**Online Shopping Cart**

Dayna Harbour

Colorado State University Global

CSC500-1: Intro to Programming

Douglas Mujeye

3 August 2025

**Online Shopping Cart**

Pseudocode:

BEGIN PROGRAM

FUNCTION: ItemToPurchase class

ATTRIBUTES:

item\_name (string)

item\_price (float)

item\_quantity (int)

item\_description (string)

METHOD: print\_item\_cost()

CALCULATE total\_cost = item\_price \* item\_quantity

PRINT item\_name, item\_quantity, item\_price, total\_cost in formatted string

METHOD: print\_item\_description()

PRINT item\_name and item\_description

FUNCTION: ShoppingCart class

ATTRIBUTES:

customer\_name (string)

current\_date (string)

cart\_items (list of ItemToPurchase objects)

METHOD: add\_item(item)

APPEND item to cart\_items list

METHOD: remove\_item(item\_name)

FOR each item in cart\_items:

IF item\_name matches:

REMOVE item

RETURN

IF no item found:

PRINT "Item not found in cart. Nothing removed."

METHOD: modify\_item(item\_to\_modify)

FOR each item in cart\_items:

IF item\_name matches:

IF item\_to\_modify has non-default values:

UPDATE item's description, price, or quantity

RETURN

IF no item found:

PRINT "Item not found in cart. Nothing modified."

METHOD: get\_num\_items\_in\_cart()

RETURN sum of item\_quantity for each item in cart\_items

METHOD: get\_cost\_of\_cart()

RETURN sum of item\_price \* item\_quantity for each item

METHOD: print\_total()

PRINT customer\_name and current\_date

IF cart is empty:

PRINT "SHOPPING CART IS EMPTY"

ELSE:

PRINT each item's cost

PRINT total cart cost

METHOD: print\_descriptions()

PRINT customer\_name and current\_date

PRINT descriptions of each item

FUNCTION: print\_menu(cart)

DEFINE menu options: add, remove, change, descriptions, cart total, quit

LOOP until user selects "q":

DISPLAY menu

GET user input

IF input is invalid:

PROMPT again

IF input is "a":

PROMPT user for item details

CREATE ItemToPurchase

CALL cart.add\_item()

IF input is "r":

PROMPT user for item name

CALL cart.remove\_item()

IF input is "c":

PROMPT user for item name and new quantity

CREATE ItemToPurchase with new quantity

CALL cart.modify\_item()

IF input is "i":

CALL cart.print\_descriptions()

IF input is "o":

CALL cart.print\_total()

MAIN FUNCTION:

PROMPT user for customer name and date

PRINT customer name and date

CREATE ShoppingCart with user input

CALL print\_menu(cart)

END PROGRAM

Python Code:

# Define the ItemToPurchase class

class ItemToPurchase:

def \_\_init\_\_(self, item\_name="none", item\_price=0.0, item\_quantity=0, item\_description="none"):

# Parameterized constructor with default values

self.item\_name = item\_name

self.item\_price = item\_price

self.item\_quantity = item\_quantity

self.item\_description = item\_description

def print\_item\_cost(self):

# Calculates and prints the total cost for this item

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

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

def print\_item\_description(self):

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

# Define the ShoppingCart class

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 = [] # This will store ItemToPurchase objects

def add\_item(self, item):

# item should be an instance of ItemToPurchase

self.cart\_items.append(item)

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

# Tries to remove item by name

found = False

for item in self.cart\_items:

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

self.cart\_items.remove(item)

found = True

break

if not found:

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

def modify\_item(self, item\_to\_modify):

# Looks for item by name and updates it if found

found = False

for item in self.cart\_items:

if item.item\_name == item\_to\_modify.item\_name:

found = True

if item\_to\_modify.item\_description != "none":

item.item\_description = item\_to\_modify.item\_description

if item\_to\_modify.item\_price != 0:

item.item\_price = item\_to\_modify.item\_price

if item\_to\_modify.item\_quantity != 0:

item.item\_quantity = item\_to\_modify.item\_quantity

break

if not found:

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\_price \* item.item\_quantity

return total\_cost

def print\_total(self):

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

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

if not self.cart\_items:

print("SHOPPING CART IS EMPTY")

else:

for item in self.cart\_items:

item.print\_item\_cost()

print(f"\nTotal: ${self.get\_cost\_of\_cart():.0f}")

def print\_descriptions(self):

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

print("Item Descriptions")

for item in self.cart\_items:

item.print\_item\_description()

def print\_menu(cart):

menu\_options = {

"a": "Add item to cart",

"r": "Remove item from cart",

"c": "Change item quantity",

"i": "Output items' descriptions",

"o": "Output shopping cart",

"q": "Quit"

