What do web servers do? Answer HTTP requests with HTTP responses.

We have already seen and used some communication between servers and clients, using JSON mainly. Today we'll see what other things can a web server send to the client.

Web servers

```
from flask import Flask

app = Flask("name of my app")

app.run()
```

Flask recap

Routes

```
@app.route("/path/<name>")
def handle(name):
    return "hello {}".format(name)
```

Methods

```
@app.route("/tweet/<tweet>", methods = ["POST"])
def handle_tweet(tweet):
    #do fancy stuff with the tweet
    return "whatever"
```

JSON

```
from flask import Flask, jsonify


@app.route("/tweet/<tweet>", methods = ["POST"])
```

serving static files

Why is this useful?

Most of the times we will not go to our webpage by opening an HTML file from our computer. Rather than that, we would go to a internet address and browse from there. For that to happen, files need to be served from the server.

Example1

See `staticfiles.py`

HTML forms are the way we have in HTML to gather data from the client to be sent to the server.

We use the `form` tag to create them:

HTML forms

inputs

There are **a lot** of different input types, to name a few: - `radio`: used to select **one of** between different elements - `checkbox`: used to make a single value selectable. - `text` - `password`: for masked text - `submit`: visible as a button that, on click, submits the form
https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input#Form_%3Cinput%3E_types
When using Flask, it's very simple to handle the form data sent by the client. We can import the `request` object from `flask`, and access to its `form` attribute. It will contain a dictionary with the form data.

Handling HTML forms in Flask

```
from flask import request


@app.route("/submit", methods = ["POST"])
def handle_form_submission():
    user = request.form["username"]
    pass = request.form["password"]

    return "the login data was: {}:{}".format(user, pass)
```

Example 2

see usingforms.py

In the same way Flask can be used to serve files (as we did with HTML and CSS files previously), it can handle file uploads too! First of all, let's see how we can gather files from the client to be sent to the user.

Handling file upload

input type="file"

something we need to consided whenever we want to handle files from the user is adding the multipart encoding to the `<form>` tag:

```
<form enctype="multipart/form-data">
</form>
```

when we make our form encoding `multipart/form-data`, we make the browser allow our `<input type="file">` fields to upload data.

handling file uploads in Flask

In Flask, as with HTML forms, it's very easy to handle file uploads. We just need to access the `files` attribute from the `request` object and use it as a dictionary.

```
from flask import request


@app.route("/handle-upload", methods = ["POST"])
```

Exercises

bluebelt

Create a webpage with an HTML form that gathers:

- a profile picture
- a username

And make those profile pictures accesible later on using the `/profile/<username>.jpg` route.

You'll need to investigate how save the uploaded file to a folder in the server.