






Project 4

Extensible Programs

 Date	Nov 27, 2023
 Module	Module 5: File Manipulation & Python Packages
 Type	Project

General Instructions

Problem: POS Machine

Scoring: 100 pts

General Instructions

1. Create one Jupyter Notebook file for this project.
2. The first cell of the file should contain header information:
 1. Your full name and ID number
 2. Date you submitted your program
3. Save the solution to the problem in a single code cell.
4. Save your Jupyter Notebook as **Project04-SURNAMEUsername-idnumber**. Zip the folder containing your Jupyter Notebook submission using the same name. Ensure that you include a properly accomplished Certificate of Authorship (Individual), citing all external sources used, in the zipped folder.
5. The zipped folder should be submitted through the appropriate Canvas submission module. Make sure to double check to see that you've successfully uploaded just that one file--the correct zipped folder.

Problem: POS Machine

A point-of-sales (POS) machine should be able to track the price of products, create transactions given the number of products to be purchased, and print the sales invoice for the customer. It should be able to:

1. store information about the prices of products and update this data when it is prompted,
2. create transactions—accepting product-ID and quantity-to-be-bought pairs until stopped,
3. assemble a transaction summary containing each product-price-quantity-amount combo in that transaction, with the total amount found at the end,
4. print out (into the console) and save (as a .txt file) the information from number 3.

Download the [Project 4 - Starter](#) .ipynb file from the drive where you got this specs file and edit it to build your fully-functional `POSMachine` class which has the following behaviors:

- (1) `__init__(self)`: initializes all fields
- (2) `start(self)`: starts the machine and prints 2 messages
 - ~ first prints a welcome message just once
 - ~ then prints a list of commands for the user to choose input from
 - ~ if the command chosen is to close the machine, asks if the user is sure—if so, close the machine
 - ~ otherwise, that command is completed and the list of command for the user are printed again
 - ~ every time a command is completed, the list of commands gets printed again
 - ~ keeps accepting commands unless the user chooses close, and is then sure about it
 - ~ consider that users might give error-inducing input, ensure your machine does not crash
- (3) `printWelcomeMessage(self)`: prints your own welcome message for the user
- (4) `printAllOptions(self)`: prints all commands the user can input
 - ~ set the price list
 - ~ show the price list
 - ~ make a new transaction
 - ~ close the POS machine
- (5) `setPriceList(self)`: loads the price list from a provided file
 - ~ prints a warning if the file type is invalid
 - ~ .csv files are valid: values are separated by a comma
 - ~ .txt files are valid: values are separated by a tab (Python escape character: `\t`)
- (6) `printPriceList(self)`: prints the current price list—showing product IDs, names, and prices
- (7) `makeNewTransaction(self)`:
 - ~ create a new transaction and then print its sales invoice afterwards
 - ~ prompt the user to add products (provided the product's ID and given that the quantity ≥ 1)
 - ~ when the user inputs a blank, the transaction is completed and a sales invoice is created
 - ~ consider that users might give error-inducing input, ensure your machine does not crash
- (8) `addProductToTransaction(self, productID, quantity)`:
 - ~ calculate the price for the given product and quantity
 - ~ add this to the current invoice
- (9) `createSalesInvoice(self)`:
 - ~ prints the sales invoice—showing name, quantity, cost for each item, and then the total cost
 - ~ save that same output as a tab-separated text file (.txt)

The following files are available in the same drive where you found this projects specs file: [Project 4 - Starter.ipynb](#), [price_list1.csv](#), and [price_list2.txt](#). No need to create these files, just use them. Also available for your reference are: [sample_invoice1.txt](#), and [sample_invoice2.txt](#).

At this point in the semester, it would be best to already know how to test your own code based on the instructions given. Remember to exhaust possibilities, not merely by how many test items you try to input, but by how many different types of test cases you try to input. Below is the scoring rubric that can serve as your guide.

Scoring: 100 pts

→ Correct Outputs[50 pts]:

- ~ 50: Correct outputs 100% of the time
- ~ 40: Correct outputs for most, but not all cases
- ~ 30: Correct outputs 50% of the time
- ~ 15: Correct outputs 25% of the time
- ~ 5: Correct outputs for a few cases
- ~ 0: No correct outputs

→ Intended Functionality[20 pts]:

- ~ 20: Abides with 100% of the given instructions
- ~ 15: Abides with most of the given instructions
- ~ 10: Abides with ~50% of the given instructions
- ~ 5: Abides with some of the given instructions
- ~ 0: Instructions and Hints were disregarded

→ Concise Design[30 pts]:

- ~ 30: Every line of code has a unique and relevant contribution
- ~ 25: A negligible amount of redundant or unnecessary lines of code
- ~ 20: Noticeable amount of redundant or unnecessary code
- ~ 15: 1 in 2 lines of code is redundant or unnecessary
- ~ 5: The redundant or unnecessary lines of code outnumber the useful ones
- ~ 0: The code does not work at all