###############################################################

Student: Aditya Singh Sandhu

Course: CSC 500 – Principles of Programming

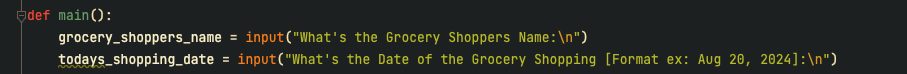
Module: 6 – Portfolio Assignment

###############################################################

Python3 Code ~

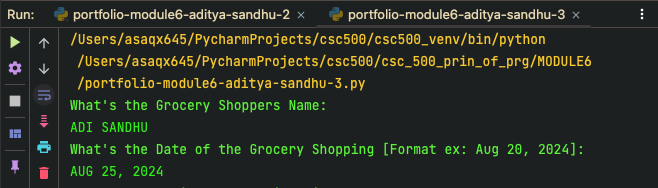
*# COPY THIS LINE OF CODE - BEGINNING*class ItemToProcure:  
 def \_\_init\_\_(self, product\_name="none", dollar\_value\_of\_product=0, numerical\_quantity\_of\_product=0):  
 self.product\_name = product\_name  
 self.dollar\_value\_of\_product = dollar\_value\_of\_product  
 self.numerical\_quantity\_of\_product = numerical\_quantity\_of\_product  
  
 def console\_log\_product\_cost(self):  
 total\_cost = self.dollar\_value\_of\_product \* self.numerical\_quantity\_of\_product  
 print(f"{self.product\_name} {self.numerical\_quantity\_of\_product} @ ${self.dollar\_value\_of\_product} = ${total\_cost}")  
  
  
def initial\_item\_to\_cart(cart):  
 print("Whats the details of the first item:")  
 item\_name = input("Whats the description of the item:\n")  
 item\_dollar\_value = float(input(f"Whats the dollar value of the {item\_name}:\n"))  
 item\_quantity = int(input(f"Whats the quantity of the {item\_name}:\n"))  
 item = ItemToProcure(item\_name, item\_dollar\_value, item\_quantity)  
 cart.add\_product(item)  
  
 print("\nHERE IS THE TOTAL COST OF THE ITEM")  
 item.console\_log\_product\_cost()  
  
 total\_cost = cart.get\_cost\_of\_cart()  
 print(f"\nTotal: ${total\_cost}")  
  
  
class ShoppingCart:  
 def \_\_init\_\_(self, grocery\_shoppers\_name="none", todays\_shopping\_date="January 1, 2020"):  
 self.grocery\_shoppers\_name = grocery\_shoppers\_name  
 self.todays\_shopping\_date = todays\_shopping\_date  
 self.cart\_items = []  
  
 def add\_product(self, item: ItemToProcure):  
 self.cart\_items.append(item)  
  
 def remove\_product(self, product\_name: str):  
 found = False  
 for item in self.cart\_items:  
 if item.product\_name == product\_name:  
 self.cart\_items.remove(item)  
 found = True  
 break  
 if not found:  
 print("The Item to be Removed, wasn't found in the cart, So nothing was REMOVED.")  
  
 def adjust\_item(self, item\_to\_modify: ItemToProcure):  
 found = False  
 for item in self.cart\_items:  
 if item.product\_name == item\_to\_modify.product\_name:  
 if item\_to\_modify.dollar\_value\_of\_product != 0:  
 item.dollar\_value\_of\_product = item\_to\_modify.dollar\_value\_of\_product  
 if item\_to\_modify.numerical\_quantity\_of\_product != 0:  
 item.numerical\_quantity\_of\_product = item\_to\_modify.numerical\_quantity\_of\_product  
 found = True  
 break  
 if not found:  
 print("The Item to be Modified, wasn't found in the cart, So nothing was MODIFIED.")  
  
