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)

Scourgify

> "Ah, well," said Tonks, slamming the trunk's lid shut, "at least it's all in. That could do with a bit of cleaning, too." She pointed her wand at Hedwig's cage. "Scourgify (https://harrypotter.fandom.com/wiki/Scouring_Charm)." A few feathers and droppings vanished.
>
> — *Harry Potter and the Order of the Phoenix*

Data, too, often needs to be "cleaned," as by reformatting it, so that values are in a consistent, if not more convenient, format. Consider, for instance, this CSV file of students, before.csv, below:

```
name,house
"Abbott, Hannah",Hufflepuff
"Bell, Katie",Gryffindor
"Bones, Susan",Hufflepuff
"Boot, Terry",Ravenclaw
"Brown, Lavender",Gryffindor
"Bulstrode, Millicent",Slytherin
"Chang, Cho",Ravenclaw
"Clearwater, Penelope",Ravenclaw
"Crabbe, Vincent",Slytherin
"Creevey, Colin",Gryffindor
"Creevey, Dennis",Gryffindor
"Diggory, Cedric",Hufflepuff
"Edgecombe, Marietta",Ravenclaw
"Finch-Fletchley, Justin",Hufflepuff
"Finnigan, Seamus",Gryffindor
"Goldstein, Anthony",Ravenclaw
```

Source: en.wikipedia.org/wiki/List_of_Harry_Potter_characters (https://en.wikipedia.org/wiki/List_of_Harry_Potter_characters)

Even though each "row" in the file has three values (last name, first name, and house), the first two are combined into one "column" (name), escaped with double quotes, with last name and first name separated by a comma and space. Not ideal if Hogwarts (https://en.wikipedia.org/wiki/Hogwarts) wants to send a form letter (https://en.wikipedia.org/wiki/Form_letter) to each student, as via mail merge (https://en.wikipedia.org/wiki/Mail_merge), since it'd be strange to start a letter with:

> Dear Potter, Harry,

Rather than with, for instance:

> Dear Harry,

In a file called `scourgify.py`, implement a program that:

- Expects the user to provide two command-line arguments:

    - the name of an existing CSV file to read as input, whose columns are assumed to be, in order, `name` and `house`, and
    - the name of a new CSV to write as output, whose columns should be, in order, `first`, `last`, and `house`.
- Converts that input to that output, splitting each `name` into a `first` name and `last` name. Assume that each student will have both a first name and last name.

If the user does not provide exactly two command-line arguments, or if the first cannot be read, the program should exit via `sys.exit` with an error message.

▼ **Hints**

- Note that `csv` module comes with quite a few methods, per docs.python.org/3/library/csv.html (https://docs.python.org/3/library/csv.html), among which are `DictReader`, per docs.python.org/3/library/csv.html#csv.DictReader (https://docs.python.org/3/library/csv.html#csv.DictReader) and `DictWriter`, per docs.python.org/3/library/csv.html#csv.DictWriter (https://docs.python.org/3/library/csv.html#csv.DictWriter).
- Note that you can tell a `DictWriter` to write its `fieldnames` to a file using `writeheader` with no arguments, per docs.python.org/3/library/csv.html#csv.DictWriter.writeheader (https://docs.python.org/3/library/csv.html#csv.DictWriter.writeheader).

Demo

```
$ python scourgify.py
Too few command-line arguments
$ python scourgify.py 1.csv
Too few command-line arguments
$ python scourgify.py 1.csv 2.csv 3.csv
Too many command-line arguments
$ python scourgify.py 1.csv 2.csv
Could not read 1.csv
$ python scourgify.py before.csv after.csv
$
```

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 scourgify
```

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

Then execute

```
cd scourgify
```

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

```
code scourgify.py
```

to make a file called `scourgify.py` where you'll write your program. Be sure to run

```
wget https://cs50.harvard.edu/python/2022/psets/6/scourgify/before.csv
```

to download before.csv into your folder.

How to Test

Here's how to test your code manually:

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

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

- Create empty files `1.csv`, `2.csv`, and `3.csv`. Run your program with `python scourgify.py 1.csv 2.csv 3.csv`. Your program should output:

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

- Run your program with `python scourgify.py invalid_file.csv output.csv`. Assuming `invalid_file.csv` doesn't exist, your program should exit using `sys.exit` and provide an error message:

```
Could not read invalid_file.csv
```

- Run your program with `python scourgify.py before.csv after.csv`. Assuming `before.csv` exists, your program should create a new file, `after.csv`, whose columns should be, in order, `first`, `last`, and `house`.

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/scourgify
```

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/scourgify
```