Penn Engineering
ONLINE LEARNING

Programming Languages and Techniques

HW 2: Supermarket

HW deadline as per Canvas.

This assignment deals with the following topics:

- Getting user input
- Error checking
- Variables & data types
- Conditionals

General Idea of the Assignment

In this assignment, you will implement a supermarket shopping "game".

Imagine you are shopping in a supermarket. This supermarket sells only four items: lottery tickets, apples, cans of beans, and soda. Lottery tickets cost $2 each, apples cost $0.99 each, cans of beans cost $1.58 each, and sodas cost $1.23 each.

You have $5 to start with. At the beginning of your shopping trip, you are offered the opportunity to buy a lottery ticket for a chance to win $2-$10. You are then sequentially asked whether you want to buy apples, cans of beans, and sodas. If you choose to purchase an item, you are asked to specify the quantity of this product. At the end of your shopping trip, you are provided with a list of products purchased and how much money you have left.

NOTE: Handle negative input from user as invalid input. If negative inputs are not handled correctly(for instance, user asks to buy apples of quantity -7), it might result in a state where the shopper has more money left after all their purchase than what they started with. Don't let your supermarket run out of business!!

Provided program implementation:

We have provided a *supermarket.py* skeleton file which includes:

1. 4 defined variables storing the unit price of an individual lottery ticket, the unit price of an individual apple, the unit price of an individual can of beans, and the unit price of an individual soda.
    a. Use these variables throughout your program to reference the unit prices of specific items.
2. 2 defined variables storing the initial money the user has and the money the user has spent. a. Use these variables throughout your program to keep track of the total amount of money the user has left and the amount of money the user is currently spending on a particular item.
3. 4 defined variables storing the amounts of lottery tickets, apples, cans of beans, and sodas the user has purchased.

a. Use these variables throughout your program to keep track of the quantities of the items being purchased.

Program logic:

1. The user will be given $5 to start shopping
2. First, print a welcome message to the user along with a list of products and their unit prices.
3. Next, tell the user how much money they have available and ask if they want to purchase a lottery ticket.
    a. If the user inputs “y” or “Y”, process a lottery ticket purchase (see Step 3b). If the user inputs anything else (e.g. “n” or “N”), print a message saying that no lottery tickets were purchased and move on to the next item.
    b. If the user chooses to purchase a lottery ticket, you will need to use the *random* module. The probability that the user wins the lottery is 33%. You can generate a random int from 0-2 to simulate this probability.
        i. If the user loses, print a message informing them that they did not win the lottery, and move on to the next item. Remember to deduct the $2 the user spent on the lottery ticket from their available money and increase the amount of lottery tickets purchased by 1.
        ii. If the user wins, you then need to calculate their winnings. You can use `winnings = random.randint(2, 10)` to generate a random int from 2 – 10 and store it in a variable "winnings". Consider this money earned and add it back to the money the user has available. Remember to also deduct the $2 the user spent on the lottery ticket and increase the amount of lottery tickets purchased by 1. Finally, print a congratulatory message to the user telling them the value of the lottery ticket.
4. Next, tell the user how much money they have available and ask if they want to purchase apples.
    a. If the user inputs “y” or “Y”, process an apple purchase (see Step 4b). If the user inputs anything else (e.g. “n” or “N”), print a message saying that no apples were purchased and move on to the next item.
    b. If the user chooses to purchase apple(s), ask them how many they want to buy.
        i. You should cast the input to an integer and catch the error if the input cannot be cast to an integer. In this case, ignore the input, print a friendly message reminding the user that only numerical values are accepted and move on to the next item.
        ii. If the user entered valid integer input, calculate the money they will need to pay. To do this, use the unit price of the item and the desired amount of the item. Then print the amount the user wants to purchase and how much it will cost. For example, “The user wants to buy 1 apple(s). This will cost $0.99.”

If the user doesn't have enough money to pay, print "Not enough money" and move on to the next item. If the user has enough money, add the number of apple(s) purchased to the variable representing the total amount of apples that the user has purchased and decrease the money that the user has left.

5. Repeat Step 4 for cans of beans and for sodas.
6. At the end of the shopping trip, tell the user how much money they have left and print the following information as well: number of lottery tickets purchased, amount of lottery winnings (this should be $0 if the user did not purchase a ticket or did not win the lottery), number of apple(s) purchased, number of can(s) of beans purchased, number of soda(s) purchased.

Program Output

Print out what the program is doing as it goes along. We have provided a *template_behavior.txt* file which shows some sample runs of the program -- yours should provide similar information.

Note, at any point when you are printing how much money the user has left, or how much an item will cost, you can round the resulting float values. To round a float, You can use Python's built-in *round* function. For example:

```
round(12374/621) #rounds to the nearest integer, which is
20 round(45.367, 2) #rounds to 2 decimal places, which is
45.37
```

Submission

Your submission should include:

- *supermarket.py* - the source code for your game

This file must include a header, in the form of a multi-line comment at the top of your script, that contains (each on its own line):

- Your name
- Your Penn ID
- Statement of work. Either:
  - A list of resources you used and/or people you received help from (including TAs/Instructor)
  - A statement that you worked alone without help

Evaluation

Correctness - 20 pts

Does the game work as expected? Did you follow the directions exactly? Is your math correct? If the user doesn't have enough money or the user enters invalid input, does the program work well (and print a useful message)? Do you print welcome/goodbye messages with clear information for the user?

Comments – 2 pts

Did you add clear and descriptive comments to all non-trivial lines of code?