Universidad Autónoma de Madrid

Computer Science Department

Computer Systems Project (PSI)
Assignment - 1

PSI Teaching Staff

Índice

1. Introduction and goal

The main goal of this course is to develop a web application using a professional development framework. Between the different options we have chosen *Django* which has been written using *Python*. *Django* will be the subject of the second assignment. In this first assignment we introduce the *Python* programming language and *Git* (`https://git-scm.com/`), a version control software which will be used within the whole course. As IDE, we recommend *Pycharm* (`https://www.jetbrains.com/es-es/pycharm/`).

2. Work description

2.1. *Git*

A large percentage of all *Git* repositories are hosted on *GitHub*. The first thing you need to do is set up a free user account. Simply visit `https://github.com`, click the "Sign up" button and follow the instructions. Your username must be the first letter of your name followed by your family name. If the username has already been assigned add a number at the username end.

A quick reference for basic Git commands is provided in the appendix. After reading this appendix, perform the following exercise:

EXERCISE 1: Introduction to *Git*

1. Create a private repository called psi_groupNumber_pairNumber_p1.
2. Clone the repository to a local computer.
3. Modify the access properties so that both teammates can modify the repository.
4. A teammate should upload the code necessary to create the program `hello.py` available in *Moodle*.
5. On a different computer the other teammate (using his github username) must: (1) download the repository, (2) modify the file so it writes "Hola Mundo" instead of "Hello World" and (3) upload the modified code to the repository.
6. The first teammate must: (1) update the modified code, (2) create a new file called `hallo.py`, identical to `hello.py` but greeting "Hallo Welt" instead of "Hello World", and (3) upload the changes to the repository.

2.2. *Python*

We assume that most of you are not familiar with the *Python* programming language. This assignment proposes some simple readings and exercises to allow students to acquire a basic knowledge that will be applied during the rest of the course. It is important to note that we focus in a set of basic aspects, but *Python* has additional features. We recommend to review the *Python* language reference (`https://docs.python.org/3/index.html`). Although some of the proposed readings and exercises were created for Python 2, in this course we are going to use Python 3 (in particular, version 3.6.9, `https://docs.python.org/3.6/reference/`). The solution to this lab assignment is almost identical in both versions (see below).

EXERCISE 2: Setup and introduction to *Python*

Check the *Google 's Python class* (`https://developers.google.com/edu/python/`) and focus on the sections entitled *Python Set Up*, *Python Intro* and *Strings*. After executing the examples proposed, implement the exercises described in the files `string1.py` and `string2.py` (available in *Moodle*). Solve these exercises using Python 3 (instead of Python 2). The only changes between versions that are relevant for this assignment are:

- `print` is not longer a statement but a function: `print 'hello'` $\rightarrow$ `print('hello')`.
- Integer division now returns a float number: $5/2 = 2 \rightarrow 5/2 = 2{,}5$. Integer division is now achieved using `//`: $5//2 = 2$.

EXERCISE 3: Data structures

Connect to *Google's Python class* (`https://developers.google.com/edu/python/`) and focus on the sections entitled: *Lists*, *Sorting* and *Dicts and Files*. After executing the examples proposed implement the exercises described in the files `list1.py` and `wordcount.py` available in *Moodle*. Use the file called `alice.txt` as input of the `wordcount.py` program.

NOTE: In addition to the simple readings of the *Google's* course, we highly recommend reading the data structure tutorial in the *Python* documentation (`https://docs.python.org/3/tutorial/datastructures.html`).

EXERCISE 4: Classes and objects

Check sections entitled *Python Classes/Objects* y *Python Inheritance* in the *Python* tutorial of the *W3 schools* (`https://www.w3schools.com/python/python_classes.asp`). After this reading, implement the exercise described in the file `vehicle.py` available in *Moodle*.

NOTE: In addition to the simple readings of the *W3 schools* course, we highly recommend the readings about classes in the *Python* documentation (`https://docs.python.org/3/tutorial/classes.html`).

EJERCICIO 5: Flake8

You may known the utility called `lint` which scan C code and determines if the code is syntactically valid. `Flake8` perform this task on *Python*. Check your code of previous exercises with `flake8` until no errors or warnings are obtained.

NOTE: Solution to all proposed exercises must be added to the *Git* repository created in exercise 1. Your files must be contained in a directory named `exerciseX` (with X 2-4).

3. Deliverables to be Handed in

This assignment does not require to hand in any report. Using *Moodle* upload a single zip file containing the solution to all proposed exercises. Include in the zip file the (hidden) folder `.git`.

4. Grading Criteria

Since the solution to the exercises proposed in this assignment is available on the Internet, this practice will be marked as PASS or FAIL. In order to pass, you must satisfy all and each of the following criteria:

- The solution to all the proposed exercises satisfies the tests included in each file when these are executed in Python3.
- `flake8 <file>` should return not a single error or warning.

NOTE 1: In this assignment, we will not take into account the code quality providing it works and a averaged human being can understand it without descending into madness.

NOTE 2: The code used to grade the project will be the one uploaded to *Moodle*. Under no circumstance we will use the one push into Github.

NOTE 3: The students who pass an initial *Python* assessment will automatically pass assignment 1, and they will not have to hand in any deliverable.

5. Appendix: *Git*

Follows a short description of the *Git* basic commands. For a detailed description see `https://git-scm.com/docs`:

- `git clone url_remote_repository`
  Copy an existing repository into the directory you ran the command from.
- `git add [<pathspec>]...`
  Update local repository and prepare the content staged for the next commit.
- `git rm <pathspec>...`
  Remove files from the local repository.
- `git commit -m "Message"`
  Record changes to the repository.
- `git status`
  Display resources that have differences between the index file and the current commit, paths that have differences between the working tree and the index file, and paths in the working tree that are not tracked.
- `git checkout -- <pathspec>`
  Update files in the working tree to match the version in the index.
- `git push`
  Update remote refs using local refs, while sending objects necessary to complete the given refs.
- `git pull`
  Download changes from a remote repository.