1. Crear un nuevo directorio de trabajo llamado “tutorial_terminal”:

```
mkdir tutorial_terminal
```

2. Colócate dentro del directorio que acabas de crear:

```
cd tutorial_terminal
```

3. Crea un archivo llamado hello_world.py:

```
touch hello_world.py
```

4. Abre el archivo que acabas de crear y escribe print(‘Hello world!’):

```
nano hello_world.py
```

```
  GNU nano 5.4
print('hello world!')
```

Tras escribir el código python pulsar ctrl+X para salir, respondemos con Y para guardar los cambios y pulsamos enter.

5. Crear un directorio llamado “scripts_python” y movemos ahí el archivo .py que creamos anteriormente:

```
mkdir scripts_python
```

```
mv hello_world.py scripts_python
```

6. Crear un directorio llamado “scripts_bash”, colocarnos ahí dentro y crear un archivo bash llamado ejecuta_python.sh:

```
mkdir scripts_bash
cd scripts_bash
```

```
touch ejecuta_python.sh
```

/. Abrir el archivo bash creado y escribir las instrucciones necesarias para hacer que se ejecute el script de python cada 5 segundos un total de 3 veces, una vez finalizado mostrar por pantalla tarea con Python completada!:

```
nano ejecuta_python.sh
```

```
for i in {1..3}; do
        python3 /workspaces/exercise-terminal-challenge/tutorial_terminal/scripts_python/hello_world.py
        sleep 5
done

echo "Tarea completada!"
```

Tras escribir el código bash pulsar ctrl+X para salir, respondemos con Y para guardar los cambios y pulsamos enter.

8. Dar permisos de ejecución y ejecutar el script .bash:

```
chmod +x ejecuta_python.sh
```

```
./ejecuta_python.sh
```

Output:

```
hello world!
hello world!
hello world!
Tarea completada!
```

9. Volver a la ruta anterior, colócate dentro del directorio scripts_python y haz una copia del script que ya existe dentro de ese mismo directorio:

```
cd ..
```

```
cd scripts_python
```

```
cp hello_world.py copia_hello_world.py
```

10. Modifica la copia de modo que cree un archivo “test.txt” y escriba lo siguiente “Tarea completada a las {fecha y hora actual}” guardar el archivo en un directorio llamado res que deberás crear previamente:

```
nano copia_hello_world.py
```

```
from datetime import datetime

fecha_hora_actual = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

with open('/workspaces/exercise-terminal-challenge/tutorial_terminal/res/test.txt', "a") as archivo:
    archivo.write(f'Tarea finalizada a las {fecha_hora_actual}')
```

11. Ejecuta la copia directamente desde el terminal, borra el archivo “hello_world.py” y comprueba que solo ha quedado la copia:

```
python3 copia_hello_world.py
```

```
rm hello_world.py
ls
```

12. Vuelve al directorio padre y muestra el contenido del archivo de texto que se creó anteriormente:

```
cd ..
```

```
cat res/test.txt
```

13. Crea un directorio llamado data y colócate dentro, posteriormente descarga el siguiente archivo .tar (https://getsamplefiles.com/download/tar/sample-1.tar), descomprímelo y muestra los nombres de los archivos que terminan en .jpg:

```
wget https://getsamplefiles.com/download/tar/sample
```

```
tar -xvf sample-1.tar
```

```
find -name "*.jpg"
```

14. Por último muestra el directorio actual en el que estás colocado y el historial de comandos ejecutados en la sesión:

```
pwd
```

```
history
```

Comandos usados en este tutorial:

mkdir
cd
cd ..
touch
nano
mv
chmod +x
./
cp
python3
rm
ls
cat
wget
tar
find
pwd
history

Otros comandos útiles:

sudo: Permite ejecutar comandos con privilegios de superusuario. Esto es necesario para realizar tareas de administración del sistema.

apt: El sistema de gestión de paquetes APT es fundamental en Ubuntu para instalar, actualizar y gestionar software. Comandos como apt-get o apt install son útiles para ello.

grep: Permite buscar patrones en archivos o en la salida de otros comandos. Es útil para filtrar información.