INFO 190S

Project 2 Change Maker

Overview

This project will give you more experience on the use of:

- integers (int)
- floats (float)
- strings (str)
- branching
- loops

Your program will simulate a simple change maker for a vending machine. It will start with a stock of coins and dollars. It will then repeatedly request the price for an item to be purchased or to quit. If given a price, it will accept nickels, dimes, quarters, one-dollar and five-dollar bills, deposited one at a time, in payment. When the user has deposited enough to cover the cost of the item, the program will calculate the coins to be dispensed in change. Alternatively, the user can cancel the purchase up until full payment has been deposited, in which case, your program will calculate the coins to be dispensed to refund any partial payment already deposited. With each purchase, the program will update the stock of coins and dollars. Before quitting, it will output the remaining stock of coins and dollars. The specifications are spelled out more thoroughly below. An example interaction with our program appears at the end of this description. All change and refunds must be in coins only, and must use the minimum number of coins possible.

Background

The algorithm for calculating the numbers of coins of each denomination to dispense so as to minimize the total number of coins is an example of a greedy algorithm. Since each coin is at least twice the value of the previous coin, you can start by figuring out the most number of quarters you can dispense (without exceeding the amount), then the most number of dimes, and then the number of nickels. Knowledge of greedy algorithms is not required to complete this project. If you are curious, you can read http://en.wikipedia.org/wiki/Greedy_algorithm.

Project Specification

Your program must produce identical output as our program. You can review the output of our program at the end of this document. Please be very careful in producing the output for your program. If it is off in the slightest, the auto-grader may not be able to give you credit. However, the auto-grader will tell you which parts of your program output differ from its own which will give you some hints on where you went wrong.

Your program must meet the following specifications:

General

- ☐ You must implement your solution in a single python file named **change_maker.py**. If you do not name your program file correctly, you will not receive credit.
- ☐ At program start, assume a stock of 25 nickels, 25 dimes, and 25 quarters. Print the contents of the stock.
- ☐ Repeatedly prompt the user for a price in the form xx.xx, where x denotes a digit, or to enter ‘q’ to quit.
- ☐ Just before quitting, print the total amount (the number of dollars and number of cents) left in the stock.

When a Price is Entered

- ☐ Check that the price entered is a (non-negative) multiple of .05 (i.e., it is payable in nickels). If not, then print an error message and start over requesting either a new price or to quit (indicated by entering a ‘q’).
- ☐ Print a menu for indicating the coin/bill deposited or to cancel payment.
- ☐ Prompt for a selection from this menu.
- ☐ If the user enters an illegal selection, re-prompt for a correct one.
- ☐ Following each deposit, print the remaining amount owed (indicate the number of dollars and the number of cents).
- ☐ When full payment has been deposited or a ‘c’ has been entered, determine the coins to be dispensed in change or as a refund. This calculation will depend on the amount to be dispensed and also on the number of coins left in the stock. For example, the least number of coins needed to make up $1.30 is 6 (5 quarters and 1 nickel). But if there are only 3 quarters, 3 dimes, and 10 nickels left in the stock, then the least number is 11 (3 quarters, 3 dimes, and 5 nickels).
- ☐ Print the numbers of the coins to be dispensed and their denominations. (Omit a denomination if no coins of that denomination will be dispensed.)
- ☐ In the case where an exact payment is deposited, print a message such as “No change.”
- ☐ If the change cannot be made up with the coins remaining, dispense the coins available without exceeding the change amount and indicate the amount still due to the user,

which will have to be collected from a store manager. For example, if the stock contains one nickel, no dimes, and a quarter and if the change amount is 15 cents, dispense just the nickel and indicate the user should collect 10 cents from a store manager.

- ☐ Print the contents of the stock following the transaction.

Exceeding

To achieve a grade in the A range for this project you must satisfy the following requirements:

- ☐ Check that the purchase price entered is formatted properly as a xx.xx floating-point value, where each x is a digit. If the price is not formatted properly, then print an error message and start over requesting either a new print or to quit. **You may not use any external libraries, exceptions, etc. Only what has been covered in this class.**

Notes and Hints

- Floating point numbers can be difficult to work with due to imprecision. To avoid imprecision in this program, you can multiply the price by 100, round, and convert to an integer (number of cents). For example, $1.15 is the same as 115 cents. To see why you need to round, try evaluating 1.15*100 in the Python shell. Now evaluate round(1.15*100).

