Lab 8 - Thanksgiving Specials for Shopping Cart

ShopMart wants you to create a Thanksgiving Special display.

In a loop, display 5 Valentine's Day products, ask the user to choose a product by number 1, 2, 3, 4 or 5.

The customer can add a product to her shopping cart. The cart show how many of each product the customer currently has on order.

The customer can do any of these actions:
Shop, **Checkout**, or **Cancel**

Shop
User sees product list, chooses which to add to her cart and sees the updated contents of her cart.

Checkout
Request payment for the total value of the cart.

Cancel
Thank the user for browsing at ShopMart, no money owed.

Products

The 5 products and their prices:

```
ShopMart's Thanksgiving Specials:
# Item ..... Price
1 Turkey .... 26.95
2 Cranberry..  3.95
3 Pumpkin Pie  8.95
4 Spicy Latte  4.95
5 Yams.......  1.99

Product number: 1
Shopping cart contents:
Qty  Item ..... Amount Subtotal
1    Turkey.....  26.95    26.95
Total ..................   26.95

Action: Shop, Checkout, Cancel
```

Later, after some more shopping

```
Shopping cart contents:

Qty  Item ..... Amount Subtotal
1    Turkey.....  26.95    26.95
2    Spicy Latte   4.95     8.90
Total ..................   36.85

Action: Shop, Checkout, Cancel
```

At checkout, we'll ask the customer to pay 36.85.

Start simply to tackle a big problem.

Don't tackle a big problem all at once.

`Pick out a simple piece and work on it.`

Suggestion:
Create a `while` loop that asks the user to choose and action (Shop, Checkout, or Cancel)
Get the user input, print a simple message ("You are shopping") and request another input.

Test your simple solution. Does it work when you start by typing 'Shop'?

A possible simple next step:
Assume that **each product sells for the same price**, such as $9. Can you get a running total of the "items" the user has requested?

Make sure you can show the **total due** when the user presses Enter to end the loop.

If your user selected 3 products, you should show a total due of $27, assuming a $9 each price.

Another refinement:
Display a list of the 5 special products. Use a list for the products. You might also want a heading "ShopMart Thanksgiving Specials" or the like.

Yet another:
When the user types 1 to select product 1, display something like "You ordered a Turkey".

Now tackle the real prices. You'll need a prices list, kept in a parallel structure to the products list.

Note: when you list the **products and prices, they will not line up** in a really pretty fashion. Let that issue go for now.

Check that you are adding the price correctly. Are you getting the right total when a user chooses just 1 item? How about 2 items? How about 2 of the same kind (we are quite happy at ShopMart to sell you as many Lattes or Pumpkin Pies as you want).

Customer's Shopping Cart

Next, represent the contents of the customer's shopping cart.
You could keep a list of all the product numbers the customer selects.
Another strategy: keep a list of the quantities of each product in the shopping cart. You can keep this list in parallel with the products and prices lists.

Next, display the current contents of the customer's shopping cart. Then ask for another products, cancel the order or checkout.

At checkout, make sure you do not insist that the user pay $0 if they did not order anything, or if they canceled their order.

Getting the products and prices to line up is a good idea, but takes extra work, so it's an Extra Credit option.

Grading rubric 40 points total

15 points	Basic simple "no frills" interaction loop Asks customer to choose an action (Shop, Checkout, Cancel)
5 points	Display list of products and prices
5 points	Shows total $ purchased so far.
10 points	Shows customer shopping cart contents.
5 point	Shows total owed at end of shopping.

You should do the work in stages, but you only need to submit 1 file, showing all the features you have working correctly.

Extra credit

+5 Display products using the same width for various products

+5 Display nicely formatted numbers `8.90` not `8.9`
Instead of

```
Qty Item  Price Subtotal
1 Turkey 26.95 26.95
2 Spicy Latte 4.95 8.9
```

Do something like this

```
Qty Item.............    Price Subtotal
1   Turkey...........    26.95    26.95
2   Spicy Latte......     4.95     8.90
```

Some example code:

Start with several lists
One for products; another for prices.

Get a main loop going, something that asks for the action to take (Shop, Checkout, Cancel).

Stay in the loop as long as the action is Shop.

In the loop, display the products on special. Ask which one the user would like to add to their shopping cart. Then show the contents of the shopping cart.

The key is to not get too detailed too quickly. Make sure the main loop works; for example, that it is not a never-ending (infinite) loop.

Functions to display the shopping cart, and display the contents of the shopping cart will help. Define "stubs" - a very simple version of the functions.

```
def display_products(products, prices):
    '''Display a list of products and prices '''
    print("ShopMart's Thanksgiving Specials:")
    print("STUB: display products and prices here")
# end def

def display_cart_qty(cartQty, products, prices):
    '''cartQty has quantities for each product
       This displays the products, prices and quantities
       in the shopping cart
    '''
    print()
    print("Shopping cart contents:")
    print("STUB Display cart contents somehow here")
    print()
# end def
```

And here are some lists we will need:

```
# Lists of products, prices:
products = ["Turkey", "Cranberry", "Pumpkin Pie", "Spicy Latte", "Yams"]
prices   = [26.95,          3.95,          8.95,          4.99,    1.99]
cartQty  = [     0,             0,             0,             0,       0]

# Track total amount owed for shopping cart contents
total_owed = 0.00
```

You might choose a different way to represent the contents of the shopping cart.

