# 16. Communicating with external processes

September 18, 2025

The `subprocess` module in Python is used to spawn new processes, connect to their input/output/error pipes, and obtain their return codes. It allows you to start new applications or programs from your Python script.

### 0.0.1 Basic Usage

1. **Import the subprocess module:**

[1]:
```
import subprocess
```

2. **Running a Simple Command:** Use `subprocess.run()` to run a command and wait for it to complete.

[2]:
```
result = subprocess.run(['ls', '-l'], capture_output=True, text=True)
print(result.stdout)
```

```
total 350
-rw-r--r-- 1 user user 17973 Sep 15 18:21 01. Introduction.ipynb
-rw-r--r-- 1 user user 58581 Sep 15 18:21 02. Python basics.ipynb
-rw-r--r-- 1 user user 34027 Sep 15 18:25 03. Control Flow.ipynb
-rw-r--r-- 1 user user 38461 Sep 15 18:25 04. Lists and Tuples.ipynb
-rw-r--r-- 1 user user 21197 Sep 15 18:21 05. Functions.ipynb
-rw-r--r-- 1 user user 20896 Sep 15 18:21 06. Methods on known data types.ipynb
-rw-r--r-- 1 user user 13148 Sep 18 17:26 07. Modules.ipynb
-rw-r--r-- 1 user user  6847 Sep 15 18:21 08. Introduction to File
Handling.ipynb
-rw-r--r-- 1 user user 24576 Sep 17 21:37 09. More data structures.ipynb
-rw-r--r-- 1 user user  9563 Sep 17 16:36 10. Exception Handling.ipynb
-rw-r--r-- 1 user user 12546 Sep 17 21:44 11. Introduction to Object-Oriented
Programming.ipynb
-rw-r--r-- 1 user user 26083 Sep 15 18:21 12. Working with different data
formats.ipynb
-rw-r--r-- 1 user user 13964 Sep 15 18:21 13. Working with databases.ipynb
-rw-r--r-- 1 user user 11883 Sep 15 18:21 14. Decorators.ipynb
-rw-r--r-- 1 user user 29164 Sep 17 21:45 15. Object-Oriented Programming.ipynb
-rw-r--r-- 1 user user 13152 Sep 18 21:56 16. Communicating with external
processes.ipynb
-rw-r--r-- 1 user user  2207 Sep 15 18:21 99. Project.ipynb
```

```
drwxr-xr-x 2 user user     0 Sep 17 21:37 images
drwxr-xr-x 2 user user     0 Sep 18 17:25 pdf
drwxr-xr-x 2 user user     0 Sep 18 21:50 practice
-rw-r--r-- 1 user user     0 Sep 15 18:19 README.md
```

- `['ls', '-l']`: The command to run. Here, it lists directory contents in long format.
- `capture_output=True`: Captures the standard output and error.
- `text=True`: Returns output as string rather than bytes.

### 0.0.2 Running a Command with Different Options

3. **Check if Command is Successful:**

[3]:
```
result = subprocess.run(['ls', '-l'], capture_output=True, text=True)
if result.returncode == 0:
    print('Command succeeded:', result.stdout)
else:
    print('Command failed:', result.stderr)
```

```
Command succeeded: total 350
-rw-r--r-- 1 user user 17973 Sep 15 18:21 01. Introduction.ipynb
-rw-r--r-- 1 user user 58581 Sep 15 18:21 02. Python basics.ipynb
-rw-r--r-- 1 user user 34027 Sep 15 18:25 03. Control Flow.ipynb
-rw-r--r-- 1 user user 38461 Sep 15 18:25 04. Lists and Tuples.ipynb
-rw-r--r-- 1 user user 21197 Sep 15 18:21 05. Functions.ipynb
-rw-r--r-- 1 user user 20896 Sep 15 18:21 06. Methods on known data types.ipynb
-rw-r--r-- 1 user user 13148 Sep 18 17:26 07. Modules.ipynb
-rw-r--r-- 1 user user  6847 Sep 15 18:21 08. Introduction to File
Handling.ipynb
-rw-r--r-- 1 user user 24576 Sep 17 21:37 09. More data structures.ipynb
-rw-r--r-- 1 user user  9563 Sep 17 16:36 10. Exception Handling.ipynb
-rw-r--r-- 1 user user 12546 Sep 17 21:44 11. Introduction to Object-Oriented
Programming.ipynb
-rw-r--r-- 1 user user 26083 Sep 15 18:21 12. Working with different data
formats.ipynb
-rw-r--r-- 1 user user 13964 Sep 15 18:21 13. Working with databases.ipynb
-rw-r--r-- 1 user user 11883 Sep 15 18:21 14. Decorators.ipynb
-rw-r--r-- 1 user user 29164 Sep 17 21:45 15. Object-Oriented Programming.ipynb
-rw-r--r-- 1 user user 13152 Sep 18 21:56 16. Communicating with external
processes.ipynb
-rw-r--r-- 1 user user  2207 Sep 15 18:21 99. Project.ipynb
drwxr-xr-x 2 user user     0 Sep 17 21:37 images
drwxr-xr-x 2 user user     0 Sep 18 17:25 pdf
drwxr-xr-x 2 user user     0 Sep 18 21:50 practice
-rw-r--r-- 1 user user     0 Sep 15 18:19 README.md
```

4. **Suppressing Output:**

[4]:
```
subprocess.run(['ls', '-l'], stdout=subprocess.DEVNULL, stderr=subprocess.
↪DEVNULL)
```

[4]:
```
CompletedProcess(args=['ls', '-l'], returncode=0)
```

- `stdout=subprocess.DEVNULL` and `stderr=subprocess.DEVNULL` suppress the output.

