



## 135- [PF] - Lab - Análisis del valor de la automatización

GitHub - ian-parra/Aws-Python: AWS Python Labs Documentation This repository serves as a comprehensive documentation of AWS Python labs. It provides a valuable resource for individuals looking for AWS Python Labs Documentation.

<https://github.com/ian-parra/Aws-Python>

### Información general sobre el laboratorio

Puede utilizar Linux para realizar muchas tareas administrativas desde el terminal o desde la línea de comandos de Bash. Python proporciona varios módulos que también puede utilizar para ejecutar comandos en la línea de comandos. En este laboratorio, utilizará `os.system()` y `subprocess.run()` para ejecutar comandos de Bash desde Python.

En este laboratorio, deberá realizar lo siguiente:

- utilizar `os.system()` para ejecutar un comando de Bash
- utilizar `subprocess.run()` para ejecutar comandos de Bash

### Tiempo de finalización estimado

30 minutos

### Acceso al IDE de AWS Cloud9

1. Para activar el entorno de su laboratorio, desplácese hasta la parte superior de estas instrucciones y seleccione **Start Lab** (Iniciar laboratorio).

Se abrirá el panel **Start Lab** (Iniciar laboratorio), donde se muestra el estado del laboratorio.

2. Espere hasta que aparezca el mensaje *Lab status: ready* (Estado del laboratorio: listo) y, a continuación, para cerrar el panel **Start Lab** (Iniciar laboratorio), haga clic en la **X**.

3. En la parte superior de estas instrucciones, elija **AWS**.

AWS Management Console se abrirá en una pestaña nueva del navegador. El sistema iniciará la sesión de forma automática.

Nota: Si no se abre una pestaña nueva del navegador, generalmente aparece un anuncio o un icono en la parte superior de este, el cual indica que el navegador no permite que se abran ventanas emergentes en el sitio. Haga clic en el anuncio o en el icono, y elija Allow pop ups (Permitir ventanas emergentes).

4. En AWS Management Console, elija **Services** (Servicios) > **Cloud9**. En el panel **Your environments** (Sus entornos), busque la tarjeta **reStart-python-cloud9** y elija **Open IDE** (Abrir IDE).

Se abre el entorno de AWS Cloud9.

Nota: Si se abre una ventana emergente con el mensaje `.c9/project.settings have been changed on disk` (Se ha modificado la configuración de `.c9/project.` en el disco), elija **Discard** (Descartar) para ignorarlo. Del mismo modo, si una ventana de diálogo le pide que **Show third-party content** (Muestre contenido de terceros), elija **No** para rechazar la indicación.

## Creación del archivo de ejercicio de Python

1. En la barra de menú, elija **File > New From Template > Python File** (Archivo > Nuevo a partir de plantilla > Archivo en Python).  
Esta acción crea un archivo sin título.
2. Elimine el código de muestra del archivo de plantilla.
3. Elija **File > Save As...** (Archivo > Guardar como...), proporcione un nombre adecuado para el archivo de ejercicios (por ejemplo, `sys-admin.py`) y guárdelo en el directorio `/home/ec2-user/environment`.

## Acceso a la sesión del terminal

1. En su IDE de AWS Cloud9, elija el icono **+** y seleccione **New Terminal** (Nuevo terminal).  
Se abre una sesión de terminal.
2. Para ver el directorio en el que está trabajando actualmente, escriba `pwd`. Este comando lleva a `/home/ec2-user/environment`.
3. En este directorio, también debería ser capaz de localizar el archivo que creó en la sección anterior.

## Ejercicio 1: Uso de `os.system`

Python posee varios módulos que permiten ejecutar comandos de Bash desde Python. En este ejercicio, utilizará `os.system()` para ejecutar el comando de Bash `ls`, que muestra el contenido de un directorio.

