CS50's Introduction to Programming with Python

OpenCourseWare

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

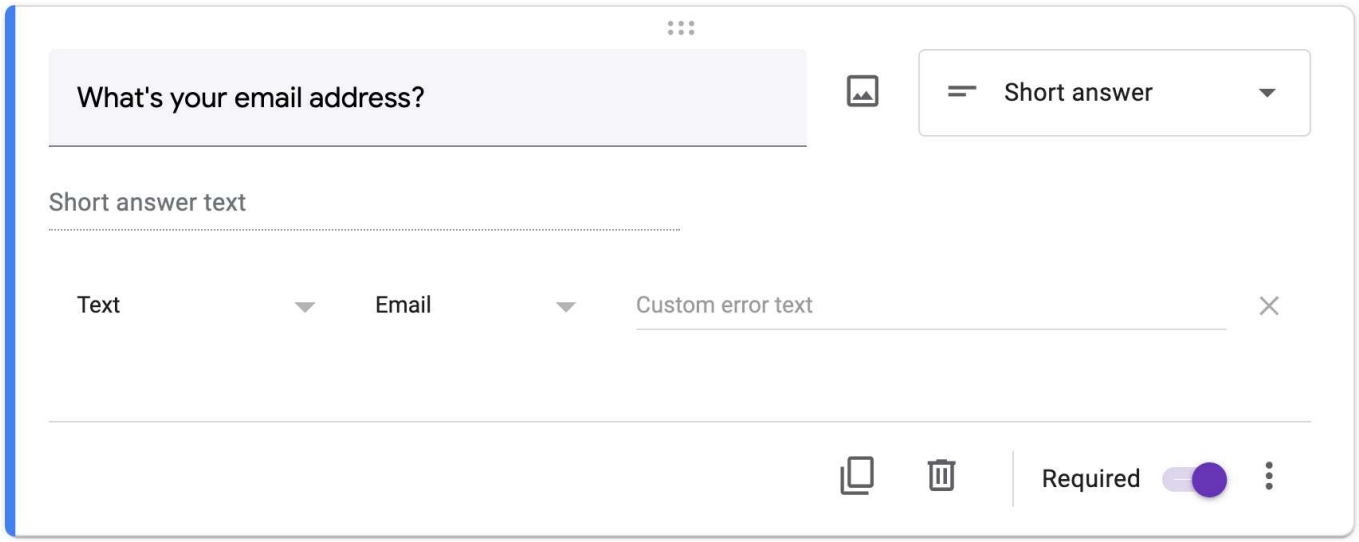
David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://www.reddit.com/user/davidjmalan) (https://www.threads.net/@davidjmalan)
(https://twitter.com/davidjmalan)

Response Validation

When creating a Google Form (https://www.google.com/forms/about/) that prompts users for a short answer (or paragraph), it's possible to enable response validation (https://support.google.com/docs/answer/3378864) and require that the user's input match a regular expression (https://support.google.com/a/answer/1371415). For instance, you could require that a user input an email address with a regex like this one (https://html.spec.whatwg.org/multipage/input.html#valid-e-mail-address):

```
^[a-zA-Z0-9.!#$%&'*+\/=?^_`{|}~-]+@[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?
(?:\.[a-zA-Z0-9](?:[a-zA-Z0-9-]{0,61}[a-zA-Z0-9])?)*$
```

Or you could more easily use Google's built-in support for validating an email address, per the screenshot below, much like you could use a library in your own code:



In a file called `response.py`, using either validator-collection (https://pypi.org/project/validator-collection/) or validators (https://github.com/kvesteri/validators) from PyPI, implement a program that prompts the user for an email address via `input` and then prints `Valid` or `Invalid`, respectively, if the input is a syntatically valid email address. You may not use `re`. And do not validate whether the email address's domain name actually exists.

▶ **Hints**

Demo

```
$ python response.py
What's your email address? malan
Invalid
$ python response.py
What's your email address? malan at harvard dot edu
Invalid
$ python response.py
What's your email address? malan@harvar
```

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 response
```

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

Then execute

```
cd response
```

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

```
code response.py
```

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

How to Test

Here's how to test your code manually:

- Run your program with `python response.py`. Ensure your program prompts you for an email, then type `malan@harvard.edu`, followed by Enter. Your program should output `Valid`.
- Run your program with `python response.py`. Type your own email, followed by Enter. Your program should output `Valid`.
- Run your program with `python response.py`. Type `malan@@@harvard.edu`, followed by Enter. Your program should output `Invalid`.
- Run your program with `python response.py`. Mistype your own email, including an extra `.` before `.com`, for example. Press enter and your program should output `Invalid`.

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/response
```

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/response
```