Data Stream:

Como el nombre indica, data Stream hace referencia a la entrada y salida de datos en el prompt, el script o el archivo. Es la manera que tenemos de introducir datos, y la manera de la que recibimos datos por parte de la computadora.

Hay tres Streams principales, una de entrada, y dos de salida. Cuando decimos Streams, nos referimos al espacio que hay entre el exterior y los datos de la computadora, o sea, es lo que ocurre, entre el teclado y una variable creándose, o entre la pantalla imprimiendo un resultado, o un error. Lo que nos lleva a mencionar los tipos de entrada y salida:

STDIN: Estas son las formas normales de introducir datos, por ejemplo con la función input(), o con el metodo sys.argv, del módulo sys.

STDOUT: Estas se refieren a lo que ocurre entre que se ejecuta el script, y es imprimido por pantalla, o sea los return, o los print, por ejemplo

STDERR: Este se refiere a cuando no obtenemos un output porque así lo deseamos, o porque lo ejecuta el script, sino porque lo que se ejecuta es un error y aparentemente, estos vienen de un lugar diferente, de una corriente o Stream diferente.

Usar comandos del sistema en Python:

A veces, hay tipos de comandos que no se encuentran en módulos de Python internos, o en los módulos externos, y que seria muy complejo tener que escribir un script solo para eso cuando de hecho ya existen en el sistema, por eso es por lo que viene muy útil, poder utilizar las funciones o métodos del sistema, importando los subprocesos.

Una captura de pantalla de una computadora

Descripción generada automáticamente

Aquí estamos usando el modulo subprocess, que nos permite acceder a los subprocesos del sistema, por ejemplo el metodo subprocess.run, nos devuelve una instancia, un objeto de la clase CompletedProcess el cual contiene informaciones realcionadas con la ejecucion del comando.

Una vez que estamos utilizando los subrocesos, se crea un ambiente para el subproceso, y este es el que asume el control del flujo de datos, y hasta que este no termine de ejecutarse, el control de flujo de datos no vuelve al ambiente en el que estamos ejecutando en script.

Capturing Output:

En caso de que necesitemos utilizar el output brindado por el subprocess, debemos al final de argumentos que le estamos pasando pasarle el “capture\_output”, como un tercer argumento. Esto lo que hara sera guardar retultados en 3 cadenas distintas. Suponiendo que estemos igualando esto a una variable result:

result.stdout: Esto devolvera el output, pero como no estamos trabajando directamente con python, lo devolvera como lo que conocemos como un bytes array, por lo que tendremos que decodificar, y separar con .decode.split() para poder usarlos. Si hay un error devolvera una cadena bytes array vacia

result.stderr: Esto devolvera el error, pero como no estamos trabajando directamente con python, lo devolvera como lo que conocemos como un bytes array, por lo que tendremos que decodificar, y separar con .decode.split() para poder usarlos. Si hay un output devolvera una cadena bytes array vacia

result.returncode : Este devuelve un numero, si es cero, el stream usado para el outpur es el stdout, si es diferente de cero, el stream usado sera stderr.