# Summary

**Step 1:** Build the ItemToPurchase class

* For the default constructor instead of 0 for price, I initialized price to 0.0 since it’s a float based on attribute definition
* I also wrapped ItemToPurchase class inside a shopping\_cart class to more easily manage custom methods for cart behavior

**Step 2:** In the main section of your code, prompt the user for two items and create two objects of the ItemToPurchase class.

* Instead of just 2 items, the program lets the user continue shopping until they checkout

**Step 3:** Add the costs of the two items together and output the total cost.

* This part is shown after the user chooses (3) from the menu to checkout.

# Source Code

# Create the shopping\_cart class

class shopping\_cart:

def \_\_init\_\_(self):

# Initialize cart as empty list

self.cart = []

self.item = self.ItemToPurchase()

# Step 1: Build the ItemToPurchase class with the following specifications:

# - item\_name (string)

# - item\_price (float)

# - item\_quantity (int)

# Default constructor Initializes item's

# - name = "none"

# - item's price = 0

# - item's quantity = 0

# Method

# - print\_item\_cost()

# Create the ItemToPurchase class

class ItemToPurchase:

# Default constructor no parameters per assignment prompt

def \_\_init\_\_(self):

self.item\_name = "none" # string

self.item\_price = 0.0 # float (prompt said 0, but made float)

self.item\_quantity = 0 # int

# print total cost of line item per assignment prompt

def print\_item\_cost(self):

self.price\_times\_qty = self.item\_price \* self.item\_quantity

print("%s %d @ $%0.2f = $%0.2f" % (self.item\_name,

self.item\_quantity,

self.item\_price,

self.price\_times\_qty))

# Add an item to the cart by creating an ItemToPurchase object and appending it to the cart list

def add\_item(self):

self.cart.append(self.ItemToPurchase())

# Skipping input validation for simplicity

# If user enters a string (non number) into item price or quantity the program will crash

print("\nItem", len(self.cart))

self.cart[-1].item\_name = str(input("Enter the item name:\n"))

self.cart[-1].item\_price = float(input("Enter the item price:\n"))

self.cart[-1].item\_quantity = int(input("Enter the item quantity:\n"))

print("Added to cart: ",end="")

self.cart[-1].print\_item\_cost()

# Remove item method by popping cart index

def remove\_item(self):

self.show\_cart()

item\_to\_remove = int(input("\nWhich item do you want to remove? (Enter Item number): ")) - 1

if item\_to\_remove in range(len(self.cart)):

print("\t%d %s removed from cart." % (self.cart[item\_to\_remove].item\_quantity, self.cart[item\_to\_remove].item\_name))

self.cart.pop(item\_to\_remove)

else:

print("\tItem number not valid.")

return self.remove\_item()

# Show cart contents method

def show\_cart(self):

print("\nCart Contains:")

print("\t-------------------------------------")

for x in range(len(self.cart)):

print("\tItem %d: " % (x+1),end="")

self.cart[x].print\_item\_cost()

print("\t-------------------------------------")

# Calculate total cost of shopping cart method

def get\_cart\_total(self):

return sum([items.price\_times\_qty for items in self.cart])

# Checkout method

def checkout(self):

self.total = self.get\_cart\_total()

# Step 3: Add the costs of the two items together and output the total cost

if (self.total > 0):

print("\nTOTAL COST")

print("-------------------------------------")

for items in self.cart:

items.print\_item\_cost()

print("-------------------------------------")

print("Total: $%.2f\n" % self.total)

print("Thank you for shopping with us!")

else:

print("\nNothing in your cart, have a nice day!")

# Get user input

def get\_input():

print("\nMenu options:")

print("\t1. Add Item")

print("\t2. Remove Item")

print("\t3. Checkout")

print("\t=====================================")

menu\_selection = input("\tPlease enter a number (1-3) and press enter: ")

# input validation

try:

menu\_selection = int(menu\_selection)

if (1 <= menu\_selection and menu\_selection <= 3):

return menu\_selection

else:

return get\_input()

except:

return get\_input()

# Init shopping cart object

my\_cart = shopping\_cart()

# Shopping loop

checkout = False # loop control variable

while not checkout:

user\_selection = get\_input()

# add or remove items from cart based on user selection, or checkout

match user\_selection:

case 1:

my\_cart.add\_item()

case 2:

my\_cart.remove\_item()

case 3:

checkout = True

my\_cart.checkout()

case \_:

pass

# Screenshot of Code Running

A screenshot of a computer screen

Description automatically generated