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

- 15 sessions

About the course

- 15 sessions
- 1 workgroup assignment

About the course

- 15 sessions
- 1 workgroup assignment
- 2 individual assignments

Syllabus

SESSION 1 (FACE TO FACE)

Course presentation. in this session we will introduce the course, the syllabus, materials we're going to use, and grading system. Web development in Python. We will understand what are the latest tools of web development in Python, from libraries for creating web servers, to others helping with the creation of development environments such as pipenv.

SESSION 2 (NON-CLASS LEARNING)

HTTP in Python - introduction

In this session we will learn about how to use HTTP in Python. Students will go over a video recorded by the professor and some written materials to do this Non class Learning.

SESSIONS 3 & 4 (FACE TO FACE)

HTTP in Python

We will use this session 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

SESSION 5 (NON-CLASS LEARNING)

HTML

Web servers can do a lot more than returning JSON. In order to create websites, we need to understand how HTML works, we will use Codecademy for this session.

SESSION 6 (FACE TO FACE)

HTML

In the second session about HTML we will learn more HTML elements and how to implement different UI patterns that we all know from common websites.

SESSION 7 (NON-CLASS LEARNING)

Individual Assignment 1

In this session the professor will send the students an assignemnt to be solved individually. There will be a forum open for asking questions about it.

SESSION 8 (FACE TO FACE)

Web servers - Authentication

In this session we will learn about how to implement authentication in web applications.

SESSION 9 (NON-CLASS LEARNING)

Connecting to databases

In this session we will learn how to make our Python applications connect to databases. The professor will create a video for the students to follow.

SESSION 10 (FACE TO FACE)

Connecting to databases

In this session we will review what we did in the last async session about databases, and learn a bit more about how render data from the database to HTML.

SESSION 11 (NON-CLASS LEARNING)

Group assignment

In this session students will do a group assignment. There will be a forum open in campus so that students can ask questions.

SESSION 12 (FACE TO FACE)

Analytical web applications - Dash

In this session we will introduce a new framework for data oriented web applications, Dash. With Dash we will be able to construct data rich applications in an easy way.

SESSION 13 (FACE TO FACE)

Deployment

In this session we will learn how to deploy our flask applications to the cloud.

SESSION 14 (NON-CLASS LEARNING)

Individual assignment 2

In this session the professor will send the students an assignemnt to be solved individually. There will be a forum open for asking questions about it.

SESSION 15 (FACE TO FACE)

QA session

In this session we will do a whirlwind tour over what we have learned in the course and we will have time to answer questions students may have

Grading criteria

Criteria	Score
Class participation	10%
Workgroups	35%
Individual work	50%

Joining the new organization

Accept this (@pepe, send link to Zoom chat)

https://classroom.github.com/a/MRDwIdta

Checkpoint

HTTP

HTTP

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

HTTP

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

HTTP

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

method	**intention**
GET	read to a resource
POST	update a resource
PUT	create a resource
DELETE	delete a resource

HTTP

Flask
web development,
one drop at a time

We will use `flask` in this course to learn and create web servers in Python.

HTTP

Flask

web development,
one drop at a time

We will use `flask` in this course to learn and create web servers in Python.

`flask` is bundled in Anaconda already, so we don't need to download it.

HTTP - flask

example: simple flask application

how do we use ‘flask‘?

HTTP - flask

```python
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:

```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 methods

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

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

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

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("hello server")


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

HTTP

Exercise

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.