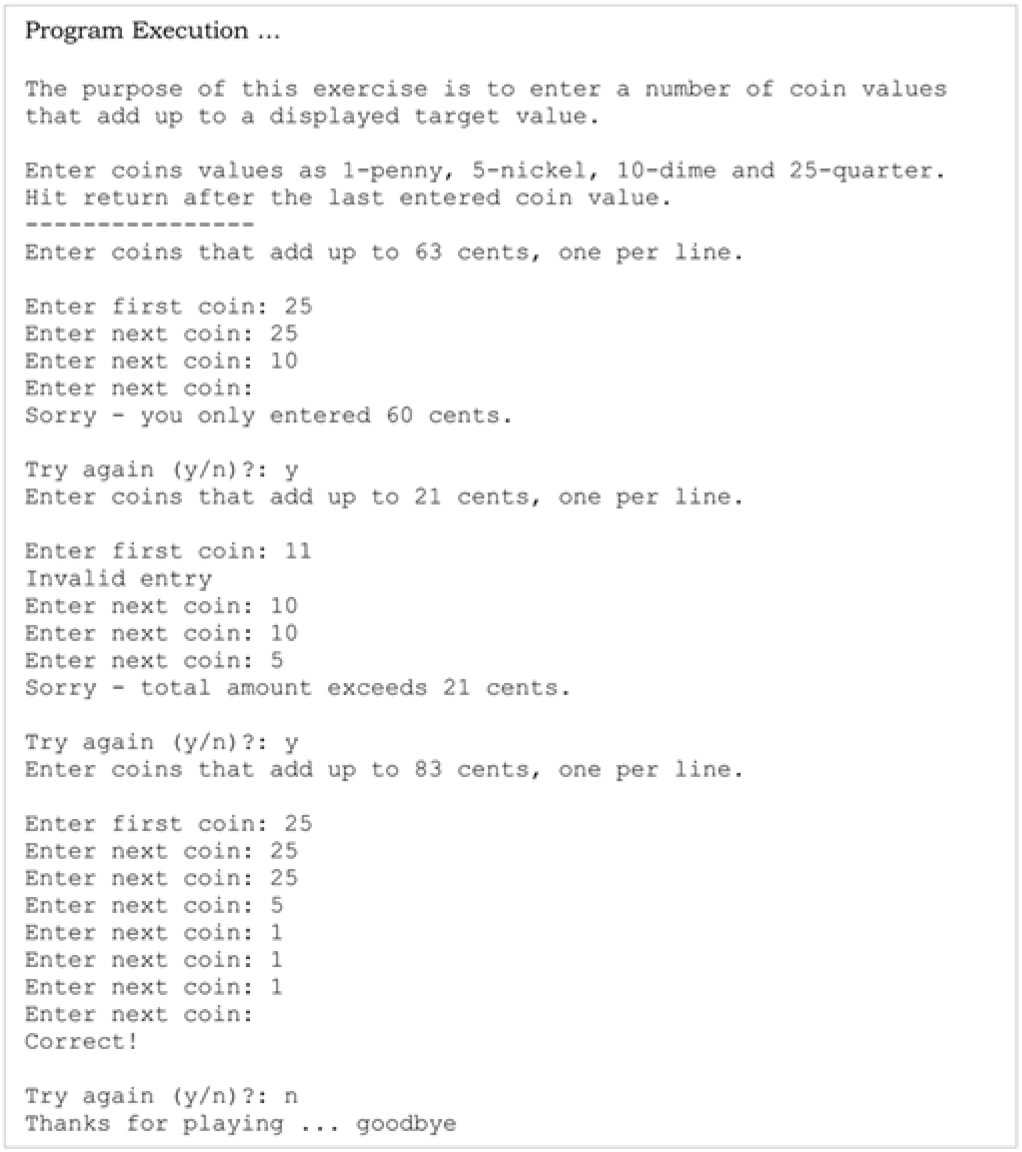
Coin Change Exercise Program

The Python program on the next few pages implements an exercise for children learning to count change. It displays a random value between 1 and 99 cents, and asks the user to enter a set of coins that sums exactly to the amount shown. The program utilizes the following programming features:

- ➤ while loop
- ➤ if statement
- ➤ Boolean flag
- ➤ random number generator



Program Execution ...

```
The purpose of this exercise is to enter a number of coin values
that add up to a displayed target value.

Enter coins values as 1-penny, 5-nickel, 10-dime and 25-quarter.
Hit return after the last entered coin value.
----------------
Enter coins that add up to 63 cents, one per line.

Enter first coin: 25
Enter next coin: 25
Enter next coin: 10
Enter next coin:
Sorry - you only entered 60 cents.

Try again (y/n)?: y
Enter coins that add up to 21 cents, one per line.

Enter first coin: 11
Invalid entry
Enter next coin: 10
Enter next coin: 10
Enter next coin: 5
Sorry - total amount exceeds 21 cents.

Try again (y/n)?: y
Enter coins that add up to 83 cents, one per line.

Enter first coin: 25
Enter next coin: 25
Enter next coin: 25
Enter next coin: 5
Enter next coin: 1
Enter next coin: 1
Enter next coin: 1
Enter next coin:
Correct!

Try again (y/n)?: n
Thanks for playing ... goodbye
```

