Intro to Programming

ASSESSMENT 2: Utility App

|  |  |
| --- | --- |
| Contribution towards overall module mark | 60% |
| Date set | November 15, 2023 |
| Marked work returned by | Within 3 weeks of submission |
| DEADLINES | Deadline : January 10, 2024 – 23:59 |

**Assessment 2: Utility App**

|  |  |
| --- | --- |
| **Student’s Name** | DANISH BALOCH |
| **Roll No** | 2023945 |
| **Github Repository Name** | DANISHMETAVERSE |
| **Github Repository Link** | <https://github.com/IntroToProgramming-CCL4/assessments-DANISHMETAVERSE/tree/main/Assessment%202> |
| **Repository Screen Shot** |  |

**The Development Document**

**We have been tasked to create a working Vending Machine program using the Python Programming language. The program should demonstrate our knowledge of programming and make use of the techniques introduced over the course of the module including the Assessment 1.**

**I Created a python program that simulates a simple fruit shop vending machine where users can interact with the program by buying fruits in specific quantities. The program maintains a menu of various fruits each associated with a code, the name of the fruit, and their prices per kilogram. The program calculates the total cost, updates the remaining balance, and allows users to continue shopping or proceed to checkout. After the user decides to checkout, the program validates the balance, generates a receipt with the details of the purchased items, their remaining balance. This inclusion of loops, conditionals, and a dictionary that contains the fruit menu demonstrates some of the key concepts of python programming**

**The first line ‘import time’ allows you to use the time module in your python scripts to work with time, I decided to implement this function in my receipt to make it look like the real life receipts.**

**‘Cart = []’ is an empty list called cart so the script can keep track of the items that the user selected. For ‘money = int(input(‘insert money: ‘))” it asks for a user for a integer that represents their balance, the variable is called ‘money’ and it converts the input to an integer if the user inputs a non integer, it will not accept it and rather show an error hinting that you put something in the input money box that's not cash! The print(current balance: $’, money’: prints current balance that u had inserted to the machine earlier. The ‘brand’ dictionary contains information about the fruits being sold in the vending machine including their respective codes, names, and prices per kilogram. The program displays the menu with their codes and prices in a formatted way using a for loop. The ‘price1 = 0’: initializes price1 outside the loop to keep track of the balance/total cost of the items that have been selected by the user. The main loop ‘while true’ reminds the user over and over again to input the code of the fruit that they want to purchase. The first inner loop ensures that the user enters a valid code. If the user enters a invalid code, the user is prompted to enter a valid code again or quit by pressing ‘Q’. If the user enters ‘Q’ the script prints a message, gives back the money and exits. If a valid code is entered, the program asks how many kilograms and calculates the cost based on the price per kilogram and the input that the user gave (input \* kilograms). The program enters a second loop (‘while True:’) that allows the user to continue shopping or proceed to checkout. If the user chooses to shop more (’action == ‘S’), the process repeats and the total cost (‘price1’) is updated. If the user chooses to proceed to checkout (action == 'Q'), the program checks if the remaining balance is sufficient. If not, it prompts the user to add more money or remove items from the cart. If the balance is sufficient, the program prints a thank you message, updates the remaining balance, displays a receipt, and exits the program. The receipt includes the selected items, quantities, prices, total cost, remaining balance, and the current date and time.**

**For the compelling aspects, my script engages users effectively by allowing them to input their money, select fruits, and complete the purchase process. The menu display is also clear making it easy for the user to navigate and understand the selection of desired fruits. It’s also effective on error handling and effectively guiding users through input mistakes and enhancing overall user experience. The script also manages the transaction well, updating the cart and providing a simple but clear receipt upon checkout. Overall for the compelling aspects, I believe my code is well-structured making it relatively easy to comprehend and modify.**

**For the areas of improvement I have a few. I believe breaking down my code into functions could enhance readability and maintainability. I also believe I can enhance the cart for accurate receipt generation and a better shopping experience. I could also add more information on the receipt instead of just having the time and remaining balance. Even though I added comments I believe I could have added more to improve the understanding of the code for future maintenance or modifications.**

**For the skills to learn for improvement, I would love to explore more string formatting techniques to represent outputs consistently and aesthetically.I would also like to learn enhancing the error handling techniques for me to create a more robust user experience.**

