CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

PYTHON - PART 2 - ACTIVITIES

Computer systems
CFGS DAW

Alfredo Oltra / Sergio García
sergio.garcia@ceedcv.es
alfredo.oltra@ceedcv.es
2019/2020

Versión:191111.1931

Licencia

Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

Actividad opcional. Normalmente hace referencia a un contenido que se ha comentado en la documentación por encima o que no se ha hecho, pero es interesante que el alumno investigue y practique.

Atención. Hace referencia a un tipo de actividad donde los alumnos suelencometer equivocaciones.

Pista. Información adicional que ayuda a la resolución del ejercicio.

PYTHON - PART 2 - ACTIVITIES

1. PREVIOUS INFORMATION

The objective of this unit is to make calls to the OS, so it is necessary to know the possible actions that the OS can perform.

In the next units we will know many of the commands that the terminal (both Linux/MacOs and Windows) has, but there are some basics that we can already know.

Keep in mind that the same functionality does not have to be done in the same way in Window and Linux/MacOs so you will indicate the necessary commands to do the activities in both OS. Unless otherwise indicated, you must perform the activity for one or another system.

2. ACTIVITIES

(1) Create a single program that shows what is the current directory.

- In Linux/MacOS you have to use the command `pwd`
- In Windows you have to use the command `cd`. This command actually exists on both (Linux/MacOS and Windows) and is used (along with a parameter) to change directories. But in Windows, if used without parameters returns the current directory.

(2) Create a program that displays the contents of the current folder

- In Linux/MacOS you have to use the command `ls`
- In Windows you have to use the command `dir`.

(3) Create a program that displays the contents of the current folder including hidden files

> The vast majority of OS commands need, or at least support, parameters. These parameters are information that is added to the command to particularize its action. In our case both (`ls` and `dir`) have many of them. You can see all available executing in Linux/MacOS `man ls` and in Windows with `help dir`.
>
> In Linux/MacOS the hidden files start with `.` , for example `.file`.

(4) Create a program that creates a folder named `SIN-Python-Block2`.

- In Linux/MacOS you have to use the command `mkdir`
- In Windows you have to use the command `md`. (`mkdir` is also valid)

(5) The solution to problem 1 has the disadvantage that you have to create different versions depending on the operating system. But Python provides other functions to be able to perform that operation in any OS. Create a new program that works on both platforms.

(6) Repeat the exercise number 4 but creating a program that works on Linux/MacOS and Windows (without using `mkdir` command).

(7) Creates a program that displays on screen a list of all files in the current directory **in green**.

- In Linux/MacOS you have to use the command `ls` with the parameter `-l`
- In Windows you have to use the command `dir`.
- The `run` function returns a special variable called *object*. Although the explanation of what an object is is not trivial, in a simplified way it could be said that it is a variable in which its information is distributed in sections. In our case, among others, there are 3 sections that interest us *stdout* (the information generated by the execution of the command), *stderr* (information on the error, if the execution was not correct) and *returncode* (zero in case everything is correct or a number referencing an error).
- To access each section you must use a dot. For example, if `myObject` is the name of the variable, `myBobjet.sdterr` allows access to the *sdterr* value of the `myObject` object.
- In order to get run to return the data correctly it is necessary to indicate as parameters `universal_newlines=True`, `stdout=subprocess.PIPE` and `stderr=subprocess.PIPE`

(8) Create a program that generates the following directory structure using *subprocess* module:

```
SIN Python
   ∟ Block2
      ∟ Activity1
   ∟ Block3
      ∟ Activity1
```

- In Linux/MacOS and Windows the command `cd` allows to change directory.

(9) Create a program that generates the directory structure of the exercise 8 that works on Linux/MacOS and Windows platforms.