Module 4 Portfolio Milestone

Figure 1

Pseudocode for Shopping Cart

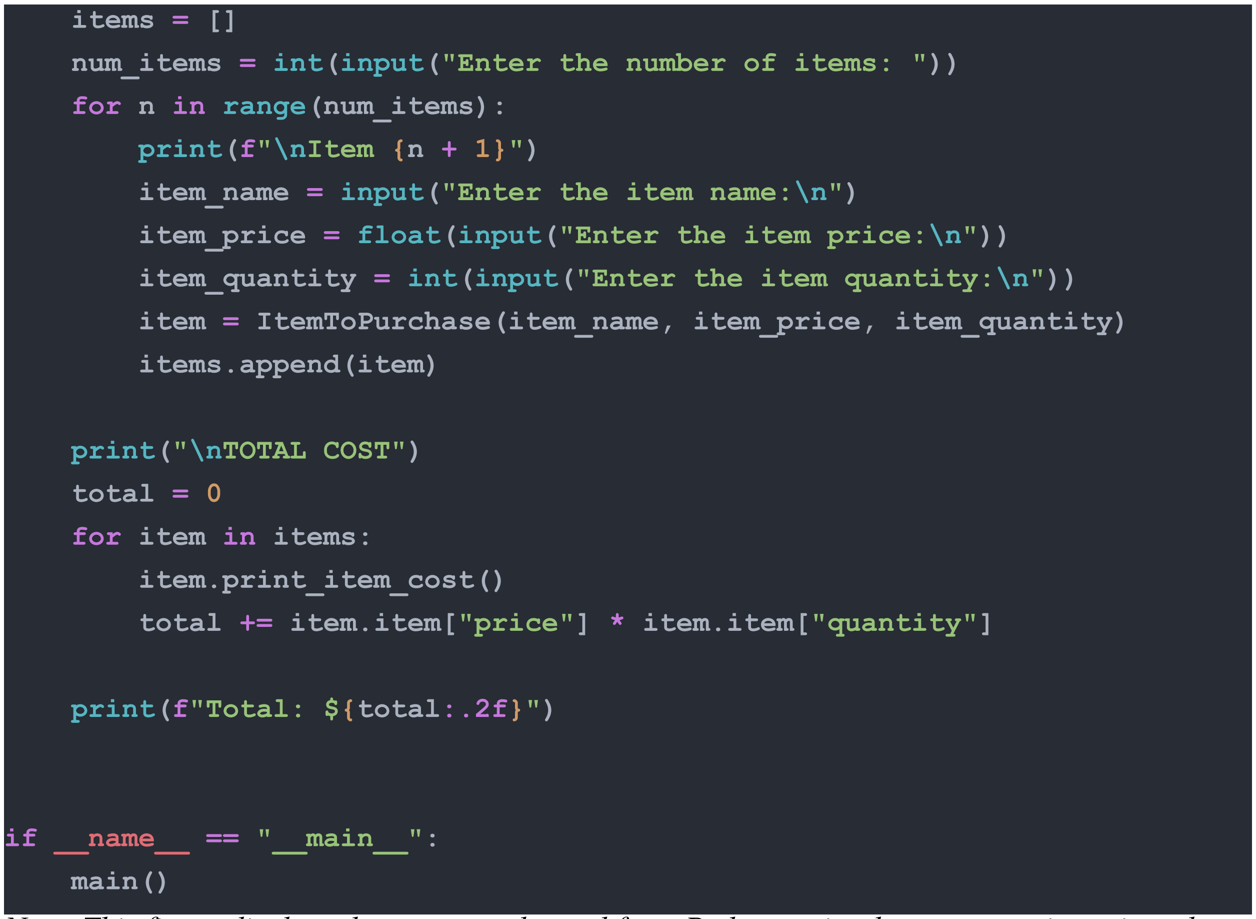
```
START
CLASS ItemToPurchase
    FUNC __init__(item_name, item_price, item_quantity)
        SET self.item to {
            “name”: item_name,
            “price”: item_price,
            “quantity”: item_quantity
    FUNC print_item_cost()
    SET cost = self.item[“price”] * self.item[“quantity”]
    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