**Overall I’m very proud of what I’ve created considering that I’m new to python. Those sleepless nights figuring out the problem of my code scratching my head was worth it after all.**

**FLOWCHART**

**We have been tasked to create a working Vending Machine program using the Python Programming language. The program should demonstrate our knowledge of programming and make use of the techniques introduced over the course of the module including the Assessment 1.**

**I Created a python program that simulates a simple fruit shop vending machine where users can interact with the program by buying fruits in specific quantities. The program maintains a menu of various fruits each associated with a code, the name of the fruit, and their prices per kilogram. The program calculates the total cost, updates the remaining balance, and allows users to continue shopping or proceed to checkout. After the user decides to checkout, the program validates the balance, generates a receipt with the details of the purchased items, their remaining balance. This inclusion of loops, conditionals, and a dictionary that contains the fruit menu demonstrates some of the key concepts of python programming**

**The first line ‘import time’ allows you to use the time module in your python scripts to work with time, I decided to implement this function in my receipt to make it look like the real life receipts.**

**‘Cart = []’ is an empty list called cart so the script can keep track of the items that the user selected. For ‘money = int(input(‘insert money: ‘))” it asks for a user for a integer that represents their balance, the variable is called ‘money’ and it converts the input to an integer if the user inputs a non integer, it will not accept it and rather show an error hinting that you put something in the input money box that's not cash! The print(current balance: $’, money’: prints current balance that u had inserted to the machine earlier. The ‘brand’ dictionary contains information about the fruits being sold in the vending machine including their respective codes, names, and prices per kilogram. The program displays the menu with their codes and prices in a formatted way using a for loop. The ‘price1 = 0’: initializes price1 outside the loop to keep track of the balance/total cost of the items that have been selected by the user. The main loop ‘while true’ reminds the user over and over again to input the code of the fruit that they want to purchase. The first inner loop ensures that the user enters a valid code. If the user enters a invalid code, the user is prompted to enter a valid code again or quit by pressing ‘Q’. If the user enters ‘Q’ the script prints a message, gives back the money and exits. If a valid code is entered, the program asks how many kilograms and calculates the cost based on the price per kilogram and the input that the user gave (input \* kilograms). The program enters a second loop (‘while True:’) that allows the user to continue shopping or proceed to checkout. If the user chooses to shop more (’action == ‘S’), the process repeats and the total cost (‘price1’) is updated. If the user chooses to proceed to checkout (action == 'Q'), the program checks if the remaining balance is sufficient. If not, it prompts the user to add more money or remove items from the cart. If the balance is sufficient, the program prints a thank you message, updates the remaining balance, displays a receipt, and exits the program. The receipt includes the selected items, quantities, prices, total cost, remaining balance, and the current date and time.**

**For the compelling aspects, my script engages users effectively by allowing them to input their money, select fruits, and complete the purchase process. The menu display is also clear making it easy for the user to navigate and understand the selection of desired fruits. It’s also effective on error handling and effectively guiding users through input mistakes and enhancing overall user experience. The script also manages the transaction well, updating the cart and providing a simple but clear receipt upon checkout. Overall for the compelling aspects, I believe my code is well-structured making it relatively easy to comprehend and modify.**

**For the areas of improvement I have a few. I believe breaking down my code into functions could enhance readability and maintainability. I also believe I can enhance the cart for accurate receipt generation and a better shopping experience. I could also add more information on the receipt instead of just having the time and remaining balance. Even though I added comments I believe I could have added more to improve the understanding of the code for future maintenance or modifications.**

**For the skills to learn for improvement, I would love to explore more string formatting techniques to represent outputs consistently and aesthetically.I would also like to learn enhancing the error handling techniques for me to create a more robust user experience.**

**Overall I’m very proud of what I’ve created considering that I’m new to python. Those sleepless nights figuring out the problem of my code scratching my head was worth it after all.**

**FLOWCHART**

[**https://drive.google.com/file/d/157FmfzZwmA9mgd\_XHFJA321Fn3TeqTAs/view?usp=sharing**](https://drive.google.com/file/d/157FmfzZwmA9mgd_XHFJA321Fn3TeqTAs/view?usp=sharing)