**Module8 Portfolio Project Git Repo**

**From Milestone 1:**

|  |
| --- |
| <https://github.com/sankar228/csu-ms-aiml/blob/ae22ea752a8fdf6bbc686751ca1a6ae0c40f527f/CSC500/module8/melestone1.py> |

**source code**

**Problem statement:** Building the ItemToPurchase class.

|  |
| --- |
| class ItemToPurchase :      item\_name: str      item\_price: float      item\_quantity: int        def \_\_init\_\_(self, item\_name= None, item\_price= 0, item\_quantity= 0):          self.item\_name = item\_name          self.item\_price= item\_price          self.item\_quantity= item\_quantity      ## Function to print the Total cost      def print\_item\_cost(self):          print(f"{self.item\_name} {self.item\_quantity} @ ${pricestr(self.item\_price)} = ${pricestr(round(float(self.item\_quantity \* self.item\_price), 2))}")    def pricestr(v: float) -> str:      return f"{round(v , 2): .2f}"    if \_\_name\_\_ == "\_\_main\_\_":      items = []      for i in range(1, 3):          print(f"Item {i}")          item\_name = input("Enter the item name: ")          item\_price = round(float(input("Enter the item price: $")), 2)          item\_quantity = int(input("Enter the item quantity: "))            items.append(ItemToPurchase(item\_name=item\_name, item\_price=item\_price, item\_quantity=item\_quantity))      print()      print("###########")      print("TOTAL COST")      total\_cost = 0      for item in items:          total\_cost += round(float(item.item\_quantity \* item.item\_price), 2)          item.print\_item\_cost()        print(f"Total: ${pricestr(total\_cost)}") |

**Code execution:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**From Milestone 2:**

|  |
| --- |
| <https://github.com/sankar228/csu-ms-aiml/blob/ae22ea752a8fdf6bbc686751ca1a6ae0c40f527f/CSC500/module8/shoppingcart.py> |

**source code**

**Problem statement:** Building the ShoppingCart.

**Assumptions:**

1. Each Item in the cart is of type ‘ItemToPurchase’ as given in the above problem statement.
2. Cart items are stored in the list only data structure
3. Item is the key to identify the uniquenes of the items in the cart list
4. Remove the item, If the user input item quantity is more than the avialble quantity in the cart list
5. Adding an item and item already exist, prompting the user to add the extra quantity only. And that would update the item quantity and price.

