mypy Python type checker

START HERE

`mypy <file.py>`
Check one file

`mypy .`
Check a directory

`mypy -p <package>`
Check a whole package

`mypy --strict .`
Strictest checks

`mypy --ignore-missing-imports .`
Silence untyped libs

`mypy --install-types`
Install missing stubs

`x = 1  # type: ignore`
Silence one line

`dmypy run -- .`
Fast daemon re-checks

Your code annotations → `mypy` → **Type checker** no code runs → `reads` → **Stubs and py.typed** library types → `reports` → **Errors** file:line [code]

CONCEPTS

mypy only checks annotated functions by default; add --check-untyped-defs to look inside the rest.

RUNNING

`mypy <file.py> <dir>`
Check files and directories

`mypy .`
Check everything under here

`mypy -p <package>`
Check a package recursively

`mypy -m <module>`
Check one module by name

`mypy --exclude '<regex>' .`
Skip matching paths

`mypy --explicit-package-bases .`
Fix 'source file found twice'
Needed with src/ layouts and missing __init__.py files; set mypy_path or namespace_packages in the config as well.

`mypy --python-executable <path> .`
Use another env's packages

`mypy -V`
Print the mypy version

STRICTNESS

`mypy --strict .`
Enable the strict flag bundle
Does not include --warn-unreachable. Flags set explicitly after --strict override it.

`mypy --check-untyped-defs .`
Check bodies of untyped functions

`mypy --disallow-untyped-defs .`
Require annotations on functions

`mypy --disallow-incomplete-defs .`
Forbid half-annotated functions

`mypy --disallow-untyped-calls .`
Typed code may not call untyped

`mypy --disallow-any-generics .`
Require type args on list, dict

`mypy --warn-return-any .`
Warn when returning Any

`mypy --warn-unused-ignores .`
Flag useless type: ignore

`mypy --warn-unreachable .`
Flag dead or redundant code

`mypy --strict-equality .`
Flag impossible == comparisons

IMPORTS AND STUBS

`mypy --ignore-missing-imports .`
Silence unresolved imports
Blunt: those modules become Any. Prefer a per-module setting.

`mypy --install-types`
Install suggested stub packages

`mypy --install-types --non-interactive .`
Install stubs without asking

`pip install types-<name>`
Add typeshed stubs for a library

In CI, pin stub packages in your dev requirements instead of using --install-types.

OUTPUT

`mypy --pretty .`
Show source snippets and carets

`mypy --show-error-codes .`
Print the [code] of each error

`mypy --no-color-output .`
Plain text for logs

CONFIG

`mypy --config-file <path> .`
Use a specific config file

`mypy --warn-unused-configs .`
Catch misspelled sections

`mypy --disable-error-code <code> .`
Turn off one error kind

Where config is found (first match wins)

File	Section
`mypy.ini`	[mypy]
`.mypy.ini`	[mypy]
`pyproject.toml`	[tool.mypy]
`setup.cfg`	[mypy]

Config files are never merged: the first one found walking up from the current directory is the only one used.

A per-module section looks like [mypy-pkg.*] in INI or [[tool.mypy.overrides]] with module = "pkg.*" in pyproject.toml.

IGNORING AND ANNOTATING

`x = f()  # type: ignore`
Suppress errors on this line

`x = f()  # type: ignore[<code>]`
Suppress only one error code

IGNORING AND ANNOTATING (cont.)

`# mypy: ignore-errors`
Skip the whole file (top comment)

`reveal_type(<expr>)`
Print the inferred type
Only for mypy runs; remove it afterwards. At runtime it fails (Python 3.11+ has typing.reveal_type).

`assert isinstance(<x>, <type>)`
Narrow a type

`if TYPE_CHECKING:`
Imports only for the checker

`def f(x: int | None) -> str:`
Annotate args and return

DAEMON AND CACHE

`dmypy run -- <args>`
Check, starting the daemon if needed

`dmypy recheck`
Re-check changed files only

`dmypy status`
Show whether the daemon runs

`dmypy stop`
Stop the daemon politely

DAEMON AND CACHE (cont.)

The daemon keeps state in memory, so re-checks after the first run take a fraction of the time.

`rm -rf .mypy_cache`
Reset after odd results

RECIPES

pyproject.toml setup

```
[tool.mypy]
python_version = "3.12"
strict = true
warn_unreachable = true

[[tool.mypy.overrides]]
module = ["yaml.*",
"requests_toolbelt.*"]
ignore_missing_imports = true
```

Adopt strictness gradually

```
# start loose, tighten per
package
[tool.mypy]
check_untyped_defs = true

[[tool.mypy.overrides]]
module = "myapp.core.*"
disallow_untyped_defs = true
```

CI check

```
pip install mypy types-requests
mypy --warn-unused-configs --
pretty -p myapp
```

Fast local loop

```
dmypy run -- --pretty src/
# edit code, run again
dmypy run -- --pretty src/
dmypy stop
```