\n"))

item1 = ItemToPurchase(name1, price1, quantity1)

*# Prompt user for details of the second item*

print("\nItem 2")

*while* True:

name2 = input("Enter the item name:\n")

*# Try to convert to integer; if successful, ask again*

*try*:

int(name2)

print("Item name cannot be an integer. Please enter a valid name.")

*except* ValueError:

*break* *# Input is not an integer, so accept it*

price2 = float(input("Enter the item price:\n"))

quantity2 = int(input("Enter the item quantity:\n"))

item2 = ItemToPurchase(name2, price2, quantity2)

*# Print the total cost summary*

print("\nTOTAL COST\n")

item1.print\_item\_cost()

item2.print\_item\_cost()

*# Calculate and print the combined total cost*

total\_cost = float(item1.item\_price) \* int(item1.item\_quantity) + float(item2.item\_price) \* int(item2.item\_quantity)

print(f"\nTotal: ${total\_cost:.2f}")

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

main()

**Complete source code**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Screenshots and code execution**

Step1: Write the command to run the python script (*python shopping\_cart.py*) and hit enter.

A screen shot of a computer program

AI-generated content may be incorrect.

**Screenshots and code execution**

Step 2: Provide the customer’s name and today’s date, and add items to the shopping cart.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**Results**

A screen shot of a shopping cart

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

**GitHub**

Link to portfolio milestone

<https://github.com/dayothompson/CSC500/tree/main/Module6>

**References**

Coddy. (2025). OOP in Python. <https://coddy.tech/courses/oop_in_python>

Zybooks. (2025).

*If-else statement.*

<https://learn.zybooks.com/zybook/CSC500-1_8/chapter/5/section/2>