CODELAB I

ASSESSMENT 2: Utility App

Tutor : Ms. Lavanya Mohan

Programming Fundamentals

|  |  |
| --- | --- |
| **Student’s Name:** | James Ryan A. Pajunar |
| **Student Id** | 04-24-139 |
| **Github Repository Name:** | Owner avatar[**Metaverse-Programming-Skill-Portfolio-By-James-Ryan**](https://github.com/JamBloxify/Metaverse-Programming-Skill-Portfolio-By-James-Ryan) |
| **Github Repository Link:** | https://github.com/JamBloxify/Metaverse-Programming-Skill-Portfolio-By-James-Ryan |
| **YouTube Link** | <https://youtu.be/e5WcuFPbsSs> |

**BRIEF**

**Your task is to create a Vending Machine program using the Python programming language. The program should demonstrate your knowledge of programming and make use of the techniques introduced over the course of the module. Your application should be accompanied by a development document**



**Specifications**

**(A short explanation of what you have been asked to build and a list of features that your vending machine includes. This section should also include a link to your GitHub repository.)**

I have been tasked to create a vending machine in Python programming, to be displayed through the console. The vending machine consists of ASCII art of the machine at the start, a menu display alongside categories that split the items between drinks and snacks, a payment procedure that keeps track of what the user orders, ID's to differentiate the items, a message that tells the user that a particular snack or drink has been dispensed, and a message that tells that user the change they have received. The machine allows users to buy more items and then choose to pay afterward.

**System Flowchart**

A diagram of a flowchart

Description automatically generated



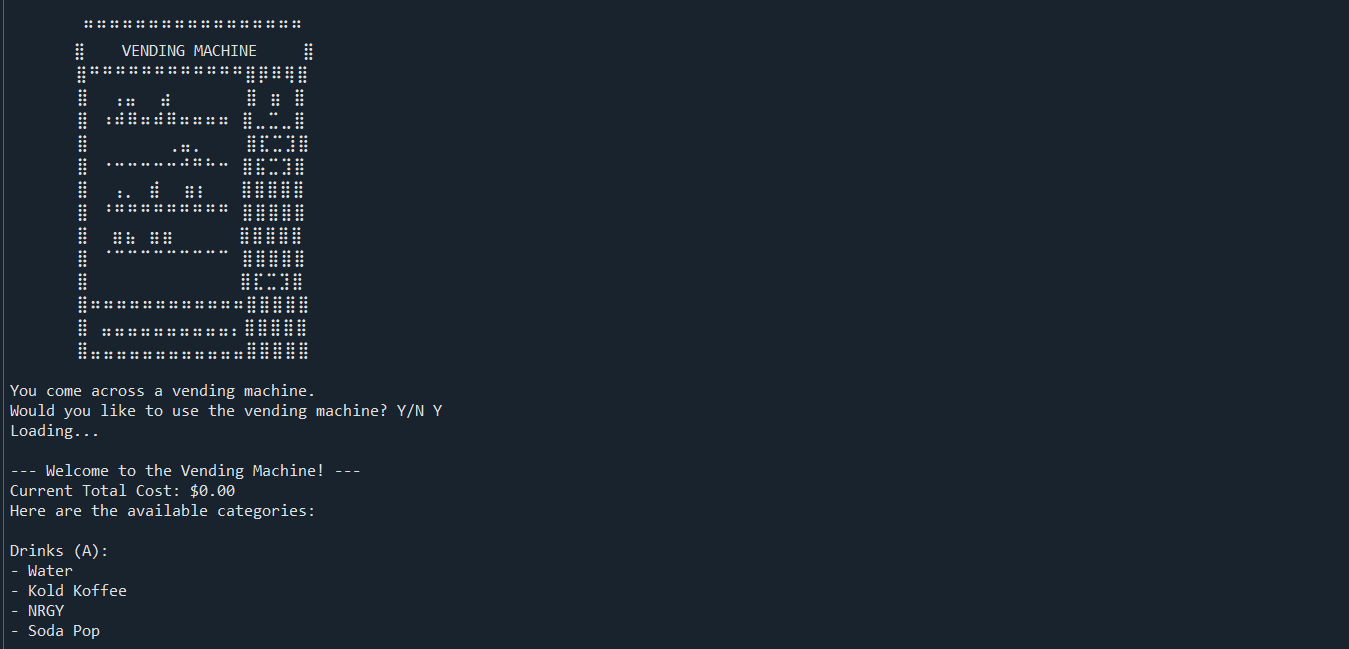
**Technical Description**

**(For this section you should provide a video which includes a walkthrough of your program running as well as a technical breakdown of your code. This technical breakdown should explain how the key features of your program have been implemented via code. The video technical description & walkthrough contributes to the overall word count at approx. 100 words per minute. The anticipated length of the video should be between 5 - 8 minutes, anything longer than 10minutes will not be reviewed and may result in a mark penalty.)**

[**https://youtu.be/e5WcuFPbsSs**](https://youtu.be/e5WcuFPbsSs)



**Output**

****

The starting menu. ASCII art of the vending machine followed up by a prompt. Typing Y will display the next menu.