1. En el panel de navegación del IDE, elija el archivo que creó en la sección **Creación del archivo de ejercicios de Python** anterior.
2. Importe el módulo `os`:

```
import os
```

1. Recuerde que un módulo contiene funciones que otros desarrolladores han escrito. La función `os.system()` recibe un argumento de tipo String. Para ejecutar un comando de Bash, escriba el siguiente comando:

```
os.system("ls")
```

1. Guarde el archivo en el IDE de Cloud 9 y seleccione **Run** (Ejecutar) para ejecutar el archivo.
2. El resultado debería mostrar el contenido de su directorio actual. Verifique que el resultado sea similar al siguiente ejemplo. Tenga en cuenta que el contenido de su directorio podría ser diferente.

```
sys-admin.py README.md
```

## Ejercicio 2: Uso de `subprocess.run`

Si bien `os.system()` es fácil de utilizar debido a que requiere un argumento de tipo String, se recomienda utilizar la función `subprocess.run()`, que es más eficiente. Puede utilizar el módulo `subprocess` para generar nuevos procesos; conectarse a canalizaciones de entrada, salida o error; y obtener códigos de error. La función `subprocess.run()` puede aceptar muchos argumentos nuevos, pero esos argumentos adicionales son opcionales.

La lista completa de argumentos para `subprocess.run()` se parece a lo siguiente:

```
subprocess.run(args, *, stdin=None, input=None, stdout=None, stderr=None, capture_output=False, shell=False, cwd=None, timeout=None, check=False, encoding=None, errors=None, text=None, env=None, universal_newlines=None)
```

1. Para este laboratorio, mantendrá el código simple.
2. En el archivo que creó para este laboratorio, importe el módulo `subprocess`:

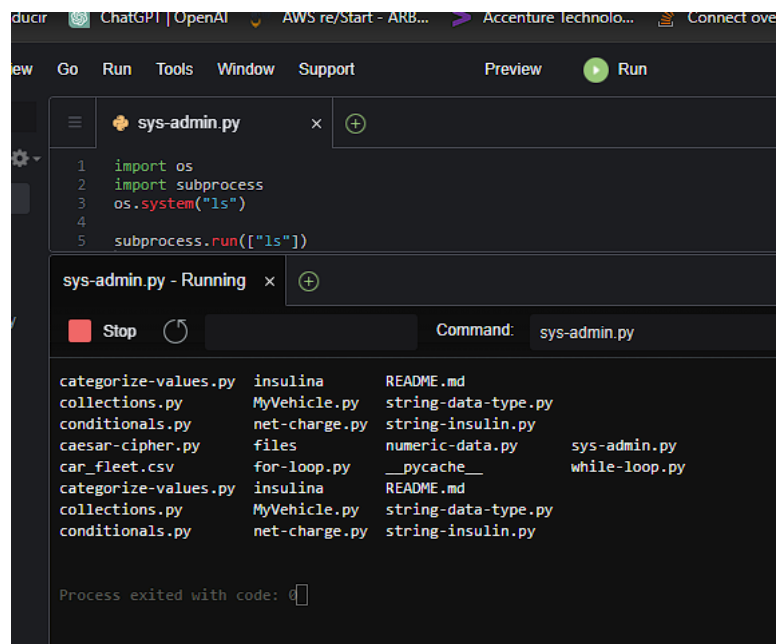
```
import subprocess
```

1. Para ejecutar el comando de Bash `ls`, escriba el siguiente comando:

```
subprocess.run(["ls"])
```

1. Guarde el archivo en el IDE de Cloud 9 y seleccione **Run** (Ejecutar) para ejecutar el archivo.
2. Confirme que el resultado muestre el archivo en el directorio de forma similar al siguiente ejemplo. (El contenido del directorio podría ser diferente).

```
sys-admin.py  sys-admin_2.py  README.md
```



The screenshot shows a Cloud 9 IDE interface. The editor displays a file named `sys-admin.py` with the following code:

```
1 import os
2 import subprocess
3 os.system("ls")
4
5 subprocess.run(["ls"])
```

Below the editor, the `sys-admin.py - Running` terminal window is open. It shows the command `sys-admin.py` and the output of the `ls` command:

```
categorize-values.py  insulina  README.md
collections.py        MyVehicle.py  string-data-type.py
conditionals.py       net-charge.py  string-insulin.py
caesar-cipher.py      files         numeric-data.py    sys-admin.py
car_fleet.csv         for-loop.py   __pycache__        while-loop.py
categorize-values.py  insulina  README.md
collections.py        MyVehicle.py  string-data-type.py
conditionals.py       net-charge.py  string-insulin.py
```

At the bottom of the terminal, it says "Process exited with code: 0".

