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)

Watch on YouTube

It turns out that (most) YouTube videos can be embedded in other websites, just like the above. For instance, if you visit https://youtu.be/xvFZjo5PgG0 (https://youtu.be/xvFZjo5PgG0) on a laptop or desktop, click **Share**, and then click **Embed**, you’ll see HTML (https://en.wikipedia.org/wiki/HTML) (the language in which web pages are written) like the below, which you could then copy into your own website’s source code, wherein `iframe` (https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe) is an HTML “element,” and `src` is one of several HTML "attributes" therein, the value of which, between quotes, is `https://www.youtube.com/embed/xvFZjo5PgG0`.

```
<iframe width="560" height="315"
src="https://www.youtube.com/embed/xvFZjo5PgG0" title="YouTube video player"
frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-
media; gyroscope; picture-in-picture" allowfullscreen></iframe>
```

Because some HTML attributes are optional, you could instead minimally embed just the below.

```
<iframe src="https://www.youtube.com/embed/xvFZjo5PgG0"></iframe>
```

Suppose that you'd like to extract the URLs of YouTube videos that are embedded in pages (e.g., `https://www.youtube.com/embed/xvFZjo5PgG0`), converting them back to shorter, shareable `youtu.be` URLs (e.g., `https://youtu.be/xvFZjo5PgG0`) where they can be watched on YouTube itself.

In a file called `watch.py`, implement a function called `parse` that expects a `str` of HTML as input, extracts any YouTube URL that's the value of a `src` attribute of an `iframe` element therein, and returns its shorter, shareable `youtu.be` equivalent as a `str`. Expect that any such URL will be in one of the formats below. Assume that the value of `src` will be surrounded by double quotes. And assume that the input will contain no more than one such URL. If the input does not contain any such URL at all, return `None`.

- `http://youtube.com/embed/xvFZjo5PgG0`
- `https://youtube.com/embed/xvFZjo5PgG0`
- `https://www.youtube.com/embed/xvFZjo5PgG0`

