Black Python formatter

CONCEPTS

Black is opinionated: few options, same output for everyone. Pin the version in your project, since style can change between releases.

Source files .py .pyi .ipynb
↓ parse
Black config + CLI flags
↓ reformat
Equivalence check AST must match
↓ write
Formatted files exit 0 / 1 / 123

Precedence: built-in defaults < pyproject.toml < command-line flags.

RUN

black .
Format all files under the current dir

black *<file.py> <dir>*
Format given files and folders

black --check .
Exit 1 if anything would change

black --diff .
Print the changes, write nothing

black --check --diff .
Show what a CI failure is about

black --diff --color .
Colored diff

black -q .
Only print errors

black -v .
Report ignored and unchanged files

black --version
Print the installed version

STYLE OPTIONS

black -l *<n>* **.**
Maximum line length (default 88)

black -t *<py311>* **.**
Target Python version (repeatable)

black -S .
Keep string quotes as they are

black -C .
Ignore magic trailing commas

FILES AND STDIN

black -
Read stdin, write formatted code to stdout

black -c "*<code>***"**
Format a code string

black --stdin-filename *<path>* **-**
Name stdin so excludes apply

black --extend-exclude '*<regex>***' .**
Skip more paths, keep defaults

black --exclude '*<regex>***' .**
Replace the default exclusions

black --force-exclude '*<regex>***' .**
Exclude even explicitly passed files

CONFIG

[tool.black]
Settings table in pyproject.toml

black --config *<file>* **.**
Use exactly this config file

black --required-version *<26>* **.**
Fail if Black's version differs

Common pyproject keys

Key	Same as
`line-length`	-l
`target-version`	-t (a list)
`skip-string-normalization`	-S
`extend-exclude`	--extend-exclude
`required-version`	--required-version

IGNORING CODE

 # fmt: skip
Leave this line alone

 # fmt: off
Start an unformatted region

 # fmt: on
End the region

EXIT CODES

Code	Meaning
`0`	Done, or nothing to change with --check
`1`	--check: files would be reformatted
`123`	Internal error