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

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-any-generics .
Require type args on list, dict

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

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

IMPORTS AND STUBS

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

mypy --install-types
Install suggested stub packages

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

OUTPUT

mypy --pretty .
Show source snippets and carets

CONFIG

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

Where config is found (first match wins)

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

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

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

def f(x: int | None) → str:
Annotate args and return

DAEMON AND CACHE

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