CS50's Introduction to Programming with Python

OpenCourseWare

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

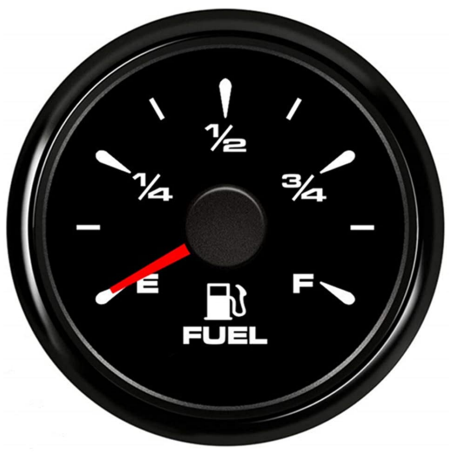
David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan) (https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/) (https://orcid.org/0000-0001-5338-2522) (https://www.quora.com/profile/David-J-Malan) (https://www.reddit.com/user/davidjmalan)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

Fuel Gauge

Fuel gauges indicate, often with fractions, just how much fuel is in a tank. For instance 1/4 indicates that a tank is 25% full, 1/2 indicates that a tank is 50% full, and 3/4 indicates that a tank is 75% full.

In a file called `fuel.py`, implement a program that prompts the user for a fraction, formatted as `X/Y`, wherein each of `X` and `Y` is an integer, and then outputs, as a percentage rounded to the nearest integer, how much fuel is in the tank. If, though, less than 1% remains, output `E` instead to indicate that the tank is empty. And if more than 99% remains, output `F` instead to indicate that the tank is full.

If, though, `X` or `Y` is not an integer, `X` is greater than `Y`, or `Y` is `0`, instead prompt the user again. (It is not necessary for `Y` to be `4`.) Be sure to catch any exceptions like `ValueError` (https://docs.python.org/3/library/exceptions.html#ValueError) or `ZeroDivisionError` (https://docs.python.org/3/library/exceptions.html#ZeroDivisionError).



Source: amazon.com/dp/B09C4FL56G (https://www.amazon.com/dp/B09C4FL56G)

▼ **Hints**

Note that you can handle two exceptions separately with code like:

```
try:
    ...
except ValueError:
    ...
except ZeroDivisionError:
    ...
```

Or you can handle two exceptions together with code like:

```
try:
    ...
except (ValueError, ZeroDivisionError):
    ...
```

Demo

```
25%
$ python fuel.py
Fraction: 1/2
50%
$ python fuel.py
Fraction: 3/4
75%
$ python fuel.py
Fraction: 4/4
F
$ python fuel.py
Fraction:
```

Recorded with asciinema

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 fuel
```

to make a folder called `fuel` into your codespace.

Then execute

```
cd fuel
```

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

```
code fuel.py
```

to make a file called `fuel.py` where you'll write your program.

How to Test

Here's how to test your code manually:

- Run your program with `python fuel.py`. Type `3/4` and press Enter. Your program should output:

  ```
  75%
  ```

- Run your program with `python fuel.py`. Type `1/4` and press Enter. Your program should output:

  ```
  25%
  ```

- Run your program with `python fuel.py`. Type `4/4` and press Enter. Your program should output:

  ```
  F
  ```

- Run your program with `python fuel.py`. Type `0/4` and press Enter. Your program should output:

  ```
  E
  ```

- Run your program with `python fuel.py`. Type `4/0` and press Enter. Your program should handle a `ZeroDivisionError` (https://docs.python.org/3/library/exceptions.html#ZeroDivisionError) and prompt the user again.
- Run your program with `python fuel.py`. Type `three/four` and press Enter. Your program should handle a `ValueError` (https://docs.python.org/3/library/exceptions.html#ValueError) and prompt the user again.
- Run your program with `python fuel.py`. Type `1.5/3` and press Enter. Your program should handle a `ValueError` (https://docs.python.org/3/library/exceptions.html#ValueError) and prompt the user again.
- Run your program with `python fuel.py`. Type `5/4` and press Enter. Your program should prompt the user again.

You can execute the below to check your code using `check50`, a program that CS50 will use to test your code when you submit. But be sure to test it yourself as well!

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