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)

NUMB3RS

In Season 5, Episode 23 of NUMB3RS (https://en.wikipedia.org/wiki/Numbers_(TV_series)), a supposed IP address (https://en.wikipedia.org/wiki/IP_address) appears on screen, `275.3.6.28`, which isn’t actually a valid IPv4 (https://en.wikipedia.org/wiki/IPv4) (or IPv6 (https://en.wikipedia.org/wiki/IPv6)) address.

An IPv4 address is a numeric identifier that a device (or, on TV, hacker) uses to communicate on the internet, akin to a postal address in the real world, typically formatted in dot-decimal notation (https://en.wikipedia.org/wiki/Dot-decimal_notation) as `#.#.#.#`. But each `#` should be a number between `0` and `255`, inclusive. Suffice it to say `275` is not in that range! If only NUMB3RS had validated the address in that scene!

In a file called `numb3rs.py`, implement a function called `validate` that expects an IPv4 address as input as a `str` and then returns `True` or `False`, respectively, if that input is a valid IPv4 address or not.

Structure `numb3rs.py` as follows, wherein you're welcome to modify `main` and/or implement other functions as you see fit, but you may not import any other libraries. You're welcome, but not required, to use `re` and/or `sys`.

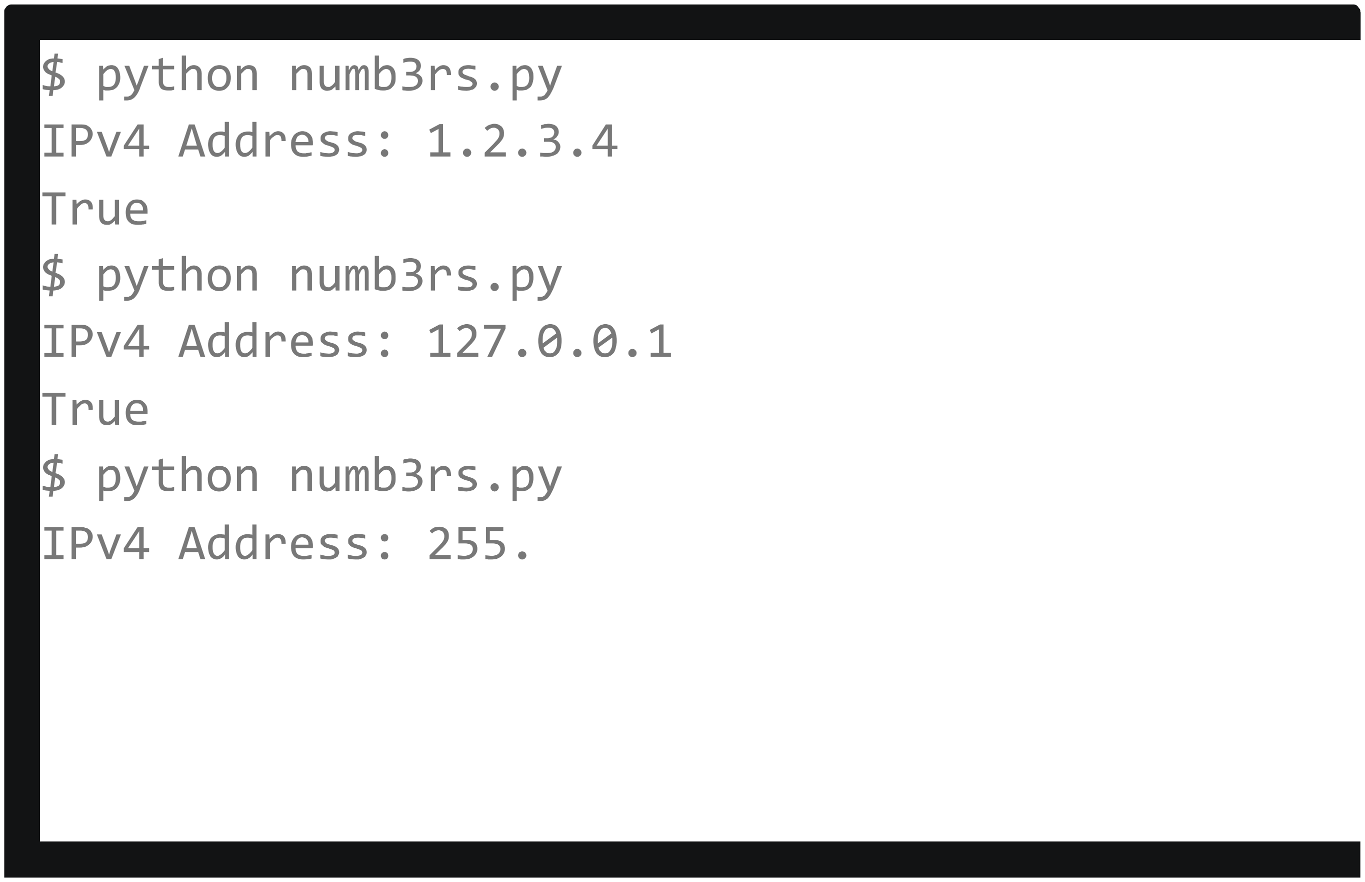
```python
import re
import sys


def main():
    print(validate(input("IPv4 Address: ")))


def validate(ip):
    ...


...


if __name__ == "__main__":
    main()
```