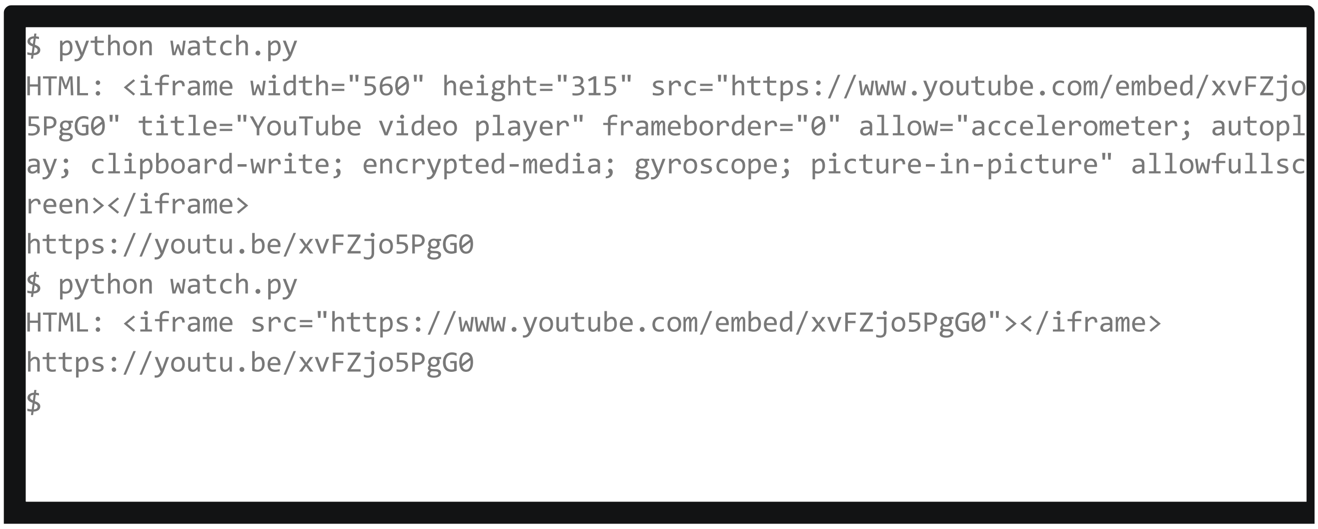
Structure `watch.py` as follows, wherein you're welcome to modify `main` and/or implement other functions as you see fit, but you may not import any other libraries. You're welcome, but not required, to use `re` and/or `sys`.

```
import re
import sys


def main():
    print(parse(input("HTML: ")))


def parse(s):
    ...


...


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

▼ **Hints**

- Recall that the `re` module comes with quite a few functions, per docs.python.org/3/library/re.html (https://docs.python.org/3/library/re.html), including `search`.
- Recall that regular expressions support quite a few special characters, per docs.python.org/3/library/re.html#regular-expression-syntax (https://docs.python.org/3/library/re.html#regular-expression-syntax).
- Because backslashes in regular expressions could be mistaken for escape sequences (like `\n`), best to use Python's raw string notation for regular expression patterns (https://docs.python.org/3/library/re.html#module-re). Just as format strings are prefixed with `f`, so are raw strings prefixed with `r`. For instance, instead of `"harvard\.edu"`, use `r"harvard\.edu"`.
- Note that `re.search`, if passed a pattern with "capturing groups" (i.e., parentheses), returns a "match object," per docs.python.org/3/library/re.html#match-objects (https://docs.python.org/3/library/re.html#match-objects), wherein matches are 1-indexed, which you can access individually with `group`, per docs.python.org/3/library/re.html#re.Match.group (https://docs.python.org/3/library/re.html#re.Match.group), or collectively with `groups`, per docs.python.org/3/library/re.html#re.Match.groups (https://docs.python.org/3/library/re.html#re.Match.groups).
- Note that `*` and `+` are "greedy," insofar as "they match as much text as possible," per docs.python.org/3/library/re.html#regular-expression-syntax (https://docs.python.org/3/library/re.html#regular-expression-syntax). Adding `?` immediately after either, a la `*?` or `+?`, "makes it perform the match in non-greedy or minimal fashion; as few characters as possible will be matched."

Demo



```
$ python watch.py
HTML: <iframe width="560" height="315" src="https://www.youtube.com/embed/xvFZjo
5PgG0" title="YouTube video player" frameborder="0" allow="accelerometer; autopl
ay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" allowfullsc
reen></iframe>
https://youtu.be/xvFZjo5PgG0
$ python watch.py
HTML: <iframe src="https://www.youtube.com/embed/xvFZjo5PgG0"></iframe>
https://youtu.be/xvFZjo5PgG0
$
```

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 watch
```

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

Then execute

```
cd watch
```

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

```
code watch.py
```

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

How to Test

Here's how to test your code manually:

- Run your program with `python watch.py`. Ensure your program prompts you for HTML, then copy/paste the below:

  ```
  <iframe src="http://www.youtube.com/embed/xvFZjo5PgG0"></iframe>
  ```

  Press enter and your program should output `https://youtu.be/xvFZjo5PgG0`. Notice how, though the `src` attribute is prefixed with `http://www.youtube.com/embed/`, the resulting link is prefixed with `https://youtu.be/`.
- Run your program with `python watch.py`. Ensure your program prompts you for HTML, then copy/paste the below:

  ```
  <iframe width="560" height="315" src="https://www.youtube.com/embed/xvFZjo5
  ```

  Press enter and your program should still output `https://youtu.be/xvFZjo5PgG0`.
- Run your program with `python watch.py`. Ensure your program prompts you for HTML, then copy/paste the below:

  ```
  <iframe width="560" height="315" src="https://cs50.harvard.edu/python"></if
  ```

  Press enter and your program should output `None`. Notice how the `src` attribute doesn't point to a YouTube link!

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/watch
```

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/watch
```