5. **Running Command without Waiting for Completion:**

[5]:
```
process = subprocess.Popen(['sleep', '5'])
print('Command started, will sleep for 5 seconds')
```

```
Command started, will sleep for 5 seconds
```

- `subprocess.Popen` starts the process without waiting for it to complete.

### 0.0.3 Advanced Usage

6. **Capturing Output:**

[6]:
```
result = subprocess.run(['ls', '-l'], capture_output=True, text=True)
print('stdout:', result.stdout)
print('stderr:', result.stderr)
```

```
stdout: total 350
-rw-r--r-- 1 user user 17973 Sep 15 18:21 01. Introduction.ipynb
-rw-r--r-- 1 user user 58581 Sep 15 18:21 02. Python basics.ipynb
-rw-r--r-- 1 user user 34027 Sep 15 18:25 03. Control Flow.ipynb
-rw-r--r-- 1 user user 38461 Sep 15 18:25 04. Lists and Tuples.ipynb
-rw-r--r-- 1 user user 21197 Sep 15 18:21 05. Functions.ipynb
-rw-r--r-- 1 user user 20896 Sep 15 18:21 06. Methods on known data types.ipynb
-rw-r--r-- 1 user user 13148 Sep 18 17:26 07. Modules.ipynb
-rw-r--r-- 1 user user  6847 Sep 15 18:21 08. Introduction to File
Handling.ipynb
-rw-r--r-- 1 user user 24576 Sep 17 21:37 09. More data structures.ipynb
-rw-r--r-- 1 user user  9563 Sep 17 16:36 10. Exception Handling.ipynb
-rw-r--r-- 1 user user 12546 Sep 17 21:44 11. Introduction to Object-Oriented
Programming.ipynb
-rw-r--r-- 1 user user 26083 Sep 15 18:21 12. Working with different data
formats.ipynb
-rw-r--r-- 1 user user 13964 Sep 15 18:21 13. Working with databases.ipynb
-rw-r--r-- 1 user user 11883 Sep 15 18:21 14. Decorators.ipynb
-rw-r--r-- 1 user user 29164 Sep 17 21:45 15. Object-Oriented Programming.ipynb
-rw-r--r-- 1 user user 13152 Sep 18 21:56 16. Communicating with external
processes.ipynb
-rw-r--r-- 1 user user  2207 Sep 15 18:21 99. Project.ipynb
drwxr-xr-x 2 user user     0 Sep 17 21:37 images
drwxr-xr-x 2 user user     0 Sep 18 17:25 pdf
drwxr-xr-x 2 user user     0 Sep 18 21:50 practice
-rw-r--r-- 1 user user     0 Sep 15 18:19 README.md
```

stderr:

7. **Redirecting Output to a File:**

[7]:
```
with open('output.txt', 'w') as f:
    subprocess.run(['ls', '-l'], stdout=f)
```

8. **Piping Commands:**

[8]:
```
p1 = subprocess.Popen(['ls', '-l'], stdout=subprocess.PIPE)
p2 = subprocess.Popen(['grep', 'py'], stdin=p1.stdout, stdout=subprocess.PIPE,␣
 ↪text=True)
p1.stdout.close()  # Allow p1 to receive a SIGPIPE if p2 exits.
output = p2.communicate()[0]
print(output)
```

```
-rw-r--r-- 1 user user 17973 Sep 15 18:21 01. Introduction.ipynb
-rw-r--r-- 1 user user 58581 Sep 15 18:21 02. Python basics.ipynb
-rw-r--r-- 1 user user 34027 Sep 15 18:25 03. Control Flow.ipynb
-rw-r--r-- 1 user user 38461 Sep 15 18:25 04. Lists and Tuples.ipynb
-rw-r--r-- 1 user user 21197 Sep 15 18:21 05. Functions.ipynb
-rw-r--r-- 1 user user 20896 Sep 15 18:21 06. Methods on known data types.ipynb
-rw-r--r-- 1 user user 13148 Sep 18 17:26 07. Modules.ipynb
-rw-r--r-- 1 user user  6847 Sep 15 18:21 08. Introduction to File
Handling.ipynb
-rw-r--r-- 1 user user 24576 Sep 17 21:37 09. More data structures.ipynb
-rw-r--r-- 1 user user  9563 Sep 17 16:36 10. Exception Handling.ipynb
-rw-r--r-- 1 user user 12546 Sep 17 21:44 11. Introduction to Object-Oriented
Programming.ipynb
-rw-r--r-- 1 user user 26083 Sep 15 18:21 12. Working with different data
formats.ipynb
-rw-r--r-- 1 user user 13964 Sep 15 18:21 13. Working with databases.ipynb
-rw-r--r-- 1 user user 11883 Sep 15 18:21 14. Decorators.ipynb
-rw-r--r-- 1 user user 29164 Sep 17 21:45 15. Object-Oriented Programming.ipynb
-rw-r--r-- 1 user user 13152 Sep 18 21:56 16. Communicating with external
processes.ipynb
-rw-r--r-- 1 user user  2207 Sep 15 18:21 99. Project.ipynb
```

- This pipes the output of `ls -l` to `grep py`.

9. **Handling Timeouts:**

[9]:
```
try:
    result = subprocess.run(['sleep', '10'], timeout=1)
except subprocess.TimeoutExpired:
    print('Command timed out')
```

```
Command timed out
```

10. **Error Handling:**

[10]:
```
try:
    result = subprocess.run(['false'], check=True)
except subprocess.CalledProcessError as e:
    print('Command failed with return code', e.returncode)
```

```
Command failed with return code 1
```

- `check=True` raises an exception if the command exits with a non-zero status.