Laboratorio de Administración y Gestión de Redes y Sistemas
Grado en Ingeniería Telemática, 2024-2025

# Scripts de sistemas en Python

Escuela de Ingeniería de Fuenlabrada
Universidad Rey Juan Carlos

2024-2025

---

## Práctica 2.1. Uso de python

Crea el directorio `~/lagrs/practica02`

Escribe un script en Python 3 con el nombre `~/lagrs/practica02/ordena_lista.py`. Incluye un comentario en las primeras líneas tu nombre, apellidos y login.

Este script tendrá una función que:

- Recibirá una lista de elementos. Y comprobará que efectivamente su argumento es una lista, mostrando un error en otro caso.
- Cada elemento de esta lista será una cadena. Mostrando un error si alguno no lo fuera.
- Modificará de forma destructiva la lista, ordenada por longitud de la cadenas. Esto es, primero la cadena más corta, última la cadena más larga.

## Práctica 2.2. Acceso a un listado de procesos

El objetivo de esta práctica es que te familiarices con la librería *subprocess* para acceder a la shell de Linux desde Python.

Escribe un script en Python 3 con el nombre `~/lagrs/practica02/mi_top.py`. Incluye un comentario en las primeras líneas tu nombre, apellidos y login.

Debe mostrar un listado con el nombre de los procesos de sistema que más CPU consuman en ese momento. Aunque hay funciones específicas de Python para obtener información sobre los procesos, usa *subprocess*, invocando a `top -n 1`. En otras palabras, el programa debe tomar la salida de esta orden, procesarla en Python, filtrarla y escribir lo filtrado en la salida estándar. En otras palabras: ofrecer un subconjunto de la información ofrecida por `top -n 1`.

- El script mostrará los procesos de sistema que ocupen un porcentaje de CPU mayor a 0.0.
- En este listado debe aparecer el nombre del proceso, su pid, el nombre del usuario propietario, el uid del usuario, los nombres de todos los grupos a los que pertenece el usuario y el porcentaje de CPU consumido por el proceso. Para obtener la información relativa al usuario, emplea la orden de shell `id`.
- La información debe aparecer en formato de columnas. Similar a lo que hace `top`. Usa format.
- No uses regexp.
- No incluyas ninguna cabecera con los nombres de los campos.
- Sobra decir que es necesario que organices tu programa en funciones. Esto es algo que debes hacer siempre, también con programas cortos como este.

## Práctica 2.3. Enlaces simbólicos

En este ejercicio usarás los enlaces simbólicos para tener diferentes versiones del mismo programa, pero con el mismo nombre. Esto es, tendrás un enlace simbólico que podrá apuntar a diferentes versiones del mismo programa.

1. Por si te equivocas en este ejercicio, haz una copia de `~/lagrs/practica02/mi_top.py` en el directorio que prefieras. Por ejemplo en `~/tmp` (seguramente tendrás que crearlo)
2. Renombra la praćtica `~/lagrs/practica02/mi_top.py` como `~/lagrs/practica02/mi_top01.py`
3. Mediante la shell, crea un enlace simbólico llamado

   `~/lagrs/practica02/mi_top.py`

   que apunte a

   `~/lagrs/practica02/mi_top01.py`.

## Práctica 2.4. Opciones

Haz una copia de `~/lagrs/practica02/mi_top01.py` llamada `~/lagrs/practica02/mi_top02.py`. Usando la librería *optparse*, añade al menos 2 opciones al script `mi_top02.py`. (Sin contar la *h* de *help*, con esta serían al menos 3). Elige las opciones que desees, todas deben extraer y/o agregar diversa información de `top`. El comportamiento que tenía el script en la fase anterior puede ser una de esas opciones. Antes de implementar las opciones que hayas elegido, consúltalas con el profesor.

Cuando acabes el script, enlaza `mi_top.py` a `mi_top02.py`.

## Práctica 2.5. Módulos

Prepara ahora una versión de tu práctica anterior con el nombre `~/lagrs/practica02/mi_top03.py`, de forma que la práctica totalidad de las funciones estén en uno o varios módulo.

1. Los módulos deberán llamarse *TULOGIN_xxxxx.py*, esto es, empezarán por tu nombre de usuario en el laboratorio (en minúscula) y a continuación, lo que te parezca más adecuado. P..e *jperez_shell.py*, *mgarcia_opciones.py*, etc.
2. Los módulos estarán en el directorio `~/lagrs/lib`. Por tanto, tendrás que incluir este directorio en la variable de entorno *PYTHONPATH*. Hazlo en el fichero *.bashrc*.
3. Enlaza (desde la shell) `mi_top.py` a `mi_top03.py`.
4. Antes de hacer nada, el programa comprobará que

   - Realmente *PYTHONPATH* existe y incluye `~/lagrs/lib`. No importa si contiene otros directorios.

     Naturalmente, esto tiene que funcionar para cualquier usuario, esto es, el programa buscará el directorio `~/lagrs/lib` del usuario que ejecuta este programa (por ejemplo el profesor), no de tu usuario en particular.
   - Tus módulos realmente están en su sitio. Usa para ello la librería `os.path`. Observa que aquí sí es necesario que el nombre del módulo comience por tu login, no por el del usuario que ejecuta este programa (por ejemplo el profesor).

   Si alguna de estas comprobaciones falla, el programa lanzará una excepción personalizada.

## Práctica 2.6. Creación de un bot de telegram

1. Si no tienes Telegram, instálalo en tu móvil, en tu tablet o en tu portátil.
2. Crea un bot de telegram, con el nombre que quieras.
3. Ponle una foto al perfil del bot.
4. Escribe un programa con el nombre `~/lagrs/practica02/hola_telegram.py` que envíe un mensaje de tipo *hola mundo* a tu cliente Telegram y luego entre en un bucle infinito atendiendo a los mensajes que reciba (de cualquier usuario). Por cada mensaje recibido, envía una respuesta cualquiera al cliente y muestra todo en la pantalla del servidor, de forma legible (no el volcado en bruto del diccionario).
5. Observa que la fecha y hora del mensaje está en hora Unix, tendrás que pasarlo a otro formato. El que quieras, pero que sea legible por una persona.
6. Usa *format* para componer la respuesta. No uses la concatenación de cadenas con el operador `+`
7. El token no estará en el fuente. El programa leerá el token desde el fichero `token.txt`, que estará en el directorio actual. (En este caso, el directorio `~/lagrs/practica02/`, pero no escribas este trayecto, que el script lea el fichero desde el directorio actual)
8. Si el fichero con el token no existe o no se puede leer, el programa mostrará un mensaje de error por *stderr* y morirá. O si lo prefieres, levantará una excepción que describa el problema.

## Práctica 2.7. Creación de un bot de telegram (II)

Escribe un pograma llamado `~/lagrs/practica02/top_telegram.py` que implemente un bot de telegram que sea un *frontend* de tu práctica `mi_top03.py`.

Esto es, si el usuario escribe `mi_top` (o si el usuario escribe `top`, como prefieras), el bot devolverá la misma salida que tu `mi_top03.py`. Si el usuario añade alguna de las opciones que has previsto, el bot devolverá el resultado de ejecutar tu comando con esa opción.

Si el bot recibe un mensaje que no se corresponde con nada de lo aquí descrito, haz que responda de la forma que creas adecuada.

- No copies código desde `mi_top03.py`, importa código desde el módulo(s) que preparaste en la práctica 2.5.
- Ten en cuenta que los mensajes de telegram tienen un tamaño máximo de 4096 caracteres, trata esto como consideres conveniente.