Comunicar procesos en Python con PIPES


```
import subprocess

def comunicar_procesos():
    proceso1 = subprocess.Popen(["echo", "Hola, proceso 2!"], stdout=subprocess.PIPE)
       proceso2 = subprocess.Popen(["grep", "Hola"], stdin=proceso1.stdout, stdout=subprocess.PIPE)
    salida = proceso2.communicate()[0]
    print(salida.decode())

comunicar_procesos()
```

1. Importación del módulo subprocess

```
import subprocess
```

import subprocess: Importa el módulo subprocess, que permite crear nuevos procesos, conectarse a sus canales de entrada/salida/error y obtener sus códigos de retorno.

2. Definición de la función comunicar_procesos

```
def comunicar_procesos():
```

def comunicar_procesos():: Define una función llamada comunicar_procesos que encapsula el código para crear y comunicar entre dos procesos.

3. Creación del primer proceso

```
proceso1 = subprocess.Popen(["echo", "Hola, proceso 2!"], stdout=subprocess.PIPE)
```

– subprocess.Popen: Inicia un nuevo proceso.
– ["echo", "Hola, proceso 2!"]: Comando que se ejecuta en el nuevo proceso. echo es un comando que imprime un texto, en este caso, «Hola, proceso 2!».
– stdout=subprocess.PIPE: Redirige la salida estándar del primer proceso a un pipe, que puede ser leído por otro proceso.

4. Creación del segundo proceso

```
proceso2 = subprocess.Popen(["grep", "Hola"], stdin=proceso1.stdout, stdout=subprocess.PIPE)
```

– subprocess.Popen: Inicia otro nuevo proceso.
– ["grep", "Hola"]: Comando que se ejecuta en el segundo

proceso. grep es un comando que busca patrones en texto. En este caso, busca la palabra «Hola».
– stdin=proceso1.stdout: Redirige la entrada estándar del segundo proceso a la salida estándar del primer proceso, creando un pipe entre los dos procesos.
– stdout=subprocess.PIPE: Redirige la salida estándar del segundo proceso a un pipe, que será leído posteriormente.

5. Comunicación entre procesos y obtención de la salida

```
salida = proceso2.communicate()[0]
```

– proceso2.communicate(): Espera a que el segundo proceso termine y lee su salida.
– [0]: Obtiene la salida estándar del segundo proceso.

6. Decodificación y muestra de la salida

```
print(salida.decode())
```

– salida.decode(): Decodifica la salida de bytes a una cadena de texto utilizando la codificación predeterminada (UTF-8).
– print(): Imprime la cadena de texto decodificada en la salida estándar (pantalla).

7. Llamada a la función comunicar_procesos

comunicar_procesos()

comunicar_procesos(): Llama a la función comunicar_procesos para ejecutar el código definido en ella.