Module 4 Portfolio Milestone

**Figure 1**

*Pseudocode for Shopping Cart*

START

CLASS ItemToPurchase

FUNC \_\_init\_\_(item\_name, item\_price, item\_quantity)

SET self.item to {

“name”: item\_name,

“price”: item\_price,

“quantity”: item\_quantity

FUNC print\_item\_cost()

SET cost = self.item[“price”] \* self.item[“quantity”]

PRINT “{self.item[“name”]} {self.item[“quantity”]} @ ${self.item[“price”]} =

${cost}”

FUNC main()

CREATE empty list “items”

PROMPT user for number of items “num\_items”

FOR n in range(0, num\_items)

PRINT “Item {n + 1}”

PROMPT user for item name “item\_name”

PROMPT user for item price “item\_price”

PROMPT user for item quantity “item\_quantity”

INSTANTIATE new ItemToPurchase object with item\_name, item\_price,

item\_quantity

APPEND new item to “items”

PRINT “TOTAL COST”

SET total to 0

FOR each item IN items

CALL item.print\_item\_cost()

ADD (item.item[“price”] \* item.item[“quantity”]) to “total”

PRINT “Total: $”, total

IF \_\_name\_\_==’\_\_main\_\_’

CALL main()

END

*Note. This pseudocode illustrates a shopping cart algorithm that utilizes classes to create a user desired number of objects, or items in this case, and initialize attributes such as item\_name, item\_price, and item\_quantity based on user input. The shopping cart total is then displayed to the user along with the item’s name, price, and quantity.*

**Figure 2**

*Source Code for Shopping Cart*

**class ItemToPurchase:**

**def \_\_init\_\_(**

**self,**

**item\_name: str = "none",**

**item\_price: float = 0,**

**item\_quantity: int = 0,**

**) -> None:**

**"""**

**Initializes an instance of ItemToPurchase class.**

**Args:**

**item\_name (str): The name of the item. Defaults to "none".**

**item\_price (float): The price of the item. Defaults to 0.**

**item\_quantity (int): The quantity of the item. Defaults to 0.**

**"""**

**self.item = {**

**"name": item\_name,**

**"price": item\_price,**

**"quantity": item\_quantity,**

**}**

**def print\_item\_cost(self) -> None:**

**"""**

**Prints the cost of the item.**

**"""**

**cost = self.item["price"] \* self.item["quantity"]**

**print(**

**f"{self.item['name']} {self.item['quantity']} @ ${self.item['price']:.2f} = ${cost:.2f}"**

**)**

**def main() -> None:**

**"""**

**Main function to get input for items and calculate total cost.**

**"""**

**items = []**

**num\_items = int(input("Enter the number of items: "))**

**for n in range(num\_items):**

**print(f"\nItem {n + 1}")**

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

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

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

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

**items.append(item)**

**print("\nTOTAL COST")**

**total = 0**

**for item in items:**

**item.print\_item\_cost()**

**total += item.item["price"] \* item.item["quantity"]**

**print(f"Total: ${total:.2f}")**

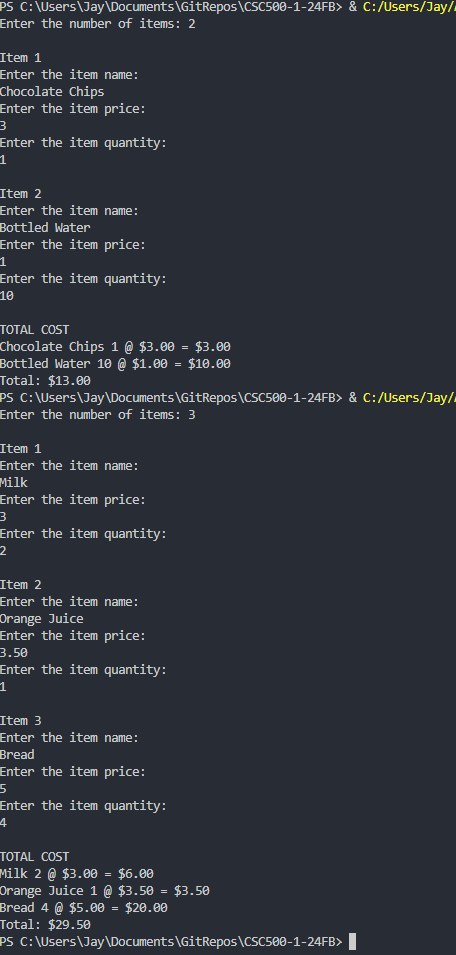
**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

*Note. This figure displays the source code used for a Python script that gets user input in order to instantiate the ItemToPurchase class as many times as the user desires. This allows the user to add items to the shopping cart to calculate a running total using loops. It calculates the total cost based on the item’s attributes, stored in a dictionary, such as item\_price and item\_quantity, and then displays the item names, quantities, prices, and total.*

**Figure 3**

*Execution and Testing for Shopping Cart*



*Note. Python output of a simple shopping cart that allows the user to provide as many items as they desire to calculate a running total. The Python script is run two times, the first being a shopping cart containing 2 items and the second shopping cart containing 3 items.*

References

Cline, J. T. [Jay4rmTheBay]. (2024). *CSC500-1-24FB* [Source code]. GitHub.<https://github.com/Jay4rmTheBay/CSC500-1-24FB>