}

choice = "" # initialize empty choice

while choice != "q":

print("\nMENU")

for key, desc in menu\_options.items():

print(f"{key} - {desc}")

choice = input("Choose an option:\n").lower()

while choice not in menu\_options:

choice = input("Invalid option. Choose a valid option:\n").lower()

if choice == "q":

break

elif choice == "a":

print("ADD ITEM TO CART")

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

description = input("Enter the item description:\n")

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

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

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

cart.add\_item(item)

elif choice == "r":

print("REMOVE ITEM FROM CART")

name = input("Enter name of item to remove:\n")

cart.remove\_item(name)

elif choice == "c":

print("CHANGE ITEM INFORMATION")

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

# Prompt for new values; use defaults if nothing entered

description = input("Enter the new description (or leave blank to skip):\n")

price\_input = input("Enter the new price (or leave blank to skip):\n")

quantity\_input = input("Enter the new quantity (or leave blank to skip):\n")

# Set fallback/defaults if fields are skipped

description = description if description else "none"

price = float(price\_input) if price\_input else 0.0

quantity = int(quantity\_input) if quantity\_input else 0

# Create a temp ItemToPurchase with just the new values

updated\_item = ItemToPurchase(item\_name=name, item\_price=price, item\_quantity=quantity, item\_description=description)

cart.modify\_item(updated\_item)

elif choice == "i":

print("OUTPUT ITEMS' DESCRIPTIONS")

cart.print\_descriptions()

elif choice == "o":

print("OUTPUT SHOPPING CART")

cart.print\_total()

def main():

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

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

print()

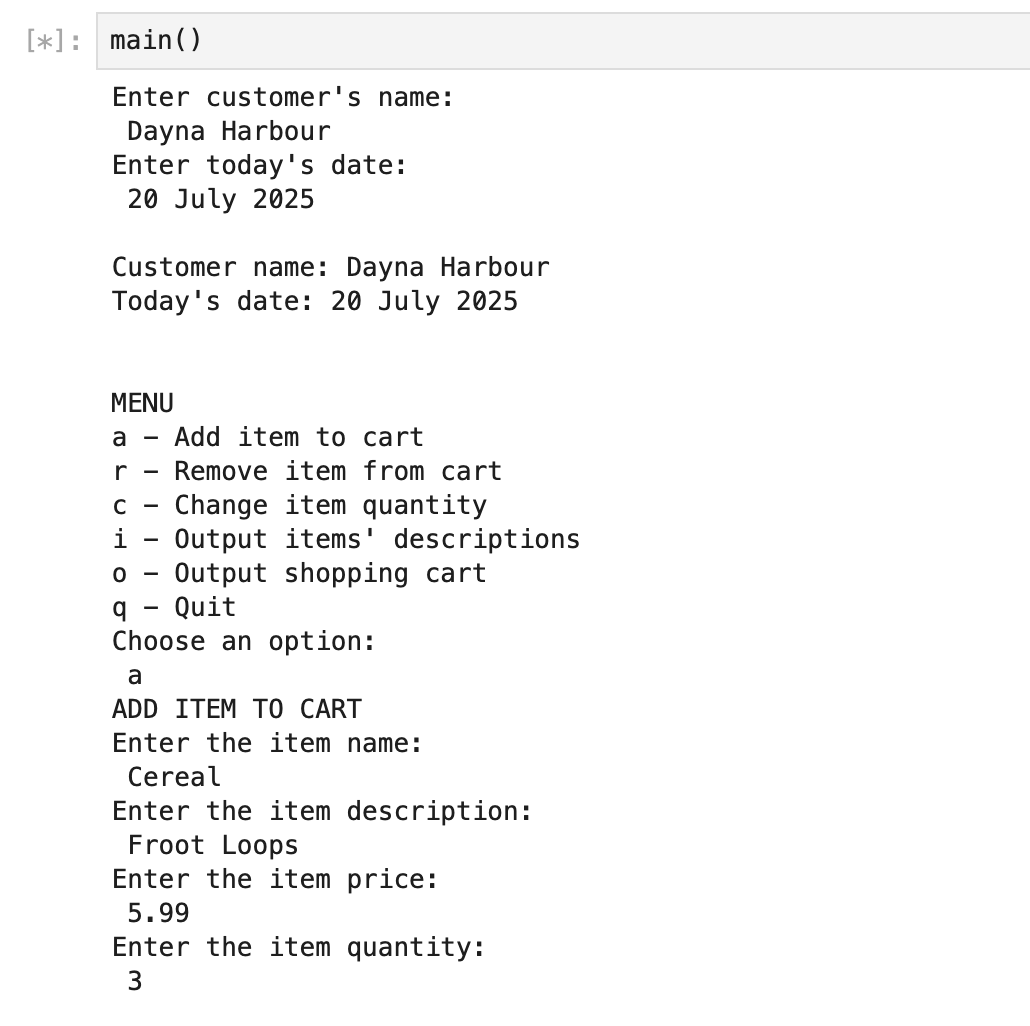
print(f"Customer name: {customer\_name}")