- The quotient (//) operation for integers will be useful for finding the numbers of each coin. For example, 195//25 is 7, the most number of quarters in 195 cents. But be careful—if the stock has fewer than 7 quarters left, you will only be able to dispense the number left in the stock. For example, if there are only 6 quarters left, then you can dispense only 6 quarters and must use dimes and nickels to make up any remaining change.

- You can use format strings (f-strings) to more easily and elegantly print monetary amounts. You can also use the Python print(…) command with appropriate string and/or integer arguments to print the number of dollars and the number of cents. Your choice.

- You do not need to check for any input errors other than those mentioned in this description. There is an *exceeding* option that requires you to perform a more detailed check on the inputs to the program.

- The logic for this project is sufficiently complex that you will need to break it into small steps that you can implement incrementally. After implementing each step, test it thoroughly, and back it up (e.g., Google Drive, Dropbox) before trying to implement the next step. For example, you could:
    - First implement the logic to initialize the stock, print it, and repeatedly prompt for a price until a ‘q’ is entered; just print the value that was input. Test and backup your code.

- Next, add code to check that the input is a legal price and, if it is not, to prompt for a new value. Again, just print the value that was input. Test and backup your code.
- Next, add code to print the menu of selections. Test and backup your code.
- Next, add code to repeatedly prompt the user to deposit a coin until the full amount is collected or a ‘c’ is entered and to print the remaining amount owed after each deposit. Test and backup your code.
- Next, add code to update the stock. Test and backup your code.
- Next, add code to calculate the change or refund to be dispensed.
- Next, add code to calculate the number of coins of each denomination to be dispensed and to update the stock.

It is not required that you develop the project exactly in this way. But if you don’t find a way to incrementally build and test your projects, they generally will be more difficult and take you longer to do. We also recommend an incremental strategy, not just because it is easier, but also because it allows the auto-grader an opportunity to give partial credit. If you hand in a program that Gradescope can partially grade and that meets some, but not all, of the requirements, we can score what you managed to get working.

Project Submission and Grading

You must submit a single python named **change_maker.py**. If you do not submit the correctly named file, you will not receive any credit. Make sure your file is named correctly!

You must submit your solution to Gradescope by the assigned due date. You are welcome to submit as many times as you would like. The auto-grader will provide you some help if you do not pass a test. If your program output does not match the auto-grader solution output exactly you must look at what the auto-grader tells you in order to adjust your output to match.

Do not wait until the last minute to work on this project! It will require time for you to think about how to solve the problem and time to implement a correct solution.

Examples

To illustrate, we show results of executing a program that meets the project specifications for several test inputs. In these examples, any lines (if any) that start with >>> were produced by the Python shell. Everything else was produced by print statements in our solution program. The characters in red indicate the test inputs provided as input by a human/user.

Remember, your program must duplicate the output below exactly.

Here are a few example runs of the program to help guide you. The first example is an interaction of using the program without **exceeding** requirements. The second example demonstrates the **exceeding** requirements showing invalid purchase prices.

