Python subprocess Module - Notes

# 🔹 Purpose

The subprocess module allows Python code to spawn new processes, connect to their input/output/error pipes, and obtain their return codes. It is commonly used to run shell commands within Python scripts.

# ⚙️ Common Functions

* subprocess.run() – Runs a command, waits for it to finish, and returns a CompletedProcess object.
* subprocess.Popen() – Executes a command in the background and allows more advanced control.
* subprocess.call() – Runs a command and returns its exit code.
* subprocess.check\_call() – Same as call() but raises CalledProcessError if the command fails.
* subprocess.check\_output() – Returns the output of the command. Raises CalledProcessError if it fails.

# 📄 Example Usage

* subprocess.run(["ls", "-l"])
* output = subprocess.check\_output(["df", "-h"])
* result = subprocess.run(["ping", "-c", "2", "google.com"], capture\_output=True, text=True)
* process = subprocess.Popen(["ls"], stdout=subprocess.PIPE)

# 📤 Output Handling

* capture\_output=True – Captures stdout and stderr (Python 3.7+)
* text=True – Returns output as string instead of bytes
* stdout=subprocess.PIPE – Redirects output for further processing

# ❗ Error Handling

* check=True – Raises exception if command returns non-zero
* subprocess.CalledProcessError – Exception raised when check=True and command fails
* try/except – Use to catch and handle subprocess errors gracefully

# 🚀 Common Use Cases

* Run shell commands from Python scripts
* Automate backups or service restarts
* Monitor system or service status
* Execute deployment tasks or CI/CD hooks