# Programming fundamentals with Python

Pepe García

2020-04-20

## Programming fundamentals with Python

https://slides.com/pepegar/pfp-20/live

## Plan for today

Review HTTP

HTTP servers

HTTP Methods

Exercises

Connecting to local HTTP servers

## HTTP Review

HTTP is a request-response protocol. HTTP **clients send requests** and HTTP **servers answer with responses**

## HTTP Methods

Depending on the intention of the request, HTTP describes different methods:

| method | **intention** |
|---|---|
| **GET** | access to a resource |
| **POST** | update a resource |
| **PUT** | create a resource |
| **DELETE** | delete a resource |

## HTTP Methods

Methods are part of the **request**. We have already seen how to use the **GET** method with **requests**:

```
import requests

requests.get("http://resource")
```

## HTTP Methods

The other methods can be used in the same way with **requests**:

```
import requests

requests.get("http://resource")
requests.put("http://resource")
requests.post("http://resource")
requests.delete("http://resource")
```

## HTTP servers

HTTP servers answer to requests from clients. We will be using the **flask** library for creating HTTP servers in Python.

HTTP servers handle **routes** in different ways.

## HTTP servers

```python
from flask import Flask

app = Flask("simplest server")

@app.route("/hello")
def hello():
    return "hello from the web!"

app.run()
```

## HTTP servers

Creating our first HTTP server.

Starting it from the terminal.

Use /anaconda3/bin/python to run it.

## HTTP routes

Our flask server can handle different routes by adding more handlers to it:

```python
@app.route("/hello")
def hello():
    return "hi!"

@app.route("/goodbye")
def hello():
    return "bye!"
```

## HTTP routes

We can also capture part of the path as a variable:

```python
@app.route("/hello/<name>")
def hello(name):
    return "hello " + name
```

## HTTP routes

Capturing segments of the URL

## HTTP methods

One can specify which methods the function handles in the **methods** parameter

```
@app.route("/hello", method=["GET"])
def hello():
    return "hi!"

@app.route("/goodbye", method=["POST"])
def hello():
    return "bye!"
```

## HTTP methods

Create a web server that stores the mood you're in right now.

It can start with happy :)

## Returning JSON

Flask has a **jsonify** function that we can use to convert the data we want to JSON:

```python
from flask import Flask, jsonify


app = Flask("simplest server")


@app.route("/hello/<name>")
def hello(name):
    return jsonify({"message": "hello", "name": name})
```

## Returning JSON

HTTP server to count the number of requests it receives

## Connecting to local HTTP servers

We can use the requests library to connect to local HTTP servers!

```python
import requests


response = requests.get("http://localhost:8080/hello/pepe")
print(response.text)
```

## Connecting to local HTTP servers

Connecting to a local http server

## Exercises

## Exercise 1

Create a HTTP server that can do simple arithmetic tasks.

- **GET /sum/1/2**

returns 3

- **GET /multiply/3/4**

returns 12

## Exercise 1

Create a web server that maintains a list of the books you've read.

You should be able to add and delete individual books, and list all the books you've read.

## Exercise 3

Create a TODO application. It should contain a **client and a server**.

The **server** should allow you to create tasks, complete tasks, and list all tasks.

The **client** should be able to use all operations from the server.

**You'll need to investigate about how to send a body in the request using the requests library.**