Task: In IDLE, open a new project and save at as CoinChange_yourLastName. Copy the code from the sample on the next page. Test and revise the program, as needed.

```python
# Coin Change Exercise Program

import random

# program greeting
print('The purpose of this exercise is to enter a number of coin values')
print('that add up to to a displayed target value.\n')
print('Enter coins values as 1-penny, 5-nickel, 10-dime and 25-quarter')
print("Hit return after the last entered coin value.")
print('----------------')

# init
terminate = False
empty_str = ''

# start game
while not terminate:
    amount = random.randint(1,99)
    print('Enter coins that add up to', amount, 'cents, one per line.\n')
    game_over = False
    total = 0

    while not game_over:
        valid_entry = False

        while not valid_entry:
            if total == 0:
                entry = input('Enter first coin: ')
            else:
                entry = input('Enter next coin: ')

            if entry in (empty_str,'1','5','10','25'):
                valid_entry = True
            else:
                print('Invalid entry')

        if entry == empty_str:
            if total == amount:
                print('Correct!')
            else:
                print('Sorry - you only entered', total, 'cents.')

            game_over = True
        else:
            total = total + int(entry)
            if total > amount:
                print('Sorry - total amount exceeds', amount, 'cents.')
                game_over = True

        if game_over:
            entry = input('\nTry again (y/n)?: ')

            if entry == 'n':
                terminate = True

print('Thanks for playing ... goodbye')
```

Notes:

On line 3, the `random` module is imported for use of function `randint`. This function is called (on line 18) to randomly generate a coin value for the user to match, stored in variable `amount`. Lines 6–10 provide the program greeting. On line 13 variable `terminate` is initialized to `False`, used to control when the main loop (and thus the program) terminates. On line 14, `empty_str` is initialized to the empty string literal `''`, used to determine when the user has entered an empty line to end the coin entries. These two variables need only be initialized once, and therefore are assigned before the main while loop.

The game begins on line 17. Since Boolean flag `terminate` is initialized to `False`, the while loop is executed. Besides variable `amount, game_over` is initialized to `False`, and `total` is initialized to `0`. Variable `game_over` serves as another Boolean flag to determine if the current game is to continue or not. The coin entry ends if either the user enters a blank line (indicat- ing that they are done entering coins) in which case the result is displayed and `game_over` is set to `True` (line 37–43), or if the total amount accumulated exceeds the total amount to be matched (on line 45–48).

At the top of the while loop, a third Boolean flag is used, `valid_entry`. The value of this flag determines whether the user should be prompted again because of an invalid input—a value other than `'1', '5', '10'`, or `'25'`, or the empty string. Note the use of the membership operator `in` (line 32). Thus, once the user inputs an appropriate value, `valid_entry` is set to `True` (line 33)—otherwise, the message `'Invalid entry'` is displayed and `valid_entry` remains `False`, causing the loop to execute again. The list of valid entered values on line 32 includes variable `empty_str` since this is the value input when the user hits return to terminate their entry of coin values. When the empty string is found (line 37), the total coin value entered in variable `total` is compared with variable amount (the amount to be matched). If equal, the message `'Correct!'` is displayed (line 39)—otherwise, a message is displayed indicating how much they entered. This amount is always less than the required amount, since whenever variable `total` exceeds `amount`, the current game ends (lines 46–48).

When line 50 is reached, Boolean flag `game_over` may be either `True` or `False`. It is `True` when the user has indicated that they have entered all their coin values (by hitting return), or if the total of the coin values entered exceeds the value in variable amount - it is `False` otherwise. Therefore, flag variable `game_over` is used to determine whether the user should be prompted to play another game (line 51). If they choose to quit the program when prompted, then Boolean variable `terminate` is set to `True`. This causes the encompassing while loop at line 17 to terminate, leaving only the final “goodbye” message on line 56 to be executed before the program terminates.

Extensions

1. Coin Change Exercise Program: Addition of Half-Dollar Coins

Modify the Coin Change Exercise program to allow for the use of half-dollar coins. Make all necessary changes in the program.

2. Coin Change Exercise Program: Raising the Challenge
Modify the Coin Change Exercise program so that the least possible number of coins must be entered. For example, the least number of coins that total to 43 cents is 6 (one quarter, one dime, one nickel, and three pennies).