This is CS50x

OpenCourseWare

Donate (https://cs50.harvard.edu/donate)

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://orcid.org/0000-0001-5338-2522)
(https://www.quora.com/profile/David-J-Malan)
(https://www.reddit.com/user/davidjmalan)
(https://www.tiktok.com/@davidjmalan) (https://twitter.com/davidjmalan)

Cash

Implement a program that calculates the minimum number of coins required to give a user change.

```
$ python cash.py
Change owed: 0.41
4
```

Specification

- Write, in a file called `cash.py` in `~/pset6/cash/`, a program that first asks the user how much change is owed and then spits out the minimum number of coins with which said change can be made, exactly as you did in Problem Set 1, except that your program this time should be written in Python.
- Use `get_float` from the CS50 Library to get the user's input and `print` to output your answer. Assume that the only coins available are quarters (25¢), dimes (10¢), nickels (5¢), and pennies (1¢).
    - We ask that you use `get_float` so that you can handle dollars and cents, albeit sans dollar sign. In other words, if some customer is owed $9.75 (as in the case where a newspaper costs 25¢ but the customer pays with a $10 bill), assume that your program's input will be `9.75` and not `$9.75` or `975`. However, if some

customer is owed $9 exactly, assume that your program's input will be `9.00` or just `9` but, again, not `$9` or `900`. Of course, by nature of floating-point values, your program will likely work with inputs like `9.0` and `9.000` as well; you need not worry about checking whether the user's input is "formatted" like money should be.

- If the user fails to provide a non-negative value, your program should re-prompt the user for a valid amount again and again until the user complies.
- Incidentally, so that we can automate some tests of your code, we ask that your program's last line of output be only the minimum number of coins possible: an integer followed by a newline.

Usage

Your program should behave per the example below.

```
$ python cash.py
Change owed: 0.41
4
```

Testing

While `check50` is available for this problem, you're encouraged to first test your code on your own for each of the following.

- Run your program as `python cash.py`, and wait for a prompt for input. Type in `0.41` and press enter. Your program should output `4`.
- Run your program as `python cash.py`, and wait for a prompt for input. Type in `0.01` and press enter. Your program should output `1`.
- Run your program as `python cash.py`, and wait for a prompt for input. Type in `0.15` and press enter. Your program should output `2`.
- Run your program as `python cash.py`, and wait for a prompt for input. Type in `1.60` and press enter. Your program should output `7`.
- Run your program as `python cash.py`, and wait for a prompt for input. Type in `23` and press enter. Your program should output `92`.
- Run your program as `python cash.py`, and wait for a prompt for input. Type in `4.2` and press enter. Your program should output `18`.
- Run your program as `python cash.py`, and wait for a prompt for input. Type in `-1` and press enter. Your program should reject this input as invalid, as by re-prompting the user to type in another number.
- Run your program as `python cash.py`, and wait for a prompt for input. Type in `foo` and press enter. Your program should reject this input as invalid, as by re-prompting the user to type in another number.
- Run your program as `python cash.py`, and wait for a prompt for input. Do not type

- Run your program as `python cash.py`, and wait for a prompt for input. Do not type anything, and press enter. Your program should reject this input as invalid, as by re-prompting the user to type in another number.

Execute the below to evaluate the correctness of your code using `check50`. But be sure to compile and test it yourself as well!

```
check50 cs50/problems/2021/x/sentimental/cash
```

Execute the below to evaluate the style of your code using `style50`.

```
style50 cash.py
```

This problem will be graded only along the axes of correctness and style.

How to Submit

Execute the below, logging in with your GitHub username and password when prompted. For security, you'll see asterisks (`*`) instead of the actual characters in your password.

```
submit50 cs50/problems/2021/x/sentimental/cash
```