Tenga en cuenta que el resultado es similar al de `os.system()` del Ejercicio 1, pero que está utilizando el módulo `subprocess` en vez del módulo `os`.

### Ejercicio 3: Uso de `subprocess.run` con dos argumentos

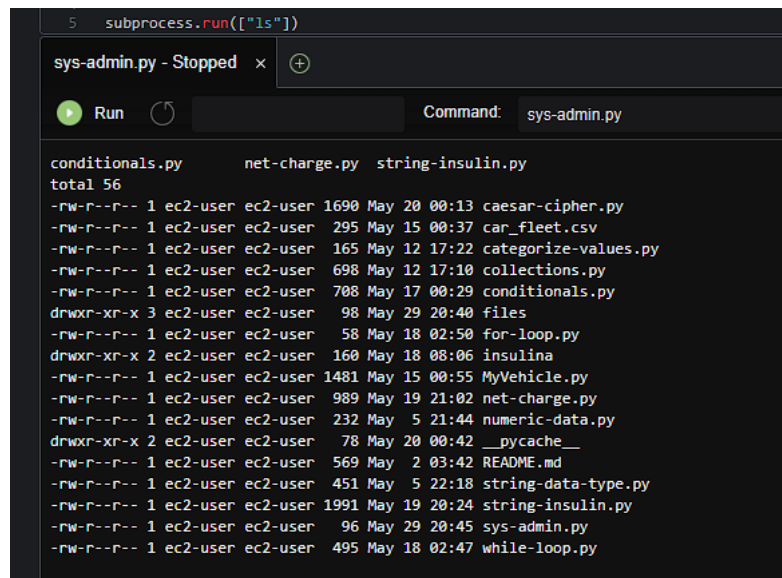
En Python, los corchetes son tipos de datos de lista, lo que significa que `run()` puede adoptar una lista de argumentos. Continúe agregando al script.

1. En el archivo de laboratorio para este ejercicio, modifique la línea final del script para incluir un argumento adicional:

```
subprocess.run(["ls", "-l"])
```

1. El argumento `"-l"` indica al comando `ls` que utilice un formato de descripción larga.
2. Guarde el archivo en el IDE de Cloud 9 y seleccione **Run** (Ejecutar) para ejecutar el archivo de nuevo.
3. Confirme que el resultado sea similar al siguiente ejemplo.

```
total 12
-rw-r--r-- 1 ec2-user ec2-user 55 Apr 16 20:20 sys-admin.py
-rw-r--r-- 1 ec2-user ec2-user 343 Apr 16 19:07 sys-admin_2.py
-rw-r--r-- 1 ec2-user ec2-user 569 Apr 6 02:17 README.md
```



```
5 subprocess.run(["ls"])

sys-admin.py - Stopped x (+)

Run Command: sys-admin.py

conditionals.py net-charge.py string-insulin.py
total 56
-rw-r--r-- 1 ec2-user ec2-user 1690 May 20 00:13 caesar-cipher.py
-rw-r--r-- 1 ec2-user ec2-user 295 May 15 00:37 car_fleet.csv
-rw-r--r-- 1 ec2-user ec2-user 165 May 12 17:22 categorize-values.py
-rw-r--r-- 1 ec2-user ec2-user 698 May 12 17:10 collections.py
-rw-r--r-- 1 ec2-user ec2-user 708 May 17 00:29 conditionals.py
drwxr-xr-x 3 ec2-user ec2-user 98 May 29 20:40 files
-rw-r--r-- 1 ec2-user ec2-user 58 May 18 02:50 for-loop.py
drwxr-xr-x 2 ec2-user ec2-user 160 May 18 08:06 insulina
-rw-r--r-- 1 ec2-user ec2-user 1481 May 15 00:55 MyVehicle.py
-rw-r--r-- 1 ec2-user ec2-user 989 May 19 21:02 net-charge.py
-rw-r--r-- 1 ec2-user ec2-user 232 May 5 21:44 numeric-data.py
drwxr-xr-x 2 ec2-user ec2-user 78 May 20 00:42 __pycache__
-rw-r--r-- 1 ec2-user ec2-user 569 May 2 03:42 README.md
-rw-r--r-- 1 ec2-user ec2-user 451 May 5 22:18 string-data-type.py
-rw-r--r-- 1 ec2-user ec2-user 1991 May 19 20:24 string-insulin.py
-rw-r--r-- 1 ec2-user ec2-user 96 May 29 20:45 sys-admin.py
-rw-r--r-- 1 ec2-user ec2-user 495 May 18 02:47 while-loop.py
```

