Portfolio Project Pt II: ShoppingCart Class

# Source Code

Below is the Python source code that implements the ItemToPurchase Class and main function.

## Part 1: ItemToPurchase Class

class ItemToPurchase:

def \_\_init\_\_(self, item\_name="none", item\_price=0.0, 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

def total(self):

return self.item\_price \* self.item\_quantity

def print\_item\_cost(self):

total\_cost = self.total()

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

def print\_item\_description(self):

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

# Part 2: Main function

import re

from ItemToPurchase import ItemToPurchase

from ShoppingCart import ShoppingCart

def get\_current\_date():

date\_pattern = re.compile(r'^(January|February|March|April|May|June|July|August|September|October|November|December) \d{2}, \d{4}$')

while True:

date = input("Enter today's date (e.g., September 14, 2024): ")

if date\_pattern.match(date):

return date

else:

print("Invalid date format. Please use the format 'Month (full name) DD, YYYY'.")

def check\_value(prompt, value\_type=float):

"""

Prompts the user to enter a non-negative value.

Args:

prompt (str): The prompt message to display to the user.

value\_type (type): The type to which the user input should be cast. Default is float.

Returns:

float or int: A non-negative number entered by the user.

"""

while True:

try:

user\_val = value\_type(input(prompt))

if user\_val < 0:

print("Please enter a non-negative value.")

else:

return user\_val

except ValueError:

print(f"Invalid input. Please enter a valid {value\_type.\_\_name\_\_} larger than 0.")

def get\_customer\_name():

pattern = re.compile(r'^[a-zA-Z\s]+$')

while True:

name = input("Enter customer's name (letters only): ")

if pattern.match(name):

return name

else:

print("Invalid name. Please use letters and spaces only.")

def print\_menu(cart):

menu = (

"\nMENU\n"

"a - Add item to cart\n"

"r - Remove item from cart\n"

"c - Change item quantity\n"

"i - Output items' descriptions\n"

"o - Output shopping cart\n"

"q - Quit\n"

)

command = ""

while command != 'q':

print(menu)

command = input("Choose an option: ")

if command == 'a':

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

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

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

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

item = ItemToPurchase(item\_name, item\_price, item\_quantity, item\_description)

cart.add\_item(item)

elif command == 'r':

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

cart.remove\_item(item\_name)

elif command == 'c':

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

item\_price = check\_value("Enter the item price: ", float)

item\_quantity = check\_value("Enter the item quantity: ", int)

item = ItemToPurchase(item\_name, item\_price, item\_quantity)

cart.modify\_item(item)

elif command == 'i':

cart.print\_descriptions()

elif command == 'o':

cart.print\_total()

elif command == 'q':

break

else:

print("Invalid option. Please choose again.")

def main():

customer\_name = get\_customer\_name()

current\_date = get\_current\_date()

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

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

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

print\_menu(cart)

if \_\_name\_\_ == '\_\_main\_\_':

main()

# Part 3: Shopping Cart

from ItemToPurchase import ItemToPurchase

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, item):

self.cart\_items.append(item)

def remove\_item(self, item\_2\_find):

found = False

for item in self.cart\_items:

if item.item\_name == item\_2\_find:

self.cart\_items.remove(item)

found = True

break

if not found:

print(f"Nothing removed, {item\_2\_find} not found.")

def modify\_item(self, item):

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

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

self.cart\_items[i].item\_price = item.item\_price

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

return

print(f"{item} not found in cart.")

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

total\_quantity = sum(item.item\_quantity for item in self.cart\_items)

return total\_quantity

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

total\_cost = sum(item.total() for item in self.cart\_items)

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()}\n")

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():.2f}")

def print\_descriptions(self):

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

print("\nItem Descriptions")

if not self.cart\_items:

print("SHOPPING CART IS EMPTY")

else:

for item in self.cart\_items:

item.print\_item\_description()

current\_date = get\_current\_date()

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

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

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

print\_menu(cart)

if \_\_name\_\_ == '\_\_main\_\_':

main()

# Screenshots of Execution

## # Part 1:

A screenshot of a computer

Description automatically generatedA screenshot of a computer screen

Description automatically generated

# Results

The program was executed successfully with multiple test inputs to validate its functionality. For the item creation the attributes were correctly updated. The calculations for total cost were accurate and reflected the correct multiplication of price and quantity for each item. As displayed in the output, both the item details and the total cost were printed in a clear and organized manner.

For instance, in the test case where "Chocolate Chips" and "Bottled Water" were added to the cart, the program accurately computed the total cost as $13.00. The individual breakdowns—$3.00 for one unit of "Chocolate Chips" and $10.00 for ten units of "Bottled Water"—were both calculated and presented correctly.

These results demonstrate the program's reliability in handling inputs, performing arithmetic operations, and outputting formatted results. Thus, the program meets the requirements set forth in the project specification.

# Git Repository

Repository for this week’s assignment can be found at this [link](https://github.com/Mr-Abe/Portfolio-Project/tree/master/module_4).