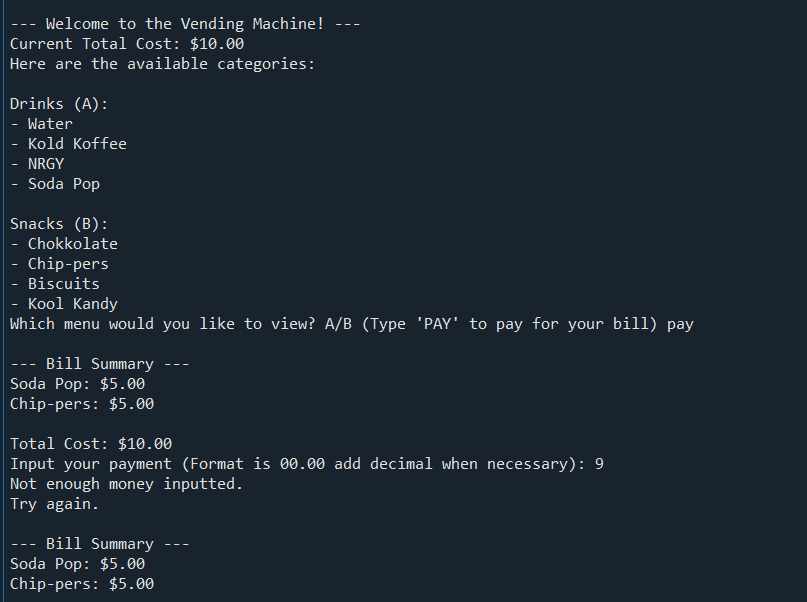
**A screenshot of a computer

Description automatically generated**

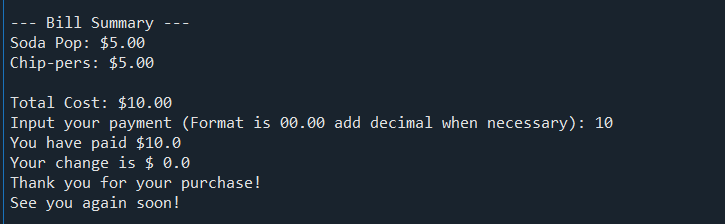
This is the display menu for the vending machine, displaying the categories and the total cost (Which stores the amount of money the user orders from the items). A prompt follows and typing A will take the user into the Drinks category. After loading, it’ll display the drink names, their IDs and corresponding price. Typing the ID into the prompt will tell the user they have bought the item and loops the prompt. Typing ‘back’ into the prompt will take the user back to the previous menu. A screenshot of a computer

Description automatically generated

The menu displays again. Typing B will take the user to the Snacks category. It functions similarly to the previous screenshot, just different IDs.



The menu shows again. Typing PAY (it isn’t case-sensitive, so the user is able to type pay in lowercase) will lead to the bill payment area. It’ll display the items and their price, adding it to display the total cost. A prompt follows, telling the user to input the right amount, and to add decimals if necessary. As shown, if the number is less than the total cost, it will tell the user that not enough money was inputted, looping the bill menu and prompt.



This is the final part. If the user inputs more than or equal to the total cost, it will tell the user how much they have paid, and how much change they have ($0 if exactly the amount). The vending machine stops after this.



**Critical Reflection**

**(This should describe what aspects of your Vending Machine you find compelling, what could be improved, and what programming skills you need to learn to make such improvements.)**

I believe what makes my vending machine compelling is the attention to detail with how it runs. The delay before the display shows up provides a sense of realism and aesthetics, like the ASCII art that was added at the start of the vending machine. If it were the backend, the function that checked for the ID inputted by the user and be able to display it. I didn't know how to do this at first, and went for a much more inefficient method. However, I later learnt through w3schools and a bit of ChatGPT, that I was able to find out a much more efficient way to check the ID. I was even tipped by a classmate for an easier dictionary method, by placing the IDs first instead of the item names. There should be more work on adding more features and quality-of-life aspects to my vending machine. All the required tasks were complete, and some optional tasks, that currently make up my vending machine, but more things could’ve been added. A stock system that would limit the number of items bought. Although I was capable, I could not pull myself to do such a task due to how many other things I had to do, I had to prioritize my time and energy on other works. When it comes to what ways I could make such improvements, it isn't so much about programming skills, but rather needing that time and energy to be able to perform the tasks. I could've added way more, but I was too lazy or prioritizing other works over this to even create the features.