Either before or after you implement `validate` in `numb3rs.py`, additionally implement, in a file called `test_numb3rs.py`, **two or more** functions that collectively test your implementation of `validate` thoroughly, each of whose names should begin with `test_` so that you can execute your tests with:

```
pytest test_numb3rs.py
```

▶ **Hints**

Demo



```
$ python numb3rs.py
IPv4 Address: 1.2.3.4
True
$ python numb3rs.py
IPv4 Address: 127.0.0.1
True
$ python numb3rs.py
IPv4 Address: 255.
```

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 numb3rs
```

to make a folder called `numb3rs` in your codespace.

Then execute

```
cd numb3rs
```

to change directories into that folder. You should now see your terminal prompt as `numb3rs/ $`. You can now execute

```
code numb3rs.py
```

to make a file called `numb3rs.py` where you'll write your program. Be sure to also execute

```
code test_numb3rs.py
```

to create a file called `test_numb3rs.py` where you'll write tests for your program.

How to Test

How to Test `numb3rs.py`

Here's how to test `numb3rs.py` manually:

- Run your program with `python numb3rs.py`. Ensure your program prompts you for an IPv4 address. Type `127.0.0.1`, followed by Enter. Your `validate` function should return `True`.
- Run your program with `python numb3rs.py`. Type `255.255.255.255`, followed by Enter. Your `validate` function should return `True`.
- Run your program with `python numb3rs.py`. Type `512.512.512.512`, followed by Enter. Your `validate` function should return `False`.
- Run your program with `python numb3rs.py`. Type `1.2.3.1000`, followed by Enter. Your `validate` function should return `False`.
- Run your program with `python numb3rs.py`. Type `192.168.001.1`, followed by Enter. Your `validate` function should return `False`.
- Run your program with `python numb3rs.py`. Type `cat`, followed by Enter. Your `validate` function should return `False`.

While leading zeros in IP addresses are technically possible in some contexts, they are generally discouraged due to potential ambiguity. For this problem, treat them as invalid. If you'd like to learn more about IP address formatting standards, see **RFC 3986, Section 7.4 (https://datatracker.ietf.org/doc/html/rfc3986#section-7.4)**.

How to Test `test_numb3rs.py`

To test your tests, run `pytest test_numb3rs.py`. Try to use correct and incorrect versions of `numb3rs.py` to determine how well your tests spot errors:

- Ensure you have a correct version of `numb3rs.py`. Run your tests by executing `pytest test_numb3rs.py`. `pytest` should show that all of your tests have passed.

- Modify the `validate` function in the correct version of `numb3rs.py`. `validate` might, for example, only check whether the first byte of the IPv4 address is valid. Run your tests by executing `pytest test_numb3rs.py`. `pytest` should show that at least one of your tests has failed.
- Again modify the correct version of `numb3rs.py`. `validate` might, for example, mistakenly return `True` when the user inputs an incorrect IPv4 format. Run your tests by executing `pytest test_numb3rs.py`. `pytest` should show that at least one of your tests has failed.

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/numb3rs
```

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/numb3rs
```