## Ejercicio 4: Uso de subprocess.run con tres argumentos

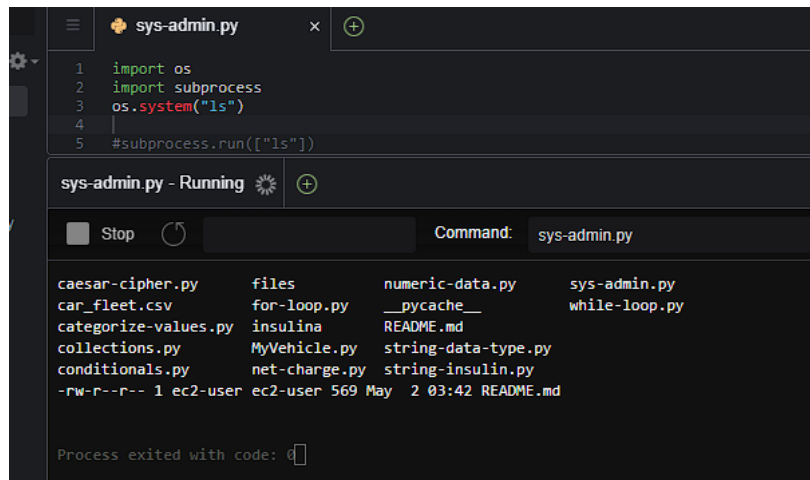
Ahora llamará a `subprocess.run()` con tres argumentos. El tercer argumento será el nombre de un directorio.

1. Vuelva al archivo Python y modifique la línea final del script:

```
subprocess.run(["ls", "-l", "README.md"])
```

1. Guarde el archivo en el IDE de Cloud 9 y seleccione **Run** (Ejecutar) el archivo.
2. Confirme que el resultado esperado sea similar al siguiente ejemplo.

```
-rw-r--r-- 1 ec2-user ec2-user 569 Apr 6 02:17 README.md
```



## Ejercicio 5: Recuperación de información del sistema

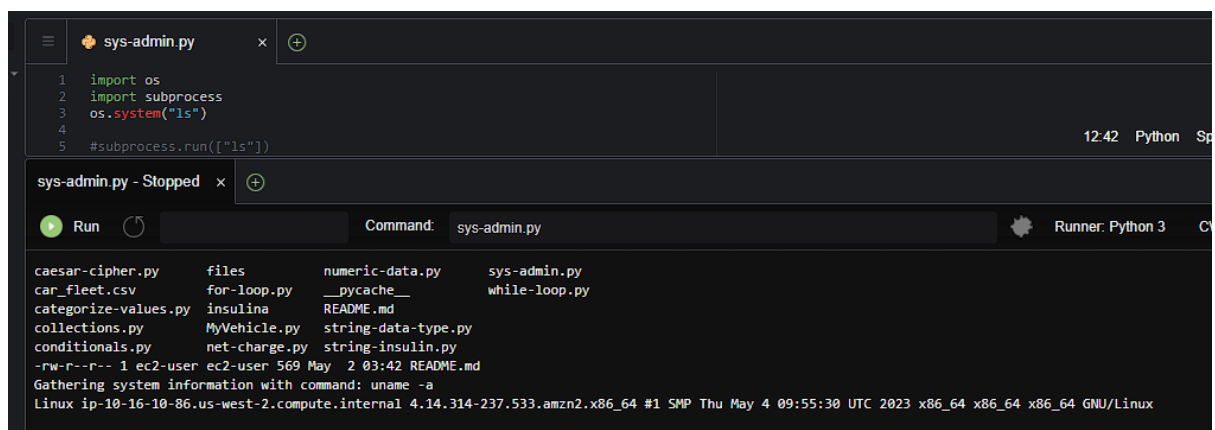
La función `subprocess.run()` es eficiente porque puede utilizarse para ejecutar cualquier comando de Bash. En este ejercicio, llamará al comando `uname` para obtener información del sistema.

