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)

Lines of Code

One way to measure the complexity of a program is to count its number of lines of code (https://en.wikipedia.org/wiki/Source_lines_of_code) (LOC), excluding blank lines and comments. For instance, a program like

```
# Say hello

name = input("What's your name? ")
print(f"hello, {name}")
```

has just two lines of code, not four, since its first line is a comment, and its second line is blank (i.e., just whitespace). That's not that many, so odds are the program isn't that complex. Of course, just because a program (or even function) has more lines of code than another doesn't necessarily mean it's more complex. For instance, a function like

```
def is_even(n):
    if n % 2 == 0:
        return True
    else:
        return False
```

isn't really twice as complex as a function like

```
def is_even(n):
    return n % 2 == 0
```

even though the former has (more than) twice as many lines of code. In fact, the former might arguably be simpler if it's easier to read! So lines of code should be taken with a grain of salt (https://en.wikipedia.org/wiki/Grain_of_salt).

Even so, in a file called `lines.py`, implement a program that expects exactly one command-line argument, the name (or path) of a Python file, and outputs the number of lines of code in that file, excluding comments and blank lines. If the user does not specify exactly one command-line argument, or if the specified file's name does not end in `.py`, or if the specified file does not exist, the program should instead exit via `sys.exit`.

Assume that any line that starts with `#`, optionally preceded by whitespace, is a comment. (A docstring (https://peps.python.org/pep-0257/) should not be considered a comment.) Assume that any line that only contains whitespace is blank.

▼ 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 `lstrip` and `startswith`.
- Note that `open` can `raise` a `FileNotFoundError`, per docs.python.org/3/library/exceptions.html#FileNotFoundError

(https://docs.python.org/3/library/exceptions.html#FileNotFoundError).

- You might find it helpful to test your program on, e.g., some of Week 6's source code (https://cdn.cs50.net/python/2022/x/lectures/6/src6/) as well as on programs of your own.

Demo

```
$ python lines.py
Too few command-line arguments
$ python lines.py foo
Not a Python file
$ python lines.py foo bar
Too many command-line arguments
$ python lines.py foo.py
File does not exist
$ python lines.py hello.py
2
$
```

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 lines
```

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

Then execute

```
cd lines
```

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

```
code lines.py
```

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

How to Test

Here's how to test your code manually:

- Run your program with `python lines.py`. Your program should exit with `sys.exit` and provide an error message:

  ```
  Too few command-line arguments
  ```

- Create two python programs, `hello.py` and `goodbye.py`. Run `python lines.py hello.py goodbye.py`. Your program

should exit with `sys.exit` and provide an error message:

```
Too many command-line arguments
```

- Create a text file called `invalid_extension.txt`. Run your program with `python lines.py invalid_extension.txt`. Your program should exit with `sys.exit` and provide an error message:

```
Not a Python file
```

- Run your program with `python lines.py non_existent_file.py`. Assuming `non_existent_file.py` doesn't exist, your program should exit with `sys.exit` and provide an error message:

```
File does not exist
```

- Create additional python programs which vary in complexity: create some with comments, some docstrings, and some whitespace. For each of these files run `python lines.py FILENAME` where `FILENAME` is the name of the file. `lines.py` should output the number of lines, excluding comments and whitespace, present in the given file.

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/lines
```

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/lines
```