 def get\_num\_items\_in\_cart(self) -> int:  
 total\_quantity = sum(item.numerical\_quantity\_of\_product for item in self.cart\_items)  
 return total\_quantity  
  
 def get\_cost\_of\_cart(self) -> float:  
 total\_cost = sum(item.dollar\_value\_of\_product \* item.numerical\_quantity\_of\_product for item in self.cart\_items)  
 return total\_cost  
  
 def print\_total(self):  
 print(f"{self.grocery\_shoppers\_name}'s Shopping Cart - {self.todays\_shopping\_date}")  
 if not self.cart\_items:  
 print("THERE IS NOTHING IN THE SHOPPING CART, IT IS EMPTY!")  
 else:  
 print(f"Total Number of Items in the Shopping Cart: {self.get\_num\_items\_in\_cart()}")  
 for item in self.cart\_items:  
 item.console\_log\_product\_cost()  
 print(f"\nTotal Cost of the Cart: ${self.get\_cost\_of\_cart()}")  
  
 def print\_descriptions(self):  
 print(f"{self.grocery\_shoppers\_name}'s Shopping Cart - {self.todays\_shopping\_date}")  
 print("Item Descriptions")  
 for item in self.cart\_items:  
 print(f"{item.product\_name}: {item.dollar\_value\_of\_product}")  
  
  
def print\_menu(cart: ShoppingCart):  
 menu = """  
 MENU  
 ADD - Add item to cart  
 REMOVE - Remove item from cart  
 MODIFY - Modify item quantity  
 IDENTIFY - Output items' descriptions  
 OUTPUT - Output shopping cart  
 QUIT - Quit  
 """  
 while True:  
 print(menu)  
 choice = input(f"{cart.grocery\_shoppers\_name}, choose an option to edit your shopping cart:\n")  
 if choice == 'QUIT':  
 break  
 elif choice == 'ADD':  
 name = input(f"{cart.grocery\_shoppers\_name}, what's the description of the item to add to the existing shopping cart:\n")  
 price = float(input(f"{cart.grocery\_shoppers\_name}, what's the price of the {name} being added to the existing shopping cart:\n"))  
 quantity = int(input(f"{cart.grocery\_shoppers\_name}, what's the quantity of the {name} being added to the existing shopping cart:\n"))  
 new\_item = ItemToProcure(name, price, quantity)  
 cart.add\_product(new\_item)  
 elif choice == 'REMOVE':  
 name = input(f"{cart.grocery\_shoppers\_name}, what's the description of the item to remove from the existing shopping cart:\n")  
 cart.remove\_product(name)  
 elif choice == 'MODIFY':  
 name = input(f"{cart.grocery\_shoppers\_name}, what's the description of the item to modify in the existing shopping cart:\n")  
 price = float(input(f"{cart.grocery\_shoppers\_name}, what's the new price of {name} (or 0 to leave unchanged):\n"))  
 quantity = int(input(f"{cart.grocery\_shoppers\_name}, what's the new quantity of {name} (or 0 to leave unchanged):\n"))  
 modified\_item = ItemToProcure(name, price, quantity)  
 cart.adjust\_item(modified\_item)  
 elif choice == 'IDENTIFY':  
 cart.print\_descriptions()  
 elif choice == 'OUTPUT':  
 cart.print\_total()  
  
  
def main():  
 grocery\_shoppers\_name = input("What's the Grocery Shoppers Name:\n")  
 todays\_shopping\_date = input("What's the Date of the Grocery Shopping [Format ex: Aug 20, 2024]:\n")  
 cart = ShoppingCart(grocery\_shoppers\_name, todays\_shopping\_date)  
 initial\_item\_to\_cart(cart) *# Only one item is initially added* print\_menu(cart) *# Proceed with the menu*if \_\_name\_\_ == "\_\_main\_\_":  
 main()  
*# COPY THIS LINE OF CODE - ENDING*

"Screenshot 1," "Main Function Line one and two execution, Input Prompt from user”