1. Vuelva al archivo Python y escriba el siguiente código:

```
command="uname"
commandArgument="-a"
print(f'Gathering system information with command: {command} {commandArgument}')
subprocess.run([command,commandArgument])
```

1. Guarde el archivo en el IDE de Cloud 9 y seleccione **Run** (Ejecutar) para ejecutar el archivo.
2. Confirme que el resultado esperado sea similar al siguiente ejemplo.

```
Gathering system information with command: uname -a
Linux ip-172-31-29-181 4.4.0-139-generic #165-Ubuntu SMP Wed Oct 24 10:58:50
UTC 2018 x86_64 x86_64 x86_64 GNU/Linux
```



## Ejercicio 6: Recuperación de información sobre el espacio en disco

Para hacer hincapié en que `subprocess.run()` le permite ejecutar cualquier comando, ejecute el comando `df` para obtener información del disco.

1. Vuelva al archivo Python y escriba el siguiente código:

```
command="ps"
commandArgument="-x"
```

```
print(f'Gathering active process information with command: {command} {commandArgument}')
subprocess.run([command,commandArgument])
```

```
1 import os
2 import subprocess
3 os.system("ls")
4
5 #subprocess.run(["ls"])
```

sys-admin.py - Stopped

Run Command: sys-admin.py Runner: Python 3

```
conditionals.py net-charge.py string-insulin.py
-rw-r--r-- 1 ec2-user ec2-user 569 May 2 03:42 README.md
Gathering system information with command: uname -a
Linux ip-10-16-10-86.us-west-2.compute.internal 4.14.314-237.533.amzn2.x86_64 #1 SMP Thu May 4 09:55:30 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
Gathering active process information with command: ps -x
```

PID	TTY	STAT	TIME	COMMAND
4751	?	S	0:00	sshd: ec2-user@notty
4752	?	Ss	0:00	bash --noprofile --norc
4753	?	S1	0:04	vfs-worker {"pingInterval":5000,"nodePath":"/home/ec2
5345	?	S	0:00	sshd: ec2-user@notty
5350	?	Ss	0:00	bash --noprofile --norc
5354	?	S1	0:00	vfs-worker {"pingInterval":5000,"nodePath":"/home/ec2
5987	?	Ss1	0:03	/opt/c9/dependencies/node-linux-x64/0ea3352564e939884
6094	?	S1	0:00	/opt/c9/dependencies/node-linux-x64/0ea3352564e939884
6526	?	S1	0:00	/opt/c9/dependencies/node-linux-x64/0ea3352564e939884
10899	?	Ss	0:00	/home/ec2-user/.c9/bin/tmux -C -u2 -L cloud92.2 new -
10900	pts/1	Ss+	0:00	bash -l -c trap 'printf "\e[01;30m\n\nProcess exited
11140	pts/1	S+	0:00	python3 /home/ec2-user/environment/sys-admin.py
11144	pts/1	R+	0:00	ps -x

Process exited with code: 0

1. Guarde el archivo en el IDE de Cloud 9 y seleccione **Run** (Ejecutar) para ejecutar el archivo.
2. Confirme que el resultado esperado sea similar al siguiente ejemplo.

```
Gathering active process information with command: ps -x
```

PID	TTY	STAT	TIME	COMMAND
18976	pts/459	S+	0:00	python3.6 lab_15_2.py
18977	pts/459	R+	0:00	ps -x
21139	pts/459	S	0:00	/bin/bash -c export OLD_HOME=/home/ccc_4dfa91ec5a_
21164	pts/459	S	0:00	bash --rcfile /home/ccc_4dfa91ec5a_45122/.termrc -

¡Felicitaciones! Ha llamado a comandos de Bash desde Python.

## Finalizar laboratorio

¡Felicitaciones! Ha llegado al final del laboratorio.

1. Elige **End Lab** (Finalizar laboratorio) en la parte superior de esta página y, a continuación, selecciona Yes (Sí) para confirmar que deseas finalizar el laboratorio.

Un panel muestra el mensaje *DELETE has been initiated... You may close this message box now* (Se ha iniciado la ELIMINACIÓN... Ya puedes cerrar este mensaje).

1. Aparece brevemente el mensaje *Ended AWS Lab Successfully* (El laboratorio de AWS finalizó correctamente), que indica que el laboratorio ha finalizado.