3. WRITTEN RESPONSES

3 a.

3.a.i.

The purpose of the program is to help people use an online pharmacy so they can make a personal kit with a variety of products and differing prices that might help them in a pandemic after their purchase.

3.a.ii.

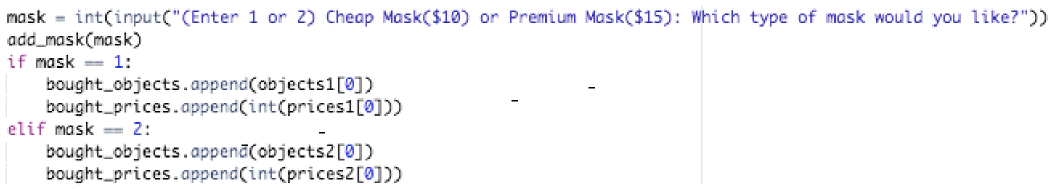
The customer gets offered 4 items, which have 2 options to choose from each. Items differ according to quality and price, in which the product is added to their cart to create a kit that helps a customer's necessities.

3.a.iii.

The program asks the user to choose 4 different inputs that vary from 1 or 2 else an error in the customer cart. Each scene outputs images of the chosen item. The program then outputs 8 results which are the item named by their quality and the price of each 4 items selected in a receipt if there is no error.

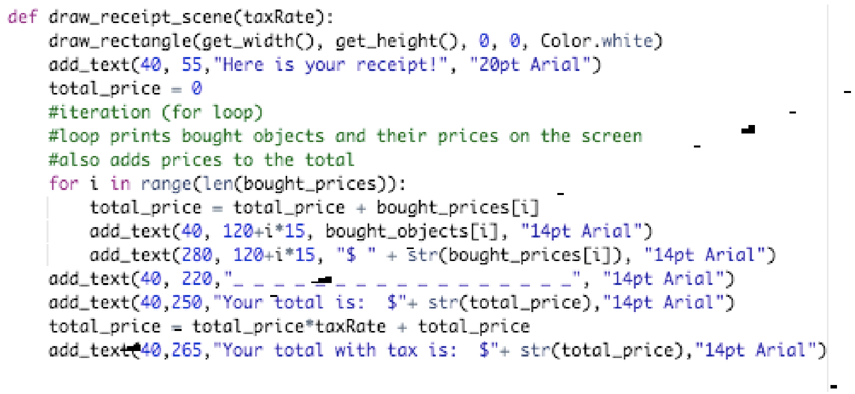
3 b.

3.b.i.



```
mask = int(input("(Enter 1 or 2) Cheap Mask($10) or Premium Mask($15): Which type of mask would you like?"))
add_mask(mask)
if mask == 1:
    bought_objects.append(objects1[0])
    bought_prices.append(int(prices1[0]))
elif mask == 2:
    bought_objects.append(objects2[0])
    bought_prices.append(int(prices2[0]))
```

3.b.ii.



```
def draw_receipt_scene(taxRate):
    draw_rectangle(get_width(), get_height(), 0, 0, Color.white)
    add_text(40, 55,"Here is your receipt!", "20pt Arial")
    total_price = 0
    #iteration (for loop)
    #loop prints bought objects and their prices on the screen
    #also adds prices to the total
    for i in range(len(bought_prices)):
        total_price = total_price + bought_prices[i]
        add_text(40, 120+i*15, bought_objects[i], "14pt Arial")
        add_text(280, 120+i*15, "$ " + str(bought_prices[i]), "14pt Arial")
    add_text(40, 220,"_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _", "14pt Arial")
    add_text(40,250,"Your total is:  $"+ str(total_price),"14pt Arial")
    total_price = total_price*taxRate + total_price
    add_text(40,265,"Your total with tax is:  $"+ str(total_price),"14pt Arial")
```

3.b.iii.

The names of the lists being used are bought_objects and bought_prices. The list of bought_objects stores the chosen item, and bought_prices stores the price of the item which corresponds to the bought_objects order.

3.b.iv.

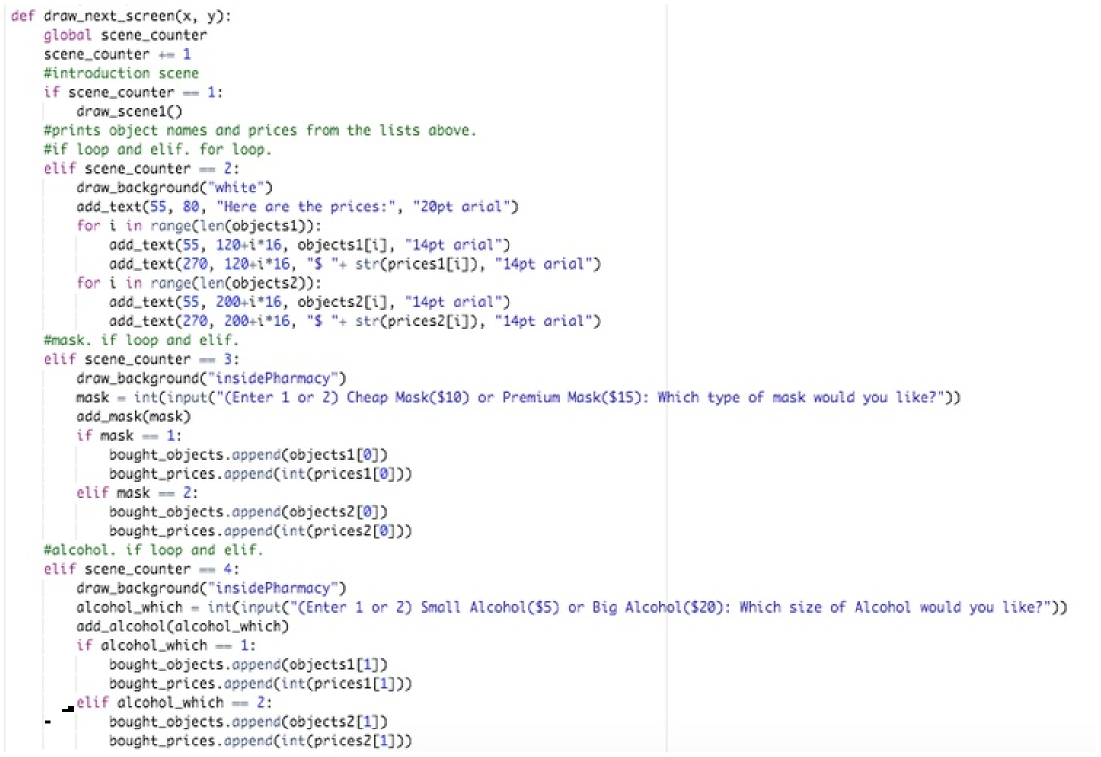
The data in the list bought_objects are the names of the items that the customer chose from the items of objects1 or objects2. The list bought_prices are the prices of items that the customer chose from the items of prices1 or prices2.

