Pylint Python linter

START HERE

`pylint <file.py>`
Lint one file

`pylint --disable=<msg>`
Silence a message

`pylint <package>`
Lint a package

`pylint --generate-toml-config`
Print a config file

`pylint --recursive=y .`
Lint a whole tree

`pylint --fail-under=9 <path>`
Gate CI on score

`pylint -E <path>`
Errors only, fast

`pylint --help-msg=<msg>`
Explain a message

Config pyproject / pylintrc → load → **Targets** files, packages → parse → **Checkers** enabled messages → report → **Output** text, json2, score + exit code

CONCEPTS

Every finding has an id (C0116) and a symbol (missing-function-docstring); either works in enable, disable and pragmas.

The first letter of an id is its category: C convention, R refactor, W warning, E error, F fatal. I is informational.

Pylint imports your code's dependencies to infer types, so run it in the project's virtualenv or you get false import-error and no-member.

RUNNING

`pylint <file.py>`
Lint a single file

`pylint <package>`
Lint a package or module by name

A directory is only linted if it is a package (has __init__.py) or you pass --recursive=y.

`pylint --recursive=y <dir>`
Lint every .py file under a directory

`pylint -j 0 <path>`
Use all CPU cores

Duplicate-code detection is incomplete when linting in parallel.

`pylint --ignore-paths=<regex>`
Skip paths matching a regex

`pylint --load-plugins=<module>`
Load a plugin module

CHOOSING MESSAGES

`pylint -E <path>`
Report errors only (--errors-only)

`pylint --disable=<msg>[,<msg>]`
Turn messages off by id or symbol

`pylint --enable=<msg>`
Turn a message back on

`pylint --disable=all --enable=<msg>`
Run only the listed messages

CHOOSING MESSAGES (cont.)

`pylint --disable=C,R <path>`
Disable whole categories (C, R, W, E, F)

OUTPUT AND SCORE

`pylint --fail-under=<score> <path>`
Exit non-zero if the score is below it

`pylint --score=n <path>`
Hide the 0-10 score (--score=n)

`pylint -f json2 <path>`
Machine-readable JSON output

`pylint --msg-template='<tmpl>'`
Custom line, e.g. '{path}:{line}: {msg_id} {msg}'

`pylint --exit-zero <path>`
Always exit 0, even with findings

HELP AND DISCOVERY

`pylint --help-msg=<msg>`
Explain a message by id or symbol

`pylint --list-msgs`
List all messages by checker

`pylint --list-msgs-enabled`
Show which messages are on and off

CONFIGURATION

`pylint --rcfile=<file> <path>`
Use a specific config file

`pylint --generate-toml-config`
Print a documented pyproject-style config

`pylint --generate-rcfile`
Print a documented .pylintrc

Both generators include the effect of the other options you pass, so pylint --disable=C0114 --generate-rcfile bakes it in.

CONFIGURATION (cont.)

`pyproject.toml: [tool.pylint.main]`
Project-wide settings (jobs, ignore-paths)

`[tool.pylint."messages control"]`
disable / enable lists in pyproject.toml

`[tool.pylint.format] max-line-length = <n>`
Change the line length limit (default 100)

`[tool.pylint.basic] good-names = [<names>]`
Allow short names like i, df, x

INLINE CONTROL

`# pylint: disable=<msg>`
End of a line: silence it for that line

`# pylint: disable=<msg>`
On its own line: silence to end of the block

At module level, from there to the end of the file.

`# pylint: disable-next=<msg>`
Silence the following line only

`pylint --enable=useless-suppression`
Find pragmas that no longer silence anything

MESSAGES YOU WILL MEET

`C0114 / C0115 / C0116`
Missing module / class / function docstring

`C0103 invalid-name`
Name breaks the naming style

`C0301 line-too-long`
Line exceeds max-line-length

`W0611 unused-import`
Imported but never used

`W0612 / W0613`
Unused variable / unused argument

`E0401 import-error`
Module cannot be imported (wrong env?)

MESSAGES YOU WILL MEET (cont.)

`E1101 no-member`
Attribute not found on inferred type

`E0602 undefined-variable`
Name is not defined

`W0102 dangerous-default-value`
Mutable default argument

`W0718 broad-exception-caught`
except Exception is too general

`W1203 logging-fstring-interpolation`
Use lazy %s formatting in logging calls

`R1705 no-else-return`
else after return is unnecessary

`R0913 too-many-arguments`
More arguments than max-args (default 5)

`R0902 / R0912 / R0914 / R0915`
Too many attributes / branches / locals / statements

RECIPES

CI gate with a fixed bar

```
# fail the job below 9.0, no score noise in logs
pylint --recursive=y --fail-under=9 src tests
```

Adopt it on an old codebase

```
# 1. only real errors first
pylint -E --recursive=y .
# 2. then one message at a time
pylint --disable=all --enable=unused-import --recursive=y .
```

pyproject.toml baseline

```
[tool.pylint.main]
jobs = 0
ignore-paths = ["^migrations/"]

[tool.pylint."messages control"]
disable = ["missing-docstring", "too-few-public-methods"]

[tool.pylint.format]
max-line-length = 88
```

JSON report for tooling

```
pylint -f json2 --output=pylint.json src
# or a compact gcc-style line
pylint --msg-template='{path}:{line}: [{msg_id}] {msg}' -sn src
```