Advanced Programming with Python. Session 1

Pepe García

2020-04-20

Advanced Programming with Python. Session 1

Professor

Pepe García

jgarciah@faculty.ie.edu

Ask me anything

About the course

- 10 sessions

About the course

- 10 sessions
- 8 lectures

About the course

- 10 sessions
- 8 lectures
- 1 workgroup assignment

About the course

- 10 sessions
- 8 lectures
- 1 workgroup assignment
- 1 individual assignment

About the course

- 10 sessions
- 8 lectures
- 1 workgroup assignment
- 1 individual assignment
- 1 final exam

Syllabus

Session 1 (today)

Course presentation

Sessions 2 & 3

We will use these two sessions to understand the basics of web servers in Python using Flask. We will learn the following subjects:

- HTTP Request-Response cycle
- HTTP Status codes
- Flask routing
- Rendering JSON

Sessions 4 & 5

In these sessions we will learn how to make our web applications render other things apart of JSON. We will render HTML files, and learn about how templating can simplify the process of presenting and gathering data from the user.

Connecting to Databases

Session 7

Analytical web applications. In this session we will learn about Dash, how can we create analytical web applications with it, and how to integrate it with Flask servers.

Session 8

Python application deployment

Session 9

Workgroup session

Session 10

Final exam

Grading criteria

Criteria\	Score
Class participation	10%
Workgroups	20%
Individual work	50%
Final exam	20%

Joining the new organization

Recap

HTTP

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 servers

```
from flask import Flask

app = Flask("simplest server")

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

app.run()
```

HTTP routes

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

```
@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:

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

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!"
```

Returning JSON

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

```
from flask import Flask, jsonify

app = Flask("hello server")

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

Practice

Create a web server that has an endpoint to which we can call to get a proper salutation. For example, calling to **/hello/Pepe** should return the json **{"message": "hello", "name": "Pepe"}**

HTTP clients

requests library

We can use requests to get an HTTP response as follows:

```
import requests

response = requests.get("url")

data = response.json()
```

Practice

Call your newly created web server **using requests**. Try the call with different parameters.