3.b.v.

The program code could not be written differently without the list, because the choices would not be stored automatically but written manually. It would be more complex, writing the customer a receipt of items that correspond to what the customer wants. The list makes it capable to use a loop to bring the chosen items and prices making the code more efficient.

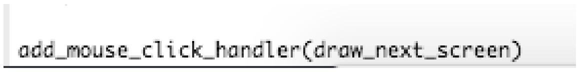
3 c.

3.c.i.



```
def draw_next_screen(x, y):
    global scene_counter
    scene_counter += 1
    #introduction scene
    if scene_counter == 1:
        draw_scene1()
    #prints object names and prices from the lists above.
    #if loop and elif. for loop.
    elif scene_counter == 2:
        draw_background("white")
        add_text(55, 80, "Here are the prices:", "20pt arial")
        for i in range(len(objects1)):
            add_text(55, 120+i*16, objects1[i], "14pt arial")
            add_text(270, 120+i*16, "$ "+ str(prices1[i]), "14pt arial")
        for i in range(len(objects2)):
            add_text(55, 200+i*16, objects2[i], "14pt arial")
            add_text(270, 200+i*16, "$ "+ str(prices2[i]), "14pt arial")
    #mask. if loop and elif.
    elif scene_counter == 3:
        draw_background("insidePharmacy")
        mask = int(input("(Enter 1 or 2) Cheap Mask($10) or Premium Mask($15): Which type of mask would you like?"))
        add_mask(mask)
        if mask == 1:
            bought_objects.append(objects1[0])
            bought_prices.append(int(prices1[0]))
        elif mask == 2:
            bought_objects.append(objects2[0])
            bought_prices.append(int(prices2[0]))
    #alcohol. if loop and elif.
    elif scene_counter == 4:
        draw_background("insidePharmacy")
        alcohol_which = int(input("(Enter 1 or 2) Small Alcohol($5) or Big Alcohol($20): Which size of Alcohol would you like?"))
        add_alcohol(alcohol_which)
        if alcohol_which == 1:
            bought_objects.append(objects1[1])
            bought_prices.append(int(prices1[1]))
        elif alcohol_which == 2:
            bought_objects.append(objects2[1])
            bought_prices.append(int(prices2[1]))
```

3.c.ii.



```
add_mouse_click_handler(draw_next_screen)
```

3.c.iii.

The identified procedure changes scenes by the mouse click of the user. According to the different scenes, it organizes what text and images should appear in order of the online shopping, and storing in the list the selected items for the kit.

3.c.iv.

The algorithm defined a function that contains nested loops for the necessary scenes in the program which continues to the next by a mouse click. Each scene continues into the other by a variable added 1 to continue into each scene after each loop. For the quality selection per item, the object images are brought after a variable is defined from 1 or 2. The variable is added inside the defined item, with an if and elf in item == 1 or item== 2 to be saved in the user blank list of bought_objects and bought_prices with items of 2 separate tuple and arrays that correspond for the cheap item and price or quality item and price. In the end, it brings the receipt of the chosen items and prices and then returns for the final loop the pharmacist with the kit in a cart. All this works with a mouse click handler.

3 d.

3.d.i.

First call:

The first call in which is in the procedure helps execute an order of scenes after each run. This arguments if the scene counted are a specific number, so it can continue to showing the user their way to purchase and make a COVID-19 kit.

Second call:

The second call in the procedure is how the user input of 1 or 2 causes a list to save the prices and the variable of an item of the pharmacy bases on the number. The numbers are later executed in a receipt with the user's choice of item type and price to give the user a total.

3 d.ii.

Condition(s) tested by first call:

The condition tested by the first call is whether a variable for a scene counter equals a specific number from all the scenes offered by the program after each loop. The testing of this condition compares how a variable scene counter equals, to show a correct order of the scenes in the program.

Condition(s) tested by second call:

The condition tested by the second call is whether a variable accord from the number 1 and 2 to save in a tuple and array list differing options of quality and price for the offered objects, to add it into the customer's purchase for a receipt given at the end of the program.

3.d.iii.

Results of the first call:

The result of the first call is displays scenes in order. It helps the scenes match the purpose of the program by ordering the text, images, and questions that are going to appear in the program in order to give it meaning.

Results of the second call:

The second call organizes each item purchased, to output the chosen items and their prices individually at the end in a receipt from the user's input.