**Appendix**

**(A copy of your code should be included in an appendix at the end of your documentation.)**

import sys # For exiting console because exit() doesn't work.

import time # Adding delay.

cost = 0 # Stores the cost of the products.

bill = [] # Stores the ID of the products.

itemdata = { # Nested Dictionary, with two categories, storing the name, ID and price.

"Drinks (A)" : {

"A1" : {"itemName" : 'Water', "itemPrice" : 1.00},

"A2" : {"itemName" : 'Kold Koffee', "itemPrice" : 3.00},

"A3" : {"itemName" : 'NRGY', "itemPrice" : 5.00},

"A4" : {"itemName" : 'Soda Pop', "itemPrice" : 5.00}

},

"Snacks (B)" : {

"B1" : {"itemName" : 'Chokkolate', "itemPrice" : 3.50},

"B2" : {"itemName" : 'Chip-pers', "itemPrice" : 5.00},

"B3" : {"itemName" : 'Biscuits', "itemPrice" : 2.50},

"B4" : {"itemName" : 'Kool Kandy',"itemPrice" : 4.50}

},

}

def displayMenu(cost): # Created a function that displays the menu with the drinks and snacks, and has global variable.

print("\n--- Welcome to the Vending Machine! ---")

print(f"Current Total Cost: ${cost:.2f}") # This prints, using variable 'cost', the total current cost to the user.

print("Here are the available categories:")

for menu, items in itemdata.items(): # for loop that accesses the nested items.

print(f"\n{menu}:") # Accesses the first information stored into the dictionary and prints it, that being 'Drinks' and 'Snacks'.

for itemdetails in items.values(): # An indented for loop that retrieves the values inside items. It skips the ID, and accesses the information nested after.

print(f"- {itemdetails['itemName']}") # Accesses the item names that is nested after ID and prints it, using f" to fill the information, and separating each information with -

def displayDrinks(itemdata): # Created a function that displays the Drink category. Global variable is the dictionary.

print("Available Drinks:")

drinks = itemdata["Drinks (A)"] # Variable made to only access the Drinks category of the dictionary.

for itemid, itemdetails in drinks.items(): # For loop that accesses the ID and other details in Drinks.

itemname = itemdetails['itemName'] # Variable that accesses the dictionary to get the name.

itemprice = itemdetails['itemPrice'] # Variable that accesses the dictionary to get the price.

print(f"- {itemname}: ID: {itemid} | Price: ${itemprice}") # Using f" to fill in the information, separating the name, ID and Price, and prints.

def displaySnacks(itemdata): # This is pretty much the same as the display drinks.

print("Available Snacks:")

snacks = itemdata["Snacks (B)"] # This accesses the snacks instead of the drinks dictionary.

for itemid, itemdetails in snacks.items():

itemname = itemdetails['itemName']

itemprice = itemdetails['itemPrice']

print(f"- {itemname}: ID: {itemid} | Price: ${itemprice}")

def buyDrinks(itemdata): # Function created to ask and act if the user wants to buy drinks in the drinks category

global cost # Using global variable instead of putting it inside the function because it breaks some of the code and prevents it from working.

global bill

while True:

buyA = input("Enter the ID of the item you want to buy: (Type 'back' to go back) ") # User input question that tells the user to enter the ID of the item they want to buy.

if buyA == 'back': # If function to check if the user inputs 'back' into the user input.

displayMenu(cost)

main()

# ^ This sends the user back to the display menu and main function area.

break

correctID = False # Variable made to check if ID inputted exists in table.

for itemid, itemdetails in itemdata["Drinks (A)"].items(): # For loop that accesses Drinks dictionary.

if itemid == buyA: # If function that efficiently checks if the itemID exists from the user input is true.

price = itemdetails["itemPrice"] # Variable that accesses the item's price.

cost += price # This adds the item's price into the global variable 'cost'.

bill.append(buyA) # Adds the user inputted ID into the front of the variable 'bill', thus why append is used.

print(f"Purchased {itemdetails['itemName']} for ${itemdetails['itemPrice']}. Total Cost: ${cost:.2f}") # Using f" to fill the information that tells the user what they purchased for how much, then displays the total cost of everything they have bought.

correctID = True # Sets the variable to true as the if statement is true.

time.sleep(2) # Delay for aesthetic purposes

break # Breaks the if loop, allowing the buyA variable to loop.

if not correctID: # Checks for opposite condition, where it checks if the correctID is true.

print("Invalid Input!") # Tells the user the ID they inputted is invalid

buyA = str(input("Enter any key to continue: ")) # Changes the buyA variable in order to restart the loop.