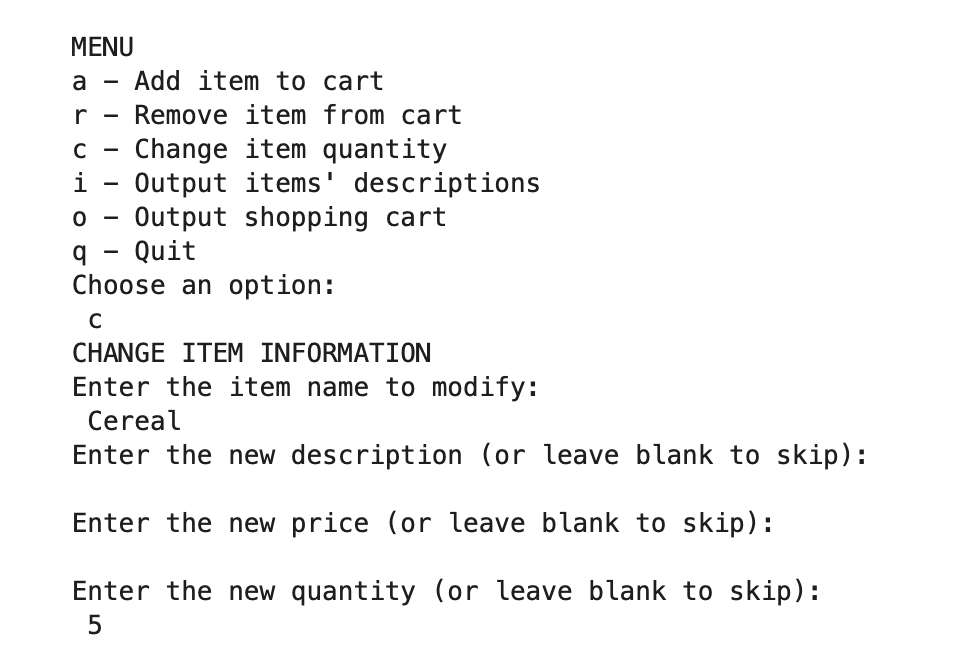
print(f"Today's date: {current\_date}\n")

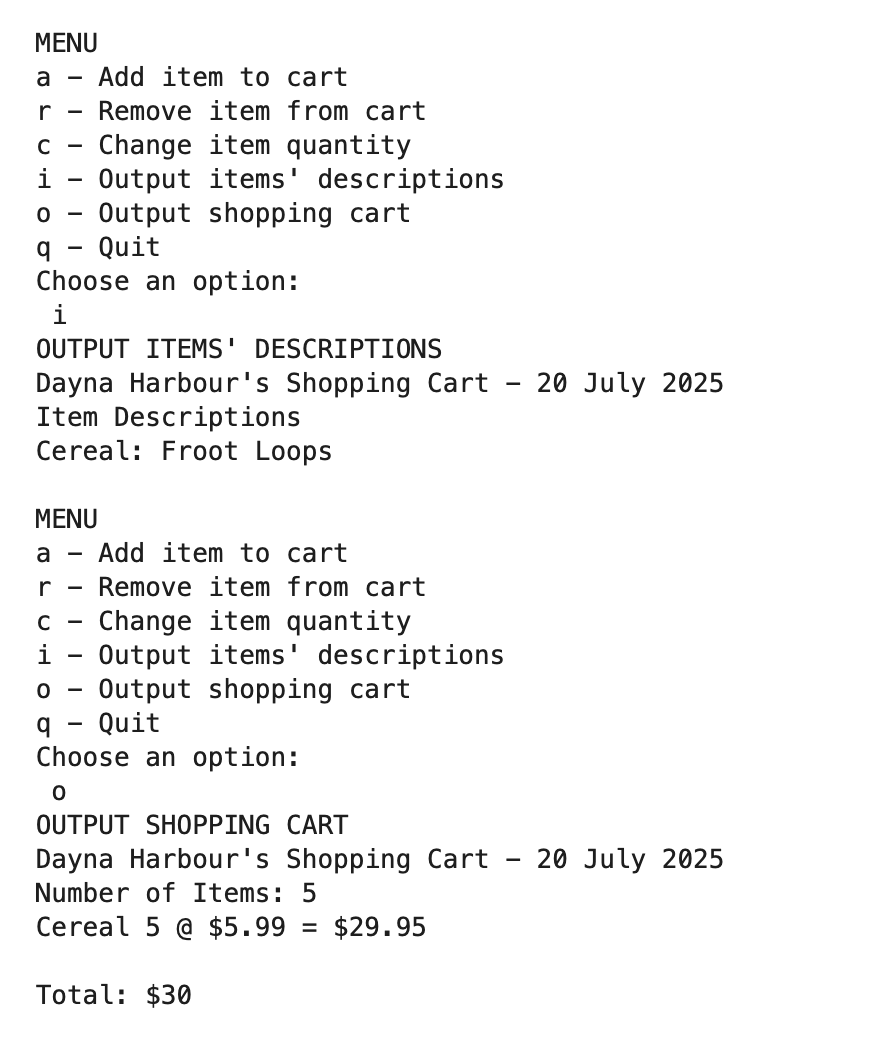
cart = ShoppingCart(customer\_name, current\_date)