Here is some main code - a loop that runs as long as you are shopping.

```
# main loop
action = input("Action: Shop, Cancel, Checkout? ")  # get user request

while action == "Shop":
    display_products(products, prices)

    choice = input("Product number: ")
    p = int(choice)
    item = products[p]
    price = prices[p]
    print("You added", item, price, "to your shopping cart")
    total_owed = total_owed + price

    cartQty[p] = cartQty[p] + 1
    display_cart_qty(cartQty, products, prices)

    action = input("Action: Shop, Cancel, Checkout? ") # get user request
# end while loop

if action == "Checkout":
    print()
    print("Checking out, please pay for your shop cart contents")
    print()
    print("Please pay ", total_owed)
    print()
    print("Thank you for shopping at ShopMart")
else:
    print("Thanks for visiting ShopMart")
    total_owed = 0.00
    print("You cancelled your order, so you owe us nothing.")
# end if
print ("ShopMart thanks you for shopping with us")
```

Here's the result you would get from something like this code:

```
ShopMart's Thanksgiving Specials:
STUB: display products and prices here

Product number: 4
You added Yams 1.99 to your shopping cart

Shopping cart contents:
STUB Display cart contents somehow here

Action: Shop, Cancel, Checkout? Shop

ShopMart's Thanksgiving Specials:
STUB: display products and prices here

Product number: 2
You added Pumpkin Pie 8.95 to your shopping cart

Shopping cart contents:
STUB Display cart contents somehow here

Action: Shop, Cancel, Checkout? Checkout
```

Here's a small change in a function to get a bit more data to display:

```
def display_products(products, prices):
    ''' (list, list) -> None
        Display a list of products and prices
    '''
    print()
    print("ShopMart's Thanksgiving Specials:")
    print("STUB: products, prices")
    print(products)
    print(prices)
    print()
# end def

def display_cart_qty(cartQty, products, prices):
    ''' (list, list, list) -> None
        Display the shopping cart's products, prices and quantities
    '''
    print()
    print("Shopping cart contents:")
    print("STUB Display cart contents somehow here")
    print(cartQty)
    print()
# end def
```

This display has a little more substance, but is far from complete:

```
Welcome to ShopMart's exclusive Thanksgiving specials

Action: Shop, Cancel, Checkout? Shop

ShopMart's Thanksgiving Specials:
STUB: products, prices
['Turkey', 'Cranberry', 'Pumpkin Pie', 'Spicy Latte', 'Yams']
[26.95, 3.95, 8.95, 4.99, 1.99]

Product number: 2
You added Pumpkin Pie 8.95 to your shopping cart

Shopping cart contents:
STUB Display cart contents somehow here
[0, 0, 1, 0, 0]
```

Here is some improvements to the display functions for better looking output:

```
def display_products(products, prices):
    ''' (list, list) -> None
        Print a list of products and prices
    '''
    print()
    print("ShopMart's Thanksgiving Specials:")
    print("  # Description          Price")
    for i in range(len(products)):
        price_show = round(prices[i], 2)
        print(i, products[i], price_show)
    # end for
    print()
# end def
```

This code does a better job of displaying the data in a more readable format.

Things don't line up, and we have an item # zero, but it's better.

with this kind of printed result:

```
Action: Shop, Cancel, Checkout? Shop

ShopMart's Thanksgiving Specials:
  # Description          Price
0 Turkey 26.95
1 Cranberry 3.95
2 Pumpkin Pie 8.95
3 Spicy Latte 4.95
4 Yams 1.99
```

Putting the products into a fixed=length string can help line up the data better.

```
# File: ShopMarShopCanceCheckout3.py
#
def fixed_width(item, width, pad_char):
    ''' (str, int, str) -> str
        returns item in a fixed width string ending in pad_char's
        Example
        >>>fixed_width('dog', 8, '.')
        dog.....
    '''
    n = len(item)
    pad_len = width - n
    fixed_width_item = item + (pad_len * pad_char)
    return fixed_width_item
# end def

def display_products(products, prices):
    ''' (list, list) -> None
        Print a list of products and prices
    '''
    print()
    print("ShopMart's Thanksgiving Specials:")
    desc = fixed_width("Description", 16, '.')
    print("  # " + desc + "      Price")
    for i in range(len(products)):
        i_show = format(i + 1, "3d")
        price_show = format(prices[i], "10,.2f")
        print(i_show, fixed_width(products[i], 16, '.'), price_show)
    # end for
    print()
# end def
```

Lining up your data takes effort, and the fixed_width function, or some similar Python code, so it counts as Extra Credit.
The results:

```
Action: Shop, Cancel, Checkout? Shop

ShopMart's Thanksgiving Specials:
  # Description.....      Price
  1 Turkey..........      26.95
  2 Cranberry.......       3.95
  3 Pumpkin Pie.....       8.95
  4 Spicy Latte.....       4.95
  5 Yams............       1.99

Product number: 3
You added Pumpkin Pie 8.95 to your shopping cart

Shopping cart contents:
Qty Item............     Price  Subtotal
  2 Turkey..........     26.95     53.90
  1 Pumpkin Pie.....      8.95      8.95
Your cart total is           $     62.85
```

We have made a minor change in the main loop as well:

```
choice = input("Product number: ")
p = int(choice) -1   # Item 1 data is in list at offset 0
item = products[p]
price = prices[p]
```