mypy Python type checker

CONCEPTS

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

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

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 -V
Print the mypy version

STRICTNESS

mypy --strict .
Enable the strict flag bundle

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

STRICTNESS (cont.)

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

IMPORTS AND STUBS

mypy --ignore-missing-imports .
Silence unresolved imports

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

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

reveal_type(*<expr>***)**
Print the inferred 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

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
```

CI check

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