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://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)

Refueling

> Note that **Fuel Gauge** from **Problem Set 3** used to specify "less than 1%" and "more than 99%." It now specifies "1% or less" and "99% or more," respectively.

In a file called `fuel.py`, reimplement Fuel Gauge from Problem Set 3, restructuring your code per the below, wherein:

- `convert` expects a `str` in `X/Y` format as input, wherein each of `X` and `Y` is an integer, and returns that fraction as a

percentage, an `int` between `0` and `100`, inclusive. If `X` and/or `Y` is not an integer, then `convert` should raise a `ValueError`. If `Y` is `0`, then `convert` should raise a `ZeroDivisionError`.

- `gauge` expects an `int` and returns a `str` that is:
  - `"E"` if that `int` is less than or equal to `1`,
  - `"F"` if that `int` is greater than or equal to `99`,
  - and `"Z%"` otherwise, wherein `Z` is that same `int`.

```
def main():
    ...


def convert(fraction):
    ...


def gauge(percentage):
    ...


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

Then, in a file called `test_fuel.py`, implement **two or more** functions that collectively test your implementations of `convert` and `gauge` thoroughly, each of whose names should begin with `test_` so that you can execute your tests with:

```
pytest test_fuel.py
```

▼ Hints

- Be sure to include

```
import fuel
```

or

```
from fuel import convert, gauge
```

atop `test_fuel.py` so that you can call `convert` and `gauge` in your tests.

- Take care to `return`, not `print`, an `int` in `convert` and a `str` in `gauge`. Only `main` should call `print`.
- Note that you can check with `pytest` whether a function has raised an exception, per docs.pytest.org/en/latest/how-to/assert.html#assertions-about-expected-exceptions (https://docs.pytest.org/en/latest/how-to/assert.html#assertions-about-expected-exceptions).

Before You Begin

Log into code.cs50.io (https://code.cs50.io/), 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_fuel
```

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

Then execute

```
cd test_fuel
```

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

```
code test_fuel.py
```

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

How to Test

To test your tests, run `pytest test_fuel.py`. Try to use correct and incorrect versions of `fuel.py` to determine how well your tests spot errors:

- Ensure you have a correct version of `fuel.py`. Run your tests by executing `pytest test_fuel.py`. `pytest` should show that all of your tests have passed.
- Modify the correct version of `fuel.py`, changing the return values of `convert`. Your program might, for example, mistakenly return a `str` instead of an `int`. Run your tests by executing `pytest test_fuel.py`. `pytest` should show that at least one of your tests has failed.
- Similarly, modify the correct version of `fuel.py`, changing the return values of `gauge`. Your program might, for example, mistakenly omit a `%` in the resulting `str`. Run your tests by executing `pytest test_fuel.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 `fuel.py` is checked thoroughly.

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

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/fuel
```