Sample Interaction (user inputs shown in red):

```
Welcome to the vending machine change maker program
Change maker initialized.
Stock contains:
   25 nickels
   25 dimes
   25 quarters
   0 ones
   0 fives

Enter the purchase price (xx.xx) or `q' to quit: 1.96
Illegal price: Must be a non-negative multiple of 5 cents.

Enter the purchase price (xx.xx) or `q' to quit: 1.95

Menu for deposits:
  'n' - deposit a nickel
  'd' - deposit a dime
  'q' - deposit a quarter
  'o' - deposit a one dollar bill
  'f' - deposit a five dollar bill
  'c' - cancel the purchase

Payment due: 1 dollars and 95 cents
Indicate your deposit: 1
Illegal selection: 1
Payment due: 1 dollars and 95 cents
Indicate your deposit: o
Payment due: 95 cents
Indicate your deposit: o

Please take the change below.
  1 nickels

Stock contains:
   24 nickels
   25 dimes
   25 quarters
   2 ones
   0 fives
```

```
Enter the purchase price (xx.xx) or `q' to quit: 3.25

Menu for deposits:
  'n' - deposit a nickel
  'd' - deposit a dime
  'q' - deposit a quarter
  'o' - deposit a one dollar bill
  'f' - deposit a five dollar bill
  'c' - cancel the purchase

Payment due: 3 dollars and 25 cents
Indicate your deposit: o
Payment due: 2 dollars and 25 cents
Indicate your deposit: d
Payment due: 2 dollars and 15 cents
Indicate your deposit: d
Payment due: 2 dollars and 5 cents
Indicate your deposit: o
Payment due: 1 dollars and 5 cents
Indicate your deposit: d
Payment due: 95 cents
Indicate your deposit: c

Please take the change below.
   9 quarters
   1 nickels

Stock contains:
   23 nickels
   28 dimes
   16 quarters
   4 ones
   0 fives

Enter the purchase price (xx.xx) or `q' to quit: .05

Menu for deposits:
  'n' - deposit a nickel
  'd' - deposit a dime
  'q' - deposit a quarter
  'o' - deposit a one dollar bill
  'f' - deposit a five dollar bill
  'c' - cancel the purchase

Payment due: 5 cents
Indicate your deposit: f

Please take the change below.
```

```
   16 quarters
   9 dimes
   1 nickels

Stock contains:
   22 nickels
   19 dimes
   0 quarters
   4 ones
   1 fives

Enter the purchase price (xx.xx) or `q' to quit: 25

Menu for deposits:
  'n' - deposit a nickel
  'd' - deposit a dime
  'q' - deposit a quarter
  'o' - deposit a one dollar bill
  'f' - deposit a five dollar bill
  'c' - cancel the purchase

Payment due: 25 dollars and 0 cents
Indicate your deposit: f
Payment due: 20 dollars and 0 cents
Indicate your deposit: f
Payment due: 15 dollars and 0 cents
Indicate your deposit: f
Payment due: 10 dollars and 0 cents
Indicate your deposit: f
Payment due: 5 dollars and 0 cents
Indicate your deposit: c

Please take the change below.
   19 dimes
   22 nickels
Machine is out of change.
See store manager for remaining refund.
Amount due is: 17 dollars and 0 cents

Stock contains:
   0 nickels
   0 dimes
   0 quarters
   4 ones
   5 fives

Enter the purchase price (xx.xx) or `q' to quit: .35
```

```
Menu for deposits:
  'n' - deposit a nickel
  'd' - deposit a dime
  'q' - deposit a quarter
  'o' - deposit a one dollar bill
  'f' - deposit a five dollar bill
  'c' - cancel the purchase

Payment due: 35 cents
Indicate your deposit: q
Payment due: 10 cents
Indicate your deposit: d

Please take the change below.
  No change due.

Stock contains:
   0 nickels
   1 dimes
   1 quarters
   4 ones
   5 fives

Enter the purchase price (xx.xx) or `q' to quit: .35

Menu for deposits:
  'n' - deposit a nickel
  'd' - deposit a dime
  'q' - deposit a quarter
  'o' - deposit a one dollar bill
  'f' - deposit a five dollar bill
  'c' - cancel the purchase

Payment due: 35 cents
Indicate your deposit: q
Payment due: 10 cents
Indicate your deposit: q

Please take the change below.
   1 dimes
Machine is out of change.
See store manager for remaining refund.
Amount due is: 5 cents

Stock contains:
   0 nickels
   0 dimes
   3 quarters
   4 ones
   5 fives

Enter the purchase price (xx.xx) or `q' to quit: q

Total: 29 dollars and 75 cents
```

Sample Exceeding Interaction (user inputs shown in red):

```
Welcome to the vending machine change maker program
Change maker initialized.
Stock contains:
   25 nickels
   25 dimes
   25 quarters
   0 ones
   0 fives

Enter the purchase price (xx.xx) or `q' to quit: aaa
Invalid purchase price. Try again

Enter the purchase price (xx.xx) or `q' to quit: a.4
Invalid purchase price. Try again

Enter the purchase price (xx.xx) or `q' to quit: 4.a
Invalid purchase price. Try again

Enter the purchase price (xx.xx) or `q' to quit: -5
Illegal price: Must be a non-negative multiple of 5 cents.

Enter the purchase price (xx.xx) or `q' to quit: .
Invalid purchase price. Try again

Enter the purchase price (xx.xx) or `q' to quit: -5.W
Invalid purchase price. Try again

Enter the purchase price (xx.xx) or `q' to quit: q

Total: 10 dollars and 0 cents
```