**Code:**

|  |
| --- |
| ## Cart Item object  class ItemToPurchase :      item\_name: str      item\_desc: str      item\_price: float      item\_quantity: int        def \_\_init\_\_(self, item\_name= None, item\_desc=None, item\_price= 0, item\_quantity= 0):          self.item\_name = item\_name          self.item\_desc = item\_desc          self.item\_price= item\_price          self.item\_quantity= item\_quantity      ## Function to print the Total cost      def print\_item\_cost(self):          print(f"{self.item\_name} {self.item\_quantity} @ ${pricestr(self.item\_price)} = ${pricestr(round(float(self.item\_quantity \* self.item\_price), 2))}")      ## Shopping Cart object, contains utility functions to dd, remove, modify and print the Cart items  class ShoppingCart:        def \_\_init\_\_(self, customer\_name= None, current\_date= "January 1, 2020", cart\_items= []):          self.customer\_name = customer\_name          self.current\_date = current\_date          self.cart\_items = cart\_items        # Add item to the cart      def add\_item(self, item: ItemToPurchase):          self.cart\_items.append(item)          print(f"item: {item.item\_name} added to the cart")        # Remove item from the Cart      def remove\_item(self, item\_name: str):          for item in self.cart\_items:              if (item.item\_name == item\_name):                    ## Logic to ask the user to input how many quantity of the item to remove from the cart                  ## if the input quantity is more than available in the cart, it will remove the item but no error                  remove\_quantity = int(input(f"Cart item {item.item\_name} has {item.item\_quantity} quantity, How many you want to remove: "))                  if(not validate\_item\_data(item\_quantity=remove\_quantity)):                      return                  if(remove\_quantity < item.item\_quantity):                      item.item\_quantity = item.item\_quantity - remove\_quantity                      self.modify\_item(item)                      return                  if(remove\_quantity >= item.item\_quantity):                      self.cart\_items.remove(item)                      print(f"item: {item\_name} removed from the cart\n\n")                      return            print(f"Item: {item\_name} not found in cart. Nothing removed.\n\n")        # Modify the existing item from the Cart      def modify\_item(self, updated\_item: ItemToPurchase):          for item in self.cart\_items:              if (item.item\_name == updated\_item.item\_name):                  is\_updated = False                  # Update the item details                  if(updated\_item.item\_desc != None):                      item.item\_desc = updated\_item.item\_desc                      is\_updated = True                  if (updated\_item.item\_price != None):                      item.item\_price = updated\_item.item\_price                      is\_updated= True                  if (updated\_item.item\_quantity != None):                      item.item\_quantity = updated\_item.item\_quantity                      is\_updated = True                    if(not is\_updated):                      print(f"No updates available for the item: {updated\_item.item\_name}\n\n")                    print(f"item: {updated\_item.item\_name} modified\n\n")                  return            print(f"Item:{updated\_item.item\_name} not found in cart. Nothing modified.\n\n")        # Get total number of items from the Cart      def get\_num\_items\_in\_cart(self) -> int:          return len(self.cart\_items)        # Total cost of the itesm in the Cart      def get\_cost\_of\_cart(self) -> float:          if (self.get\_num\_items\_in\_cart() == 0):              print("SHOPPING CART IS EMPTY")              return 0          else:              total\_price = 0              for item in self.cart\_items:                  total\_price += (item.item\_quantity \* item.item\_price)                return round(total\_price, 2)        # Search for item in the cargt      def is\_item\_exist(self, itemname) -> bool:          for item in self.cart\_items:              if (item.item\_name == itemname):                  return True          return False        # Show the Cart item details      def print\_total(self):          print("OUTPUT SHOPPING CART")          print(f"{self.customer\_name}'s Shopping Cart - {self.current\_date}")          print(f"Number of Items: {len(self.cart\_items)}")            for item in self.cart\_items:              print(f"{item.item\_name} {item.item\_quantity} @ ${round(item.item\_price, 2): .2f} = ${round(item.item\_quantity \* item.item\_price, 2): .2f}")            print(f"Total: ${self.get\_cost\_of\_cart(): .2f}\n\n")        # Show the Cart item and its description      def print\_descriptions(self):          print("OUTPUT ITEMS' DESCRIPTIONS")          print(f"{self.customer\_name}'s Shopping Cart - {self.current\_date}")          if (self.get\_num\_items\_in\_cart() == 0):              print("SHOPPING CART IS EMPTY")              return            print("Item Descriptions")          for item in self.cart\_items:              print(f"{item.item\_name}: {item.item\_desc}")          print("\n\n")  ### Main Execution starts here  # Format output float values  def pricestr(self, v: float) -> str:      return f"{round(v , 2): .2f}"    # validate input values  def validate\_item\_data(item\_price= None, item\_quantity= None) -> bool:      is\_valid\_date = True      if (item\_price != None and item\_price < 0):          print(f"please enter valid price")          is\_valid\_date = False      if (item\_quantity != None and item\_quantity <= 0):          print(f"please enter valid quantity")          is\_valid\_date = False        return is\_valid\_date    # Print Menu  def menu\_items():      print("MENU")      print("a - Add item to cart")      print("r - Remove item from cart")      print("c - Change item description, price, and/or quantity")      print("i - Output items' descriptions")      print("o - Output shopping cart")      print("q - Quit")      print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n")  # Print the Menu and promt user to input the operation to perform, like add, remove, modify, print the Cart details  # optionally we can input 'q' to quit the execution  def print\_menu():      customer\_name = input("Enter customer's name: ").strip()      current\_date = input("Enter today's date (eg: February 1, 2020): ").strip()      cart = ShoppingCart(customer\_name=customer\_name, current\_date=current\_date)      while True:          print("\n")          menu\_items()          menu\_option = input("Choose an option from Menu: ").strip()          if(menu\_option == "q"):              exit(1)          elif(menu\_option == "a"):              print("ADD ITEM TO CART")              item\_name = input("Enter item name: ").strip()                # item name is mandatory field in the cart              if(item\_name == None or item\_name == ""):                  print("Item name can not be null")                  continue                # Check if the item already exist in the cart              if(cart.is\_item\_exist(itemname=item\_name)):                  item\_quantity = int(input(f"Item {item\_name} already exist in the cart, please enter the extra quantity required: "))                  # validate the price and quantity values                  if(not validate\_item\_data(item\_price=item\_price, item\_quantity= item\_quantity)):                      continue                  item.item\_quantity += item\_quantity                else:                  item\_desc = input("Enter item description: ").strip() or None                    item\_price = input("Enter the item price: $").strip()                  item\_price = round(float(item\_price), 2) if item\_price else None                    item\_quantity= input("Enter the item quantity: ").strip()                  item\_quantity = int(item\_quantity)                    # validate the price and quantity values                  if(not validate\_item\_data(item\_price=item\_price, item\_quantity= item\_quantity)):                      continue                    item = ItemToPurchase(item\_name=item\_name, item\_desc=item\_desc, item\_price=item\_price, item\_quantity=item\_quantity)                  cart.add\_item(item)            elif(menu\_option == "r"):              print("REMOVE ITEM FROM CART")              if (cart.get\_num\_items\_in\_cart() == 0):                  print("SHOPPING CART IS EMPTY")                  continue                item\_name = input("Enter item name to remove: ").strip()              cart.remove\_item(item\_name)          elif(menu\_option == "c"):              print("CHANGE ITEM QUANTITY")              if (cart.get\_num\_items\_in\_cart() == 0):                  print("SHOPPING CART IS EMPTY")                  continue                item\_name = input("Enter the item name: ").strip()                if(cart.is\_item\_exist(itemname=item\_name)):                  item\_quantity= input("Enter the new quantity: ").strip()                  item\_quantity = int(item\_quantity) if item\_quantity else None                    updated\_item = ItemToPurchase(item\_name=item\_name, item\_quantity=item\_quantity)                  cart.modify\_item(updated\_item)              else:                  print(f"item: {item\_name} not found in cart. Nothing modified.")          elif(menu\_option == "i"):              cart.print\_descriptions()          elif(menu\_option == "o"):              cart.print\_total()          else:              print("plase enter valid choice:")      ## Main Function call  if \_\_name\_\_ == "\_\_main\_\_":      print\_menu() |

**Code execution:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**