CS50's Introduction to Programming with Python

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://www.reddit.com/user/davidjmalan)
(https://www.threads.net/@davidjmalan) (https://twitter.com/davidjmalan)

Back to the Bank

In a file called `bank.py`, reimplement Home Federal Savings Bank from Problem Set 1, restructuring your code per the below, wherein `value` expects a `str` as input and returns an `int`, namely `0` if that `str` starts with "hello", `20` if that `str` starts with an "h" (but not "hello"), or `100` otherwise, treating the `str` case-insensitively. You can assume that the string passed to the `value` function will not contain any leading spaces. Only `main` should call `print`.

```
def main():
    ...


def value(greeting):
    ...


if __name__ == "__main__":
    main()
```

Then, in a file called `test_bank.py`, implement **three or more** functions that collectively test your implementation of `value` thoroughly, each of whose names should begin with `test_` so that you can execute your tests with:

```
pytest test_bank.py
```

▼ **Hints**

- Be sure to include

```
import bank
```

or

```
from bank import value
```

atop `test_bank.py` so that you can call `value` in your tests.

- Take care to `return`, not `print`, an `int` in `value`. Only `main` should call `print`.

Before You Begin

Log into cs50.dev (https://cs50.dev/), click on your terminal window, and execute `cd` by itself. You should find that your terminal window's prompt resembles the below:

```
$
```

Next execute

```
mkdir test_bank
```

to make a folder called `test_bank` in your codespace.

Then execute

```
cd test_bank
```

to change directories into that folder. You should now see your terminal prompt as `test_bank/ $`. You can now execute

```
code test_bank.py
```

to make a file called `test_bank.py` where you'll write your tests.

How to Test

To test your tests, run `pytest test_bank.py`. Be sure you have a copy of a `bank.py` file in the same folder. Try to use correct and incorrect versions of `bank.py` to determine how well your tests spot errors:

- Ensure you have a correct version of `bank.py`. Run your tests by executing `pytest test_bank.py`. `pytest` should show that all of your tests have passed.
- Modify the correct version of `bank.py`, changing the values provided for each greeting. Your program might, for example, mistakenly provide $100 to a customer greeted by

> "Hello" and $0 to a customer greeted with "What's up"! Now, run your tests by executing `pytest test_bank.py`. `pytest` should show that at least one of your tests has failed.

You can execute the below to check your tests using `check50`, a program CS50 will use to test your code when you submit. (Now there are tests to test your tests!). Be sure to test your tests yourself and determine which tests are needed to ensure `bank.py` is checked thoroughly.

```
check50 cs50/problems/2022/python/tests/bank
```

Green smilies mean your program has passed a test! Red frownies will indicate your program output something unexpected. Visit the URL that `check50` outputs to see the input `check50` handed to your program, what output it expected, and what output your program actually gave.

How to Submit

In your terminal, execute the below to submit your work.

```
submit50 cs50/problems/2022/python/tests/bank
```