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)

Math Interpreter

Python already supports math, whereby *you* can write code to add, subtract, multiply, or divide values and even variables. But let's write a program that enables *users* to do math, even without knowing Python.

In a file called `interpreter.py`, implement a program that prompts the user for an arithmetic expression and then calculates and outputs the result as a floating-point value formatted to one decimal place. Assume that the user's input will be formatted as `x y z`, with one space between `x` and `y` and one space between `y` and `z`, wherein:

- `x` is an integer
- `y` is `+`, `-`, `*`, or `/`
- `z` is an integer

For instance, if the user inputs `1 + 1`, your program should output `2.0`. Assume that, if `y` is `/`, then `z` will not be `0`.

Note that, just as `python` itself is an interpreter for Python, so will your `interpreter.py` be an interpreter for math!

▼ **Hints**

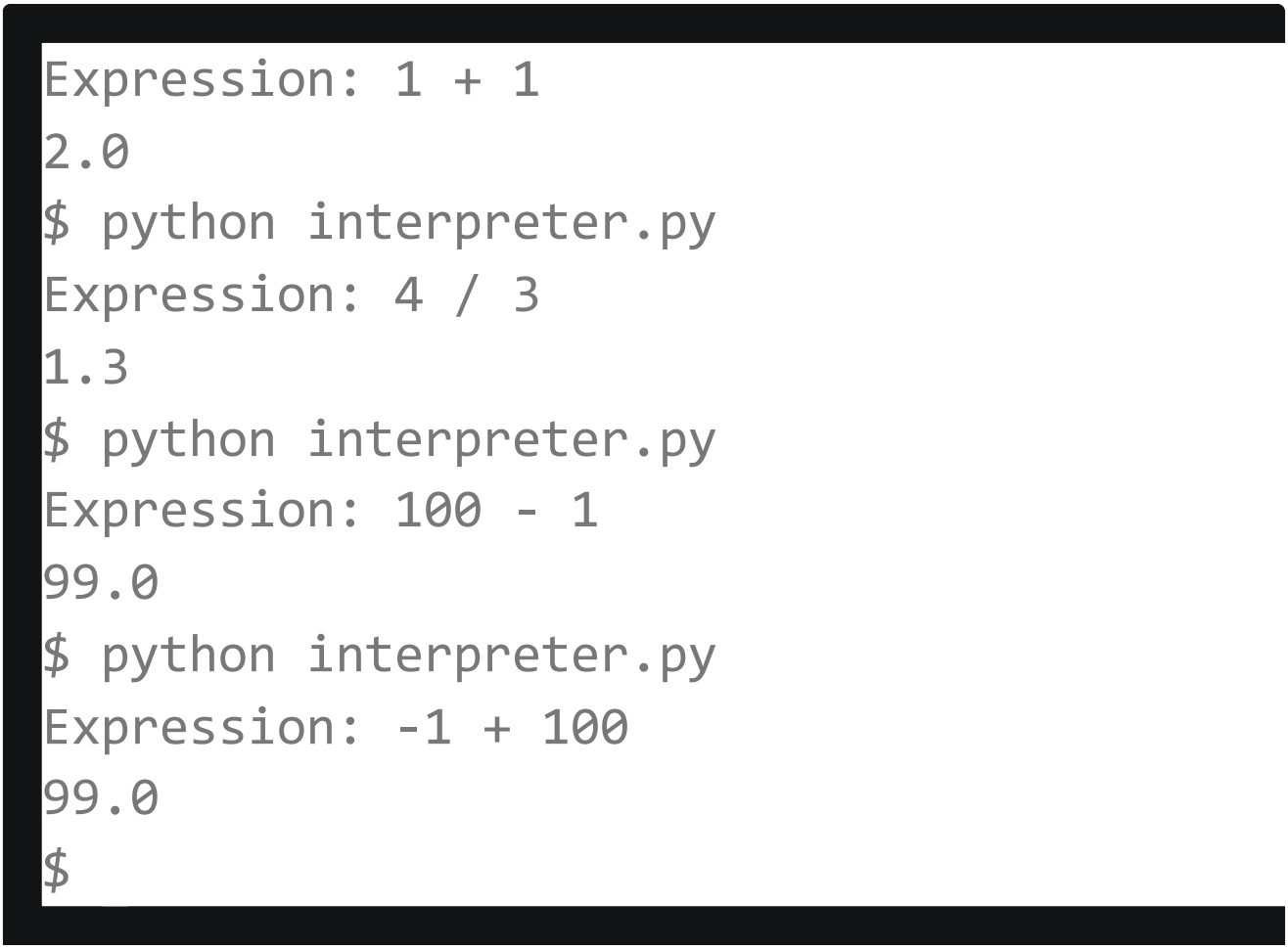
Recall that a `str` comes with quite a few methods, per docs.python.org/3/library/stdtypes.html#string-methods

(https://docs.python.org/3/library/stdtypes.html#string-methods), including `split`, which separates a `str` into a sequence of values, all of which can be assigned to variables at once. For instance, if `expression` is a `str` like `1 + 1`, then

```
x, y, z = expression.split(" ")
```

will assign `1` to `x`, `+` to `y`, and `1` to `z`.

Demo



Recorded with asciinema

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 interpreter
```

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

Then execute

```
cd interpreter
```

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

```
code interpreter.py
```

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

How to Test

Here's how to test your code manually:

- Run your program with `python interpreter.py`. Type `1 + 1` and press Enter. Your program should output:

  ```
  2.0
  ```

- Run your program with `python interpreter.py`. Type `2 - 3` and press Enter. Your program should output:

  ```
  -1.0
  ```

- Run your program with `python interpreter.py`. Type `2 * 2` and press Enter. Your program should output

  ```
  4.0
  ```

- Run your program with `python interpreter.py`. Type `50 / 5` and press Enter. Your program should output

  ```
  10.0
  ```

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/interpreter
```

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/interpreter
```