Sistemas Informáticos

Scripting en Python 03. Actividades no evaluables 01

Autores: Sergi García, Alfredo Oltra

Actualizado Noviembre 2025

Scripting en Python - Parte 03

1. Información previa

El objetivo de esta unidad es realizar llamadas al sistema operativo, por lo que es necesario conocer las posibles acciones que puede realizar el SO. En las próximas unidades conoceremos muchos de los comandos que tiene la terminal (tanto en Linux/MacOS como en Windows), pero ya podemos aprender algunos básicos.

Ten en cuenta que la misma funcionalidad no tiene por qué realizarse de la misma forma en Windows y en Linux/MacOS, por lo que deberás indicar los comandos necesarios para hacer las actividades en ambos sistemas operativos. Salvo que se indique lo contrario, debes realizar la actividad en uno u otro sistema (no en ambos).

2. Ejercicio 01

Crea un único programa que muestre cuál es el directorio actual. En Linux/MacOS debes usar el comando “pwd”. En Windows debes usar el comando “cd”.Este comando existe tanto en Linux/MacOS como en Windows y se utiliza (junto con un parámetro) para cambiar de directorio, pero en Windows, si se usa sin parámetros, devuelve el directorio actual.

3. Ejercicio 02

Crea un programa que muestre el contenido de la carpeta actual.
En Linux/MacOS debes usar el comando “ls”.
En Windows debes usar el comando “dir”.

4. Ejercicio 03

Crea un programa que muestre el contenido de la carpeta actual, incluyendo los archivos ocultos.
La gran mayoría de comandos del sistema operativo necesitan, o al menos admiten, parámetros.
Estos parámetros son información que se añade al comando para particularizar su acción.
En nuestro caso, tanto “ls” como “dir” tienen muchos parámetros.
Puedes ver todos los disponibles ejecutando en Linux/MacOS: “man ls” y en Windows: “help dir”.
En Linux/MacOS los archivos ocultos empiezan por “.”, por ejemplo “.archivo”.

5. Ejercicio 04

Crea un programa que cree una carpeta llamada “SIN-Python-Block2”. En Linux/MacOS debes usar el comando “mkdir”. En Windows debes usar el comando “md” (también es válido “mkdir”).

6. Ejercicio 05

La solución al problema 1 tiene la desventaja de que hace falta crear versiones diferentes dependiendo del sistema operativo.
Pero Python ofrece otras funciones para poder realizar esa operación en cualquier SO.
Crea un nuevo programa que funcione en ambas plataformas.

7. Ejercicio 06

Repite el ejercicio número 4 pero creando un programa que funcione en Linux/MacOS y Windows (sin usar el comando “mkdir”).

8. Ejercicio 07

Crea un programa que muestre en pantalla una lista de todos los archivos del directorio actual en color verde. En Linux/MacOS debes usar el comando “ls” con el parámetro “-l”. En Windows debes usar el comando “dir”.

La función run devuelve una variable especial llamada objeto.
Aunque la explicación de lo que es un objeto no es trivial, de forma simplificada puede decirse que es una variable cuya información está distribuida en secciones.
En nuestro caso, entre otras, hay tres secciones que nos interesan:

“stdout”: la información generada por la ejecución del comando.
“stderr”: la información del error, si la ejecución no fue correcta.
“returncode”: cero si todo ha ido bien, o un número que representa un error.

Para acceder a cada sección debes usar un punto.
Por ejemplo, si “miObjeto” es el nombre de la variable, “miObjeto.stderr” permite acceder al valor stderr del objeto “miObjeto”.

Para que run devuelva los datos correctamente es necesario indicar como parámetros:
“universal_newlines=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE”.

9. Ejercicio 08

Crea un programa que genere la siguiente estructura de directorios usando el módulo subprocess:

SIN Python
- Block2
- Activity1
- Block3
- Activity1

En Linux/MacOS y Windows, el comando “cd” permite cambiar de directorio.

10. Ejercicio 09

Crea un programa que genere la estructura de directorios del ejercicio 8 y que funcione en las plataformas Linux/MacOS y Windows.

11. Ejercicio 10

Escribe un programa que solicite el nombre del usuario, su edad y el número de años de antigüedad en la empresa. Con esta información, crea una carpeta con el nombre del usuario y, si la suma de la edad y los años en la empresa es mayor que 35, una subcarpeta llamada “private”.