12. Communicating with external processes

October 16, 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.

[13]:
```
result = subprocess.run(['ls', '-l'], capture_output=True, text=True)
print(result.stdout)
```

```
total 856
-rw-r--r--  1 john  staff   3609 Jun 17 08:28 01. Introduction.ipynb
-rw-r--r--  1 john  staff  61194 Jun 17 14:12 02. Built-in Types.ipynb
-rw-r--r--  1 john  staff  23467 Jun 17 16:58 03. Control Flow.ipynb
-rw-r--r--  1 john  staff  61260 Jun 18 15:35 04. More on functions and
iterables.ipynb
-rw-r--r--  1 john  staff  36848 Jun 19 09:32 05. Advanced Data Structures.ipynb
-rw-r--r--@ 1 john  staff  15299 Jun 17 08:28 06. Decorators.ipynb
-rw-r--r--@ 1 john  staff  31846 Jun 20 09:47 07. Object-Oriented
Programming.ipynb
-rw-r--r--  1 john  staff  11324 Jun 20 10:24 08. Context Managers.ipynb
-rw-r--r--  1 john  staff  65142 Jun 17 08:28 09. Testing your code.ipynb
-rw-r--r--  1 john  staff  17148 Jun 20 15:34 10. Logging.ipynb
-rw-r--r--@ 1 john  staff  26466 Jun 20 16:40 11. Concurrent execution.ipynb
-rw-r--r--  1 john  staff  18648 Jun 20 16:48 12. Communicating with external
processes.ipynb
-rw-r--r--  1 john  staff  12150 Jun 17 08:28 books.csv
-rw-r--r--@ 1 john  staff     11 Jun 17 08:28 example.txt
-rw-r--r--@ 1 john  staff  15222 Jun 17 08:28 multithreading.png
-rw-r--r--  1 john  staff     58 Jun 20 10:22 out.txt
-rw-r--r--  1 john  staff     27 Jun 20 10:23 out2.txt
-rw-r--r--  1 john  staff   1295 Jun 19 19:37 output.txt
```

```
-rw-r--r--  1 john  staff    899 Jun 20 16:23 process_files.py
-rw-r--r--  1 john  staff    593 Jun 17 08:28 users.json
```

- `['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 824
-rw-r--r--  1 john  staff   3609 Jun 17 08:28 01. Introduction.ipynb
-rw-r--r--  1 john  staff  61194 Jun 17 14:12 02. Built-in Types.ipynb
-rw-r--r--  1 john  staff  23467 Jun 17 16:58 03. Control Flow.ipynb
-rw-r--r--  1 john  staff  61260 Jun 18 15:35 04. More on functions and
iterables.ipynb
-rw-r--r--  1 john  staff  36848 Jun 19 09:32 05. Advanced Data Structures.ipynb
-rw-r--r--@ 1 john  staff  15299 Jun 17 08:28 06. Decorators.ipynb
-rw-r--r--@ 1 john  staff  31743 Jun 17 08:28 07. Object-Oriented
Programming.ipynb
-rw-r--r--  1 john  staff  10936 Jun 17 08:28 08. Context Managers.ipynb
-rw-r--r--  1 john  staff  65142 Jun 17 08:28 09. Testing your code.ipynb
-rw-r--r--  1 john  staff  17026 Jun 17 08:28 10. Logging.ipynb
-rw-r--r--@ 1 john  staff  26354 Jun 19 19:29 11. Concurrent execution.ipynb
-rw-r--r--  1 john  staff  13159 Jun 19 19:36 12. Communicating with external
processes.ipynb
-rw-r--r--  1 john  staff  12150 Jun 17 08:28 books.csv
-rw-r--r--@ 1 john  staff     11 Jun 17 08:28 example.txt
-rw-r--r--@ 1 john  staff  15222 Jun 17 08:28 multithreading.png
-rw-r--r--  1 john  staff    899 Jun 17 08:28 process_files.py
-rw-r--r--  1 john  staff    593 Jun 17 08:28 users.json
```

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 824
-rw-r--r--  1 john  staff   3609 Jun 17 08:28 01. Introduction.ipynb
-rw-r--r--  1 john  staff  61194 Jun 17 14:12 02. Built-in Types.ipynb
-rw-r--r--  1 john  staff  23467 Jun 17 16:58 03. Control Flow.ipynb
-rw-r--r--  1 john  staff  61260 Jun 18 15:35 04. More on functions and
iterables.ipynb
-rw-r--r--  1 john  staff  36848 Jun 19 09:32 05. Advanced Data Structures.ipynb
-rw-r--r--@ 1 john  staff  15299 Jun 17 08:28 06. Decorators.ipynb
-rw-r--r--@ 1 john  staff  31743 Jun 17 08:28 07. Object-Oriented
Programming.ipynb
-rw-r--r--  1 john  staff  10936 Jun 17 08:28 08. Context Managers.ipynb
-rw-r--r--  1 john  staff  65142 Jun 17 08:28 09. Testing your code.ipynb
-rw-r--r--  1 john  staff  17026 Jun 17 08:28 10. Logging.ipynb
-rw-r--r--@ 1 john  staff  26354 Jun 19 19:29 11. Concurrent execution.ipynb
-rw-r--r--  1 john  staff  13159 Jun 19 19:36 12. Communicating with external
processes.ipynb
-rw-r--r--  1 john  staff  12150 Jun 17 08:28 books.csv
-rw-r--r--@ 1 john  staff     11 Jun 17 08:28 example.txt
-rw-r--r--@ 1 john  staff  15222 Jun 17 08:28 multithreading.png
-rw-r--r--  1 john  staff    899 Jun 17 08:28 process_files.py
-rw-r--r--  1 john  staff    593 Jun 17 08:28 users.json

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 john  staff   3609 Jun 17 08:28 01. Introduction.ipynb
-rw-r--r--  1 john  staff  61194 Jun 17 14:12 02. Built-in Types.ipynb
-rw-r--r--  1 john  staff  23467 Jun 17 16:58 03. Control Flow.ipynb
-rw-r--r--  1 john  staff  61260 Jun 18 15:35 04. More on functions and
iterables.ipynb
-rw-r--r--  1 john  staff  36848 Jun 19 09:32 05. Advanced Data Structures.ipynb
-rw-r--r--@ 1 john  staff  15299 Jun 17 08:28 06. Decorators.ipynb
-rw-r--r--@ 1 john  staff  31743 Jun 17 08:28 07. Object-Oriented
Programming.ipynb
-rw-r--r--  1 john  staff  10936 Jun 17 08:28 08. Context Managers.ipynb
-rw-r--r--  1 john  staff  65142 Jun 17 08:28 09. Testing your code.ipynb
-rw-r--r--  1 john  staff  17026 Jun 17 08:28 10. Logging.ipynb
-rw-r--r--@ 1 john  staff  26354 Jun 19 19:29 11. Concurrent execution.ipynb
-rw-r--r--  1 john  staff  13159 Jun 19 19:36 12. Communicating with external
processes.ipynb
-rw-r--r--  1 john  staff    899 Jun 17 08:28 process_files.py
```

- 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.
```