# The buySnacks is the same as buyDrinks. Only the dictionary changed.

def buySnacks(itemdata):

global cost

global bill

while True:

buyB = input("Enter the ID of the item you want to buy: (Type 'back' to go back) ")

if buyB == 'back':

displayMenu(cost)

main()

break

correctID = False

for itemid, itemdetails in itemdata["Snacks (B)"].items():

if itemid == buyB:

price = itemdetails["itemPrice"]

cost += price

bill.append(buyB)

print(f"Purchased {itemdetails['itemName']} for ${itemdetails['itemPrice']}. Total Cost: ${cost:.2f}")

correctID = True

time.sleep(2)

break

if not correctID:

print("Invalid Input!")

buyB = str(input("Enter any key to continue: "))

def main(): # Main interface in which the user is able to access the other functions of the vending machine

while True: # while loop that will repeat if an input is invalid.

dors = input("Which menu would you like to view? A/B (Type 'PAY' to pay for your bill) ").casefold() # Asks for user input whether to check the categories or pay.

if dors=="A".casefold(): # Checks if input is A, which is the Drinks category

print("Loading...") # Aesthetic

time.sleep(1)

displayDrinks(itemdata) # Opens the drink category

time.sleep(1)

buyDrinks(itemdata) # displays the input question to the user

break # Stops the loop

elif dors=="B".casefold(): # ^ Same is applied here from above, just checks if it is snacks instead.

print("Loading...")

time.sleep(1)

displaySnacks(itemdata)

time.sleep(1)

buySnacks(itemdata)

break

elif dors == 'PAY'.casefold(): # Checks if user would like to pay instead

displayBill(itemdata, bill) # Displays the bill function

break # Stops loop

else: # Else in case the user inputs an invalid input

print("Invalid Menu!") # Tells the user

dors = str(input("Enter any key to continue: ")) # used to restart the loop

else:

sys.exit() # Not sure why this is here, it doesn't change anything nor does it affect any of the code. I'm going to leave it here IN CASE it does but it's apparently exiting cus the user inputs so?

def displayBill(itemdata, bill): # Payment and bill function is made here

if not bill: # Checks for the opposite condition, in which case, the bill is empty.

print("Your bill is empty.") # Tells the user

time.sleep(2)

main() # Returns to the main function

return # stops this function

print("\n--- Bill Summary ---")

totalcost = 0 # Stores the cost of the ID stored in the bill variable

for item\_id in bill: # This for loop is to check the price since the bill variable stores the ID.

for category in itemdata.values(): # category variable to check the dictionary values

if item\_id in category: # checks if item ID is inside the dictionary values

item\_name = category[item\_id]['itemName'] # variable name is obtained from itemName

item\_price = category[item\_id]['itemPrice'] # variable price is obtained from itemPrice

totalcost += item\_price # the item price is added into total cost variable

print(f"{item\_name}: ${item\_price:.2f}") # Using f", it fills the information of the name and price of the item corresponding to the ID in the bill variable

print(f"\nTotal Cost: ${totalcost:.2f}") # prints the total cost with 2 decimal points

pay = float(input("Input your payment (Format is 00.00 add decimal when necessary): ")) # Tells the user to input their payment, and also provides reference to the decimal format.

change = pay - totalcost # change variable minuses the total cost from the pay

if pay >= totalcost: # checks if the input is more than or equal to the total cost variable

print(f"You have paid ${pay}") # tells the user how much they have paid by filling in the information

print("Your change is $",change) # tells the user their change using the change variable

time.sleep(1)

print("Thank you for your purchase!") # Thanks the user

time.sleep(1)

print("See you again soon!") # See you again by Tyler, The Creator (Or Wiz Khalifa, depends on what you listened to first)

return

elif pay < totalcost: # Checks if the pay is less than the total cost

print("Not enough money inputted.") # Tells the user that they're broke

time.sleep(2)

print("Try again.") # Try again

time.sleep(2)

displayBill(itemdata, bill) # Displays the bill once more and loops it

# Wow, very cool ascii art of a vending machine

print("""

