isort Python import sorter

START HERE

`isort .`
Sort imports in place

`isort --check-only --diff .`
CI check, with the changes

`isort --profile black .`
Match Black's style

`isort --diff <file.py>`
Preview one file

`isort -l <n> .`
Set line length

`isort -`
Sort stdin to stdout

`isort --show-config`
See the settings in use

`isort --version`
Which version?

Imports top of each file → `classify` → **Sections** FUTURE, STDLIB, THIRDPARTY, FIRSTPARTY, LOCALFOLDER → `sort` → **Order in section** import x, then from x import y → `wrap` → **Output** line length, multi-line mode

CONCEPTS

isort rewrites files in place by default. Run it with --diff first or commit before the first run. With Black, always set --profile black (or the config equivalent), or the two tools fight over wrapping.

Config is found automatically: pyproject.toml ([tool.isort]), .isort.cfg, setup.cfg, tox.ini or .editorconfig. Command-line flags override it.

RUN

`isort .`
Sort every Python file below here

`isort <file.py> <dir>`
Sort given files and folders

`isort --check-only .`
Exit 1 if any file would change

`isort --diff .`
Print the changes, write nothing

`isort --check-only --diff .`
Show why a CI check fails

`isort --diff --check-only <file.py>`
Diff plus failing exit code, one file

`isort -d <file.py>`
Print sorted result to stdout

`isort -c -q .`
Check, print errors only

`isort --color --diff .`
Colored diff

`isort --atomic .`
Skip saving if the result has a syntax error

`isort -j <n> .`
Process files in parallel

`isort --only-modified -v .`
Verbose, but only for changed files

`isort --interactive .`
Confirm each change

STYLE

`isort --profile black .`
Use a preset (black, django, google, ...)

`isort -l <n> .`
Maximum import line length

STYLE (cont.)

`isort -m <mode> .`
Multi-line wrap mode (GRID, VERTICAL, 3, ...)

`isort --tc --up .`
Trailing comma and parentheses

`isort --sl .`
One imported name per line

`isort --ca .`
Combine 'as' imports on one line

`isort --fss .`
Sort by module, ignore import / from style

`isort --ff .`
Put from imports before straight imports

`isort --dt .`
Do not order by CONSTANT / Class / func

`isort --case-sensitive .`
Respect case when sorting modules

`isort --float-to-top .`
Move mid-file imports to the top

`isort --lai <n> .`
Blank lines after the import block

`isort --remove-redundant-aliases .`
Drop 'import os as os'

SECTIONS

`isort -p <pkg> .`
Treat <pkg> as first party

`isort --src <dir> .`
Add a source path (finds first-party code)

`isort -o <pkg> .`
Treat <pkg> as third party

`isort -b <module> .`
Treat <module> as standard library

`isort --known-local-folder <name> .`
Treat <name> as a local folder

`isort --sd <SECTION> .`
Default section for unknown modules

`isort --py <ver> .`
Stdlib list for a Python version (310, auto)

`isort --ds .`
Put all imports in one section

SECTIONS (cont.)

`isort -t <module> .`
Force a module to the top of its section

`isort --virtual-env <dir> .`
Venv used to detect third party

Sections, in order

Section	Holds
`FUTURE`	__future__
`STDLIB`	Python standard library
`THIRDPARTY`	Installed packages
`FIRSTPARTY`	Your own packages
`LOCALFOLDER`	Relative imports (from . import x)

IMPORTS AND FILES

`isort -a '<import>' .`
Add an import to every file

`isort --rm '<import>' .`
Remove an import from every file

`isort -s <name> .`
Skip a file or directory

`isort --extend-skip <name> .`
Skip more, keep the default skips

`isort --sg '<glob>' .`
Skip by glob pattern

`isort --gitignore .`
Skip files listed in .gitignore

`isort --filter-files <file.py>`
Apply skips to explicitly named files

`isort --ext <ext> .`
Also sort files with this extension

`isort --show-files .`
List files isort would process

`isort --filename <name> -`
Tell stdin its filename for config lookup

CONFIG

`isort --show-config <file.py>`
Show settings and where they came from

`isort --settings-path <path> .`
Use a specific config file or folder

`isort --resolve-all-configs .`
Use the closest config for each subfolder

CONFIG (cont.)

`isort --honor-noqa .`
Respect '# noqa' comments

Skip markers in code

Marker	Effect
`# isort: skip`	Leave this import line alone
`# isort: skip_file`	Ignore the whole file
`# isort: off / on`	Pause and resume sorting
`# isort: split`	Sort above and below separately

Common pyproject keys

Key	Same as
`profile`	--profile
`line_length`	-l
`src_paths`	--src
`known_first_party`	-p
`skip / extend_skip`	-s / --extend-skip
`force_sort_within_sections`	--fss

In config files the names use underscores (line_length), on the CLI hyphens or short flags.

EXIT CODES

Code	Meaning
`0`	Done, or nothing to change with --check-only
`1`	--check-only: files would change
`2`	Bad command-line arguments

RECIPES

pyproject.toml

```
[tool.isort]
profile = "black"
line_length = 100
src_paths = ["src", "tests"]
skip = ["migrations"]
```

CI gate

```
isort --check-only --diff .
```

Sort only changed files

```
git diff --name-only --diff-filter=d -- '*.py' | xargs isort
```

Pre-commit hook

```
# .pre-commit-config.yaml
repos:
  - repo: https://github.com/PyCQA/isort
    rev: 9.0.2
    hooks:
      - id: isort
        args: ["--profile", "black"]
```