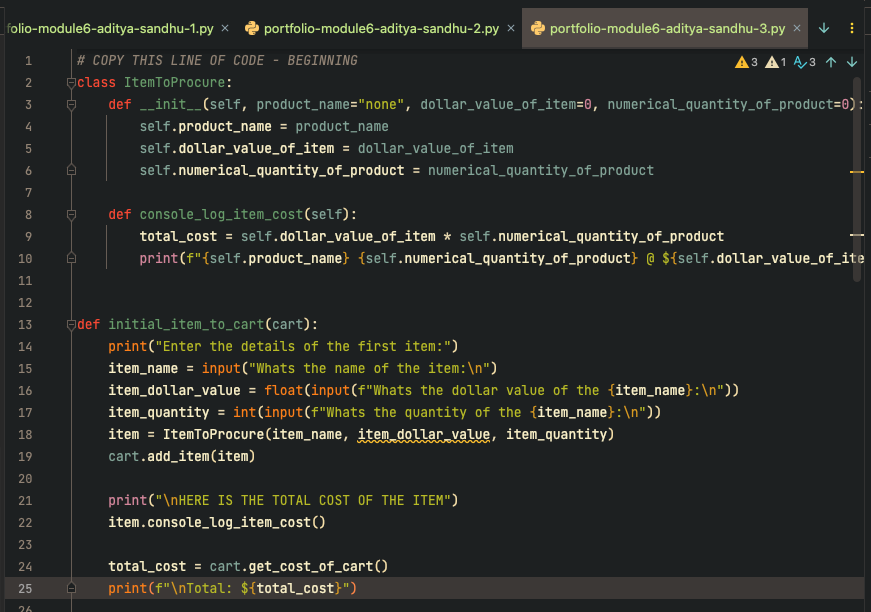


Asking for the Grocery Shoppers Name and the format of the data when the shopping is occurring

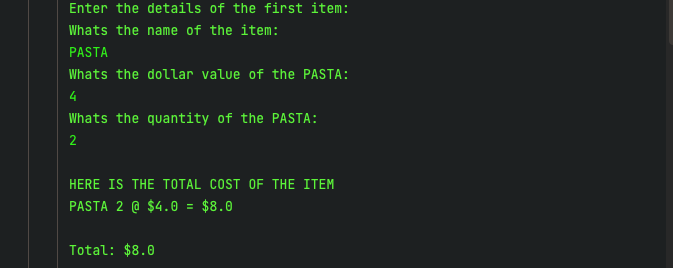
“Screenshot 2”, “Output of the Name and Date Input from Shopper”



“Screenshot 3”, “The ItemToProcure Function and its Cart Creation Logic”



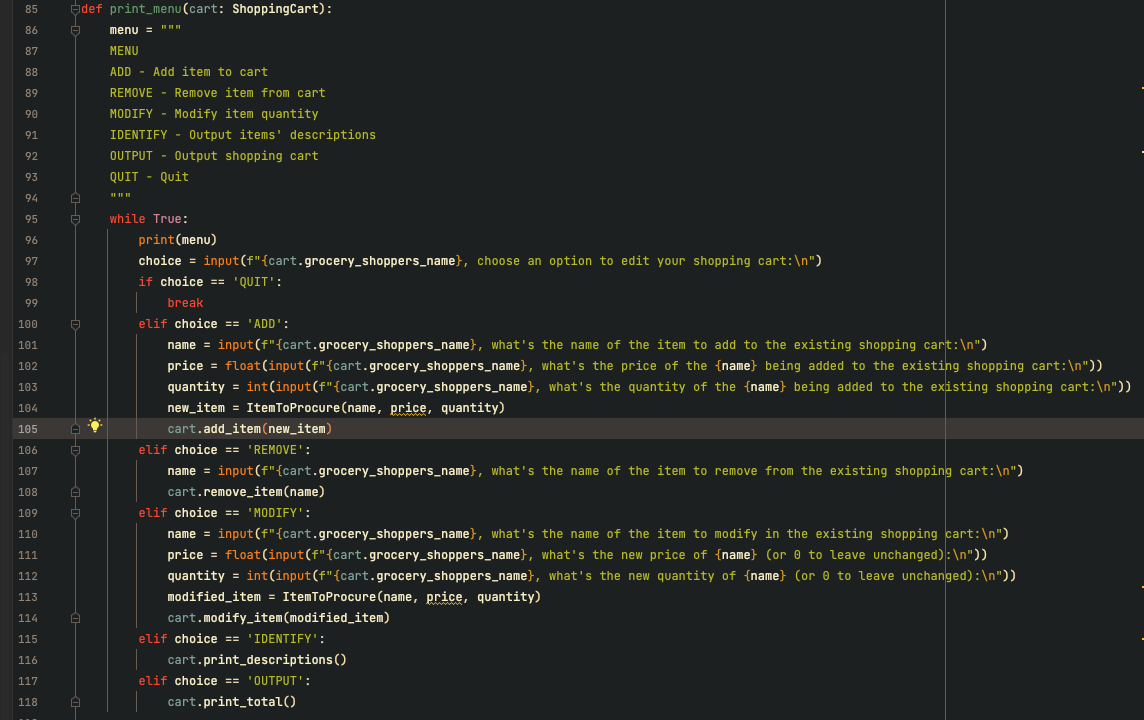
“Screenshot 4”, “Output on the Concole of the Class ItemToProcure”