⠛⠛⠛⠛⠛⠛⠛⠛⠛⠛⠛⠛⠛⠛⠛⠛⠛

⣿ VENDING MACHINE ⣿

⠀ ⣿⠛⠛⠛⠛⠛⠛⠛⠛⠛⠛⠛⠛⣿⡿⠿⢿⣿⠀⠀⠀⠀⠀⠀

⠀⠀⠀⠀⠀⠀⣿⠀⠀⢠⣤⠀⠀⣴⠀⠀⠀⠀⠀ ⣿⠀⣶⠀⣿⠀⠀⠀⠀⠀⠀

⠀⠀⠀⠀⠀⠀⣿⠀⠰⠾⠿⠶⠾⠿⠶⠶⠶⠶⠀⣿⣀⣉⣀⣿⠀⠀⠀⠀⠀⠀

⠀⠀⠀⠀⠀⠀⣿⠀⠀⠀⠀⠀⠀⠀⢀⣤⡀⠀⠀ ⣿⣏⣉⣹⣿⠀⠀⠀⠀⠀⠀

⠀⠀⠀⠀⠀⠀⣿⠀⠐⠒⠒⠒⠒⠒⠚⠛⠓⠒⠀⣿⣯⣉⣹⣿⠀⠀⠀⠀⠀⠀

⠀⠀⠀⠀⠀⠀⣿⠀⠀⢠⡀⠀⣾⠀⠀⣶⡆⠀⠀ ⣿⣿⣿⣿⣿⠀⠀⠀⠀⠀⠀

⠀⠀⠀⠀⠀⠀⣿⠀⠘⠛⠛⠛⠛⠛⠛⠛⠛⠛⠀⣿⣿⣿⣿⣿⠀⠀⠀⠀⠀⠀

⠀⠀⠀⠀⠀⠀⣿⠀⠀⣶⣦⠀⣶⣶⠀⠀⠀⠀⠀ ⣿⣿⣿⣿⣿⠀⠀⠀⠀⠀⠀

⠀⠀⠀⠀⠀⠀⣿⠀⠈⠉⠉⠉⠉⠉⠉⠉⠉⠉⠀⣿⣿⣿⣿⣿⠀⠀⠀⠀⠀⠀

⠀⠀⠀⠀⠀⠀⣿⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀ ⣿⣏⣉⣹⣿⠀⠀⠀⠀⠀⠀

⠀⠀⠀⠀⠀⠀⣿⠶⠶⠶⠶⠶⠶⠶⠶⠶⠶⠶⠶⣿⣿⣿⣿⣿⠀⠀⠀⠀⠀⠀

⠀⠀⠀⠀⠀⠀⣿⠀⣤⣤⣤⣤⣤⣤⣤⣤⣤⣤⡄⣿⣿⣿⣿⣿⠀⠀⠀⠀⠀⠀

⠀⠀⠀⠀⠀⠀⣿⣤⣤⣤⣤⣤⣤⣤⣤⣤⣤⣤⣤⣿⣿⣿⣿⣿⠀⠀⠀⠀⠀⠀

""")

print("""You come across a vending machine.""") # No way, vending machine

time.sleep(2)

start = input("Would you like to use the vending machine? Y/N ").casefold() # No

if start == "Y".casefold(): # Checks if user inputted Y

print("Loading...")

time.sleep(2)

displayMenu(cost) #

main()

# where did my N go 💀? It's fine, it works normal if you do N, it'll exit anyways.

# I got everything important done.

# • A menu of drinks and snacks presented via the console. The number and range of items is up to you. ✅

# • A set of numbers or codes that the user can input to select a particular drink or snack. ✅

# • A way of managing money so the user can input any amount of money and have the correct change returned. ✅

# • A message that tells the user that a particular drink or snack has been dispensed. ✅

# • A message that tells the user how much change they have received. ✅

# • Comments in the code to explain key operations. ✅ (I did comments on nearly everything though but it's fine.)

# As for additional ones as stated in the document.

# • A method of categorising items in the vending machine to improve the user experience (e.g. ‘Chocolate’ or ‘Hot Drinks’). ✅ (Drinks and Snacks, that's it)

# • A way of allowing users to buy additional items. ✅

# • Appropriate error checking to validate inputs and ensure the user has enough money 🟡 (Erm, I did for majority of them. Few just won't work)

# • An intelligence system for suggesting purchases. For example, if you buy a coffee, the vending machine may suggest that you buy biscuits. ❌ (no.)

# • A stock system meaning the machine may run out of products. ❌ (Absolutely not. I could, but I am too lazy.)

# That is all, folks, I shall document now.



**Reference List**

W3schools

ChatGPT

StackOverflow

GeeksforGeeks

Reddit

YouTube