

Realizar una comunicación entre dos procesos en Python (un proceso envía información a otro proceso y el otro proceso responde)

Padre (envía información al proceso hijo y recibe la respuesta del hijo)

[crayon-6a9d4f92963d0066331631/]

Hijo (el hijo recibe el mensaje del padre y responde al padre)

[crayon-6a9d4f92963d9698157946/]

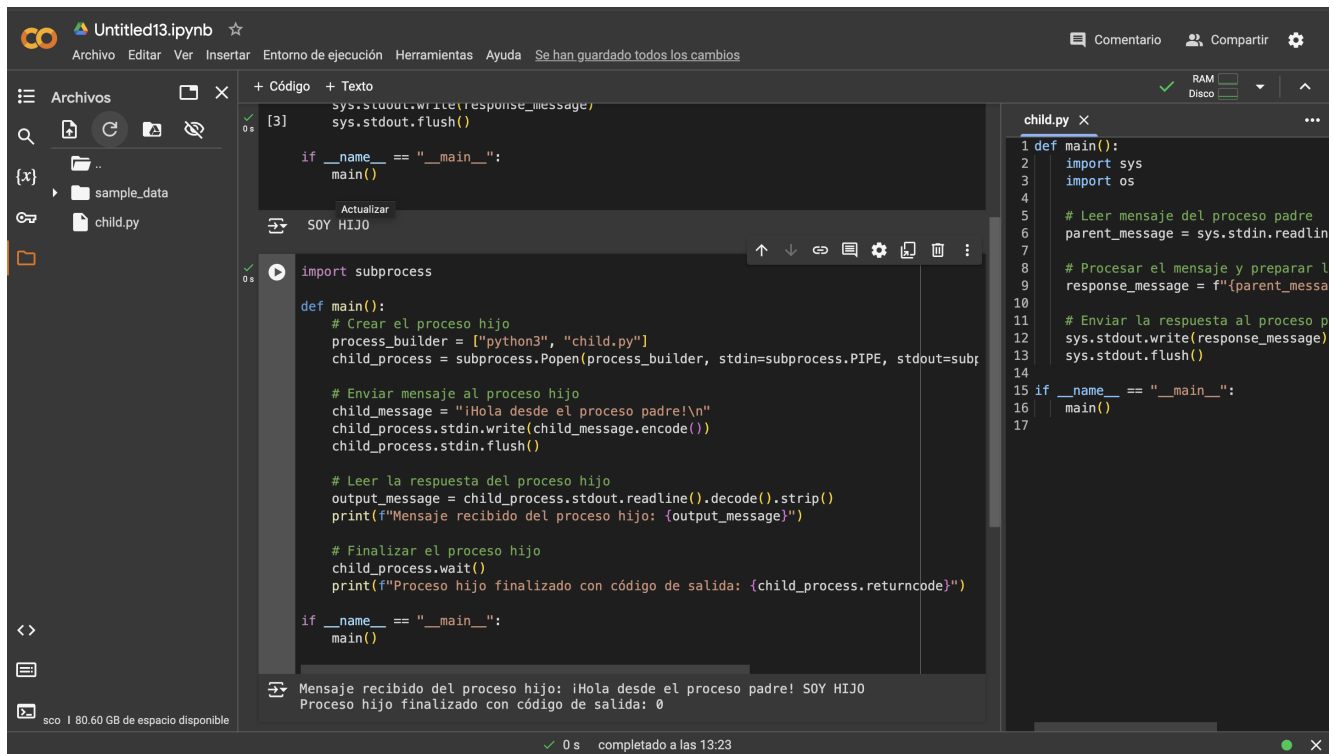
Explicación del código

parent_process.py:

- Utiliza `subprocess.Popen` para iniciar el proceso hijo con `stdin` y `stdout` redirigidos.
- Escribe un mensaje al `stdin` del proceso hijo y lo envía.
- Lee la respuesta del `stdout` del proceso hijo.
- Espera a que el proceso hijo termine y obtiene el código de salida.

child_process.py:

- Lee el mensaje del stdin que proviene del proceso padre.
- Prepara una respuesta y la escribe en el stdout.



The screenshot shows a Jupyter Notebook environment with two code cells. The first cell contains a simple script that writes a response message to stdout and flushes it. The second cell contains a more complex script that uses the `subprocess` module to create a child process, send it a message, and receive a response. The output of the second cell shows the messages being exchanged between the parent and child processes.

```
sys.stdout.write(response_message)
sys.stdout.flush()

if __name__ == "__main__":
    main()
```

Actualizar
SOY HIJO

```
import subprocess

def main():
    # Crear el proceso hijo
    process_builder = ["python3", "child.py"]
    child_process = subprocess.Popen(process_builder, stdin=subprocess.PIPE, stdout=subprocess.PIPE)

    # Enviar mensaje al proceso hijo
    child_message = "¡Hola desde el proceso padre!\n"
    child_process.stdin.write(child_message.encode())
    child_process.stdin.flush()

    # Leer la respuesta del proceso hijo
    output_message = child_process.stdout.readline().decode().strip()
    print(f"Mensaje recibido del proceso hijo: {output_message}")

    # Finalizar el proceso hijo
    child_process.wait()
    print(f"Proceso hijo finalizado con código de salida: {child_process.returncode}")

if __name__ == "__main__":
    main()
```

Mensaje recibido del proceso hijo: ¡Hola desde el proceso padre! SOY HIJO
Proceso hijo finalizado con código de salida: 0

child.py

```
1 def main():
2     import sys
3     import os
4
5     # Leer mensaje del proceso padre
6     parent_message = sys.stdin.readline()
7
8     # Procesar el mensaje y preparar la respuesta
9     response_message = f"¡Hola desde el proceso hijo! {parent_message}"
10
11    # Enviar la respuesta al proceso padre
12    sys.stdout.write(response_message)
13    sys.stdout.flush()
14
15 if __name__ == "__main__":
16     main()
17
```

0 s | 80.60 GB de espacio disponible

0 s completado a las 13:23