# Pylint

<https://pylint.pycqa.org/en/latest/tutorial.html>

## Bug Types

Pylint can detect most of Python static code defects. It also checks coding style and spelling in comments.

For a full explanation of all Pylint defects, check:

* <https://pylint.readthedocs.io/en/latest/user_guide/messages/index.html>
* <https://github.com/britonad/pylint-errors>

## Coding Standards

## Usage

### General

The pylint CLI offers a wide range of commands and options:

**usage: pylint [options]**

**options:**

**-h, --help** show this help message and exit

**Commands:**

Options which are actually commands. Options in this group are mutually exclusive.

**--rcfile RCFILE** Specify a configuration file to load.

**--output OUTPUT** Specify an output file.

**--help-msg HELP\_MSG [HELP\_MSG ...]** Display a help message for the given message id and exit.

The value may be a comma separated list of message ids.

**--list-msgs** Display a list of all pylint's messages divided by whether they're emittable with the given interpreter.

**--list-msgs-enabled** Display a list of what messages are enabled, disabled and non-emittable with the given configuration.

**--list-groups** List pylint's message groups.

**--list-conf-levels** Generate pylint's confidence levels.

**--list-extensions** List available extensions.

**--full-documentation** Generate pylint's full documentation.

**--generate-rcfile** Generate a sample configuration file according to the current configuration.

You can put other options before this one to get them in the generated configuration.

**--generate-toml-config**

Generate a sample configuration file according to the current configuration.

You can put other options before this one to get them in the generated configuration.

The config is in the .toml format.

**--long-help** Show more verbose help.

**Main:**

**--init-hook INIT\_HOOK** Python code to execute, usually for sys.path manipulation such as pygtk.require().

**--errors-only, -E** In error mode, messages with a category besides ERROR or FATAL are suppressed,

and no reports are done by default. Error mode is compatible with disabling specific errors.

**--verbose , -v** In verbose mode, extra non-checker-related info will be displayed.

**--enable-all-extensions** Load and enable all available extensions. Use --list-extensions to see a list all available extensions.

**--ignore <file>[,<file>...]** Files or directories to be skipped. They should be base names, not paths. (default: ('CVS',))

**--ignore-patterns <pattern>[,<pattern>...]** Files or directories matching the regular expression patterns are skipped.

The regex matches against base names, not paths. The default value ignores

Emacs file locks (default: (re.compile('^\\.#'),))

**--ignore-paths <pattern>[,<pattern>...]** Add files or directories matching the regular expressions patterns to the ignore-list.

The regex matches against paths and can be in Posix or Windows format.

Because '\\' represents the directory delimiter on Windows systems,

it can't be used as an escape character. (default: [])

**--persistent <y or n>** Pickle collected data for later comparisons. (default: True)

**--load-plugins <modules>**  List of plugins (as comma separated values of python module names) to load,

usually to register additional checkers. (default: ())

**--fail-under <score>** Specify a score threshold under which the program will exit with error. (default: 10)

**--fail-on <msg ids>** Return non-zero exit code if any of these messages/categories are detected,

even if score is above --fail-under value. Syntax same as enable.

Messages specified are enabled, while categories only check already-enabled messages. (default: )

**--jobs <n-processes>, -j <n-processes>**

Use multiple processes to speed up Pylint.

Specifying 0 will auto-detect the number of processors available to use,

and will cap the count on Windows to avoid hangs. (default: 1)

**--limit-inference-results <number-of-results>**

Control the amount of potential inferred values when inferring a single object.

This can help the performance when dealing with large functions or nested conditions. (default: 100)

**--extension-pkg-allow-list <pkg[,pkg]>**

A comma-separated list of package or module names from where C extensions may be loaded.

Extensions are loading into the active Python interpreter and may run arbitrary code. (default: [])

**--extension-pkg-whitelist <pkg[,pkg]>**

A comma-separated list of package or module names from where C extensions may be loaded.

Extensions are loading into the active Python interpreter and may run arbitrary code.

(This is an alternative name to extension-pkg-allow-list for backward compatibility.) (default: [])

**--suggestion-mode <y or n>**

When enabled, pylint would attempt to guess common misconfiguration and emit user-friendly hints

instead of false-positive error messages. (default: True)

**--exit-zero** Always return a 0 (non-error) status code, even if lint errors are found.

This is primarily useful in continuous integration scripts. (default: False)

**--from-stdin** Interpret the stdin as a python script, whose filename needs to be passed

as the module\_or\_package argument. (default: False)

**--source-roots <path>[,<path>...]**

Add paths to the list of the source roots. Supports globbing patterns.

The source root is an absolute path or a path relative to the current working

directory used to determine a package namespace for modules located under the source root. (default: ())

**--recursive <yn>** Discover python modules and packages in the file system subtree. (default: False)

**--py-version <py\_version>**

Minimum Python version to use for version dependent checks.

Will default to the version used to run pylint. (default: (3, 12))

**--ignored-modules <module names>**

List of module names for which member attributes should not be checked

(useful for modules/projects where namespaces are manipulated during runtime and

thus existing member attributes cannot be deduced by static analysis).