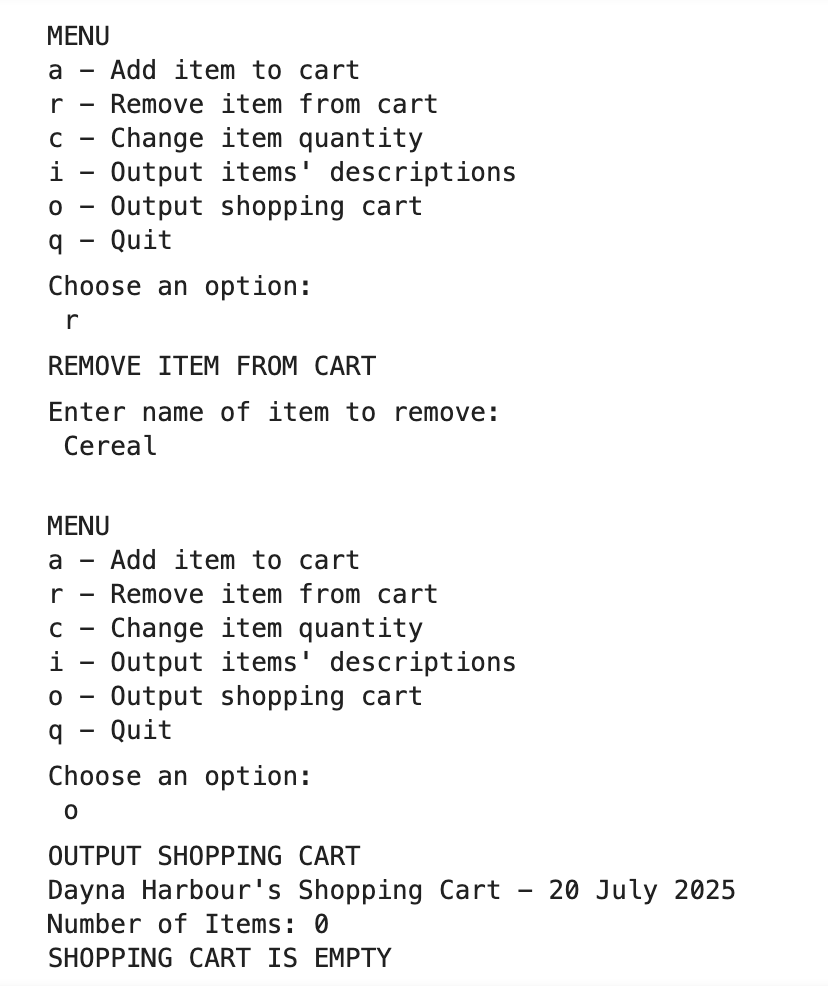
print\_menu(cart)

Screenshots:

Initiate and add item:

Change quantity:

Output of cart and items descriptions:

Remove item and print empty cart:

GIT: [https://github.com/dharbour12/IntroToProgramming/blob/main/PortfolioProjectFinal.ipynb](https://github.com/dharbour12/IntroToProgramming/blob/main/PortfolioMilestone4.ipynb)

**References**

*Python Classes and Objects [With Examples]*. (2019). Programiz.com. https://www.programiz.com/python-programming/class

W3 Schools. (2019). *Python Functions*. W3schools.com. https://www.w3schools.com/python/python\_functions.asp

‌ *zyBooks*. (2025). Zybooks.com. [https://learn.zybooks.com/zybook/CSC500-1\_8/](https://learn.zybooks.com/zybook/CSC500-1_8/chapter/6/section/12)

‌