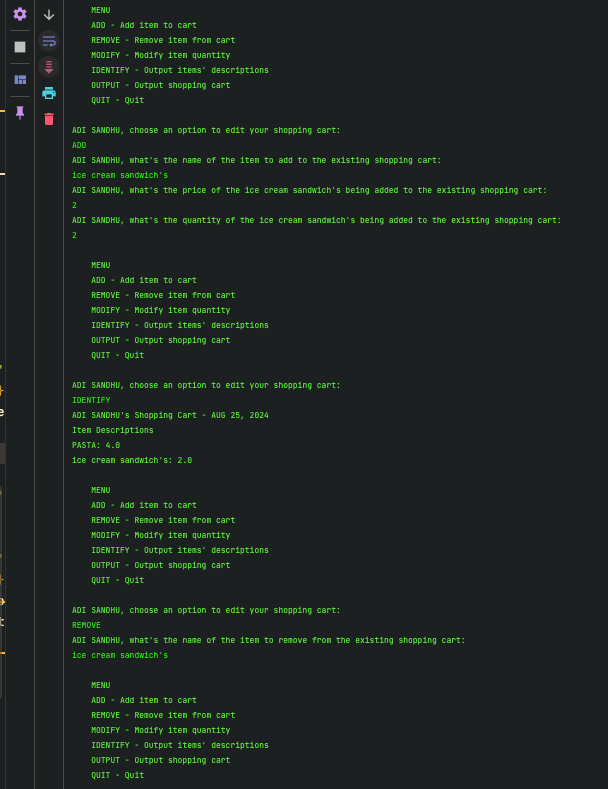
“Screenshot 5”, “The ShoppingCart Class, that has the add item, remove item, modify item, get num of items in shopping cart, cost of cart, print tota and print description functions that are passed to print menu function to display ”



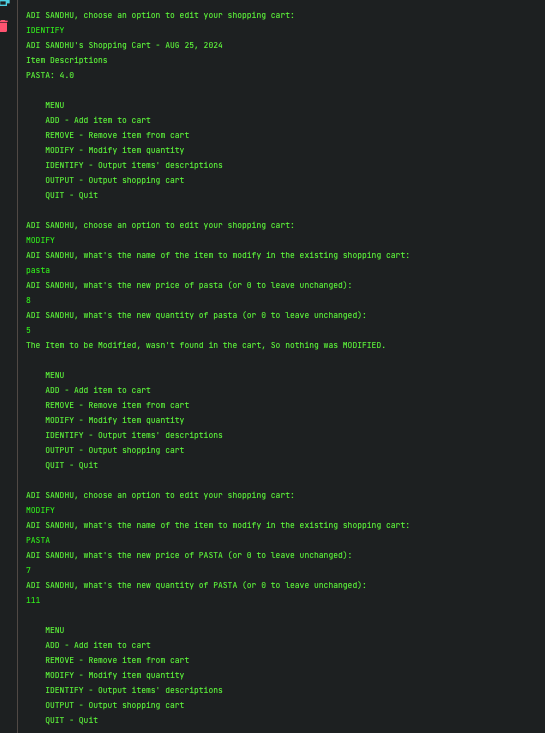
“Screenshot 6”, “The print\_menu function with the menu, and the ‘if, elif control flow that calls the add\_item function, remove\_item, modify\_item, print\_description,and print\_total functions’ from the ShoppingCart Class”