```python
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.
    """
```

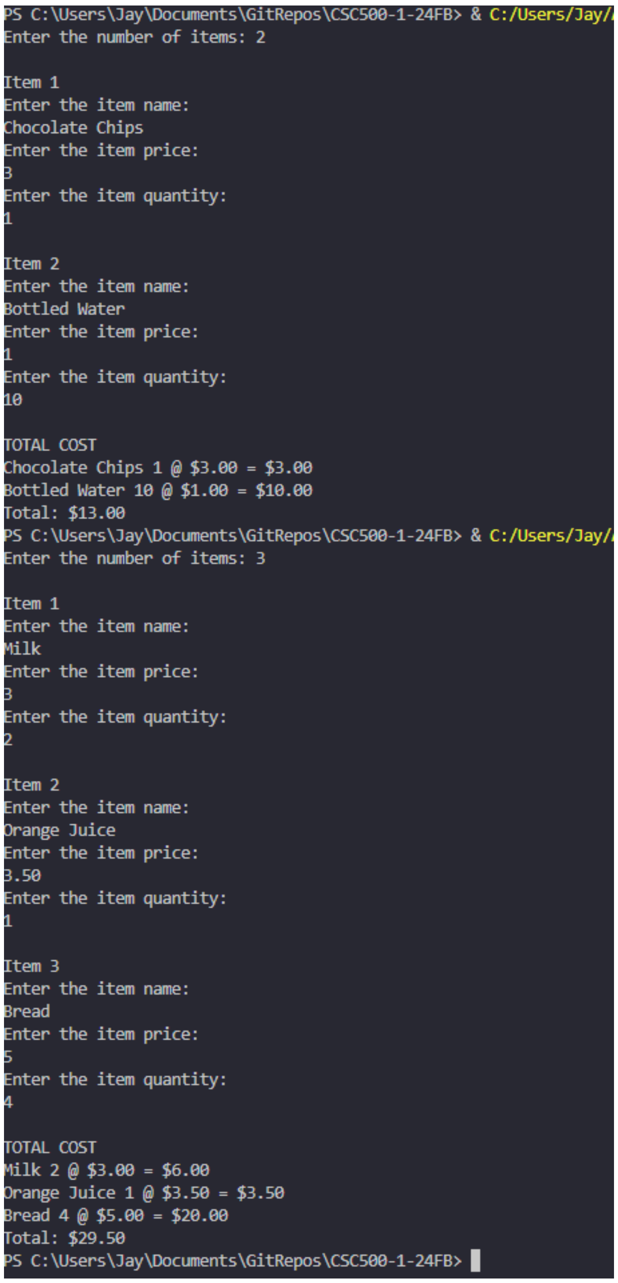


```
    items = []
    num_items = int(input("Enter the number of items: "))
    for n in range(num_items):
        print(f"\nItem {n + 1}")
        item_name = input("Enter the item name:\n")
        item_price = float(input("Enter the item price:\n"))
        item_quantity = int(input("Enter the item 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



```
PS C:\Users\Jay\Documents\GitRepos\CSC500-1-24FB> & C:/Users/Jay/
Enter the number of items: 2

Item 1
Enter the item name:
Chocolate Chips
Enter the item price:
3
Enter the item quantity:
1

Item 2
Enter the item name:
Bottled Water
Enter the item price:
1
Enter the item quantity:
10

TOTAL COST
Chocolate Chips 1 @ $3.00 = $3.00
Bottled Water 10 @ $1.00 = $10.00
Total: $13.00
PS C:\Users\Jay\Documents\GitRepos\CSC500-1-24FB> & C:/Users/Jay/
Enter the number of items: 3

Item 1
Enter the item name:
Milk
Enter the item price:
3
Enter the item quantity:
2

Item 2
Enter the item name:
Orange Juice
Enter the item price:
3.50
Enter the item quantity:
1

Item 3
Enter the item name:
Bread
Enter the item price:
5
Enter the item quantity:
4

TOTAL COST
Milk 2 @ $3.00 = $6.00
Orange Juice 1 @ $3.50 = $3.50
Bread 4 @ $5.00 = $20.00
Total: $29.50
PS C:\Users\Jay\Documents\GitRepos\CSC500-1-24FB>
```

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