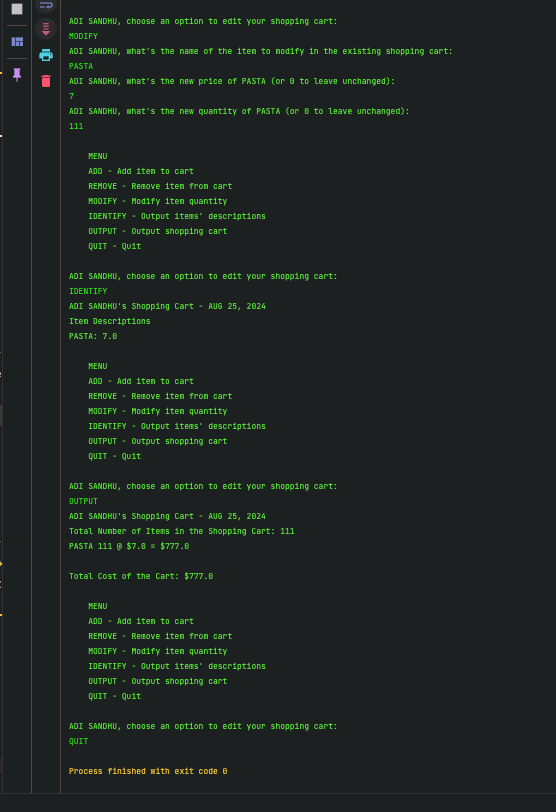
“Screenshot 7”, ”After PASTA Being Added, Menu Displayed, ADD Option Selected via Input, ‘ice cream sandwich’s added, at price 2 and quantity 2, then MENU Displayed again, and IDENTIFY Selected, and CART Displayed with Item Description Function Called. The Menu Displayed again. Next REMOVE Prompt passed to the program, and ice cream sandwich’s removed”



“Screenshot 8”, ”After ice cream sandwich’s are removed, IDENTIFY is passed as an INPUT, to MENU, and the shopping cart is displayed with ice cream sandwich’s no longer part of the cart. Then MENU Displayed again, and MODIFY is Selected, and ‘PASTA’, an object in the cart is passed as INPUT, the next prompt is to identify the new price, and quantity.”



“Screenshot 9”, ”After the MODIFY function executed, the MENU is dispayed again,and IDENTIFY reveals that the PASTA is at QUANTITY 7, the price was modified to 111 and isn’t displayed, the MENU Displays again, and OUTPUT Selection revels the new Price of PASTA and the totat cost of the cart, then QUIT exits the program”



GITHUB LINK - <https://github.com/65AR645ASAN/csc_500_prin_of_prg/blob/main/MODULE6/portfolio-assignment-module6-aditya-sandhu.docx>

**APA citations:**

**W3Schools. (n.d.). Python Conditions. Retrieved August 25, 2024, from** [**https://www.w3schools.com/python/python\_conditions.asp**](https://www.w3schools.com/python/python_conditions.asp)

**Python Software Foundation. (n.d.). The Python Tutorial: More Control Flow Tools. Retrieved August 25, 2024, from** [**https://docs.python.org/3/tutorial/controlflow.html**](https://docs.python.org/3/tutorial/controlflow.html)

**W3Schools. (n.d.). Python Classes and Objects. Retrieved August 25, 2024, from** [**https://www.w3schools.com/python/python\_classes.asp**](https://www.w3schools.com/python/python_classes.asp)

**Python Software Foundation. (n.d.). The Python Tutorial: Classes. Retrieved August 25, 2024, from** [**https://docs.python.org/3/tutorial/classes.html**](https://docs.python.org/3/tutorial/classes.html)

**GeeksforGeeks. (n.d.). Python Functions. Retrieved August 25, 2024, from** [**https://www.geeksforgeeks.org/python-functions/**](https://www.geeksforgeeks.org/python-functions/)

**Reddit. (n.d.). Pass class as argument to another class. Retrieved August 25, 2024, from** [**https://www.reddit.com/r/learnpython/comments/12fk1we/pass\_class\_as\_argument\_to\_another\_class/**](https://www.reddit.com/r/learnpython/comments/12fk1we/pass_class_as_argument_to_another_class/)

**Sikka, S. (2024, January 24). Python Type Hinting: How To Get Started and Why You Should Use It. Dagster. Retrieved August 25, 2024, from** [**https://dagster.io/blog/python-type-hinting**](https://dagster.io/blog/python-type-hinting)