It supports qualified module names, as well as Unix pattern matching. (default: ())

**--analyse-fallback-blocks <y or n>**

Analyse import fallback blocks. This can be used to support both Python 2 and 3 compatible code,

which means that the block might have code that exists

only in one or another interpreter, leading to false positives when analysed. (default: False)

**--clear-cache-post-run <y or n>**

Clear in-memory caches upon conclusion of linting. Useful if running pylint in a server-like mode.

(default: False)

**Reports:**

Options related to output formatting and reporting

**--output-format <format>, -f <format>**

Set the output format. Available formats are: text, parseable, colorized, json2

(improved json format), json (old json format) and msvs (visual studio).

You can also give a reporter class, e.g. mypackage.mymodule.MyReporterClass.

**--reports <y or n>, -r <y or n>**

Tells whether to display a full report or only the messages. (default: False)

**--evaluation <python\_expression>**

Python expression which should return a score less than or equal to 10.

You have access to the variables 'fatal', 'error', 'warning', 'refactor',

'convention', and 'info' which contain the number of messages in each category,

as well as 'statement' which is the total number of statements analyzed.

This score is used by the global evaluation report (RP0004).

(default: max(0, 0 if fatal else 10.0 - ((float(5 \* error + warning + refactor +

convention) / statement) \* 10)))

**--score <y or n>, -s <y or n>**

Activate the evaluation score. (default: True)

**--msg-template <template>**

Template used to display messages. This is a python new-style format string used to

format the message information. See doc for all details. (default: )

**Messages control:**

Options controlling analysis messages

**--confidence <levels>**

Only show warnings with the listed confidence levels. Leave empty to show all.

Valid levels: HIGH, CONTROL\_FLOW, INFERENCE, INFERENCE\_FAILURE,

UNDEFINED. (default: ['HIGH', 'CONTROL\_FLOW', 'INFERENCE', 'INFERENCE\_FAILURE', 'UNDEFINED'])

**--enable <msg ids>, -e <msg ids>**

Enable the message, report, category or checker with the given id(s).

You can either give multiple identifier separated by comma (,) or put this option

multiple time (only on the command line, not in the configuration file where it should appear only once).

**--disable <msg ids>, -d <msg ids>**

Disable the message, report, category or checker with the given id(s).

You can either give multiple identifiers separated by comma (,) or put this option multiple times

(only on the command line, not in the configuration file where it should appear only once).

You can also use "--disable=all" to disable everything first and then re-enable specific checks.

For example, if you want to run only the similarities checker,

you can use "--disable=all --enable=similarities".

If you want to run only the classes checker, but have no Warning level messages displayed,

use "--disable=all --enable=classes --disable=W".

**Basic:**

**--good-names <names>** Good variable names which should always be accepted, separated by a comma.

(default: ('i', 'j', 'k', 'ex', 'Run', '\_'))

**--good-names-rgxs <names>**

Good variable names regexes, separated by a comma. If names match any regex,

they will always be accepted (default: )

**--bad-names <names>** Bad variable names which should always be refused, separated by a comma.

(default: ('foo', 'bar', 'baz', 'toto', 'tutu', 'tata'))

**--bad-names-rgxs <names>**

Bad variable names regexes, separated by a comma. If names match any regex,

they will always be refused (default: )

**--name-group <name1:name2>**

Colon-delimited sets of names that determine each other's naming style

when the name regexes allow several styles. (default: ())

**--include-naming-hint <y or n>**

Include a hint for the correct naming format with invalid-name. (default: False)

**--property-classes <decorator names>**

List of decorators that produce properties, such as abc.abstractproperty.

Add to this list to register other decorators that produce valid properties.

These decorators are taken in consideration only for invalid-name. (default: ('abc.abstractproperty',))

**--argument-naming-style <style>**

Naming style matching correct argument names. (default: snake\_case)

**--argument-rgx <regexp>**

Regular expression matching correct argument names. Overrides argument-naming-style.

If left empty, argument names will be checked with the set naming style. (default: None)

**--attr-naming-style <style>**

Naming style matching correct attribute names. (default: snake\_case)

**--attr-rgx <regexp>** Regular expression matching correct attribute names. Overrides attr-naming-style.

If left empty, attribute names will be checked with the set naming style. (default: None)

**--class-naming-style <style>**

Naming style matching correct class names. (default: PascalCase)

**--class-rgx <regexp>** Regular expression matching correct class names. Overrides class-naming-style.

If left empty, class names will be checked with the set naming style. (default: None)

**--class-attribute-naming-style <style>**

Naming style matching correct class attribute names. (default: any)

**--class-attribute-rgx <regexp>**

Regular expression matching correct class attribute names. Overrides class-attribute-naming-style.

If left empty, class attribute names will be checked with the set naming style. (default: None)

**--class-const-naming-style <style>**

Naming style matching correct class constant names. (default: UPPER\_CASE)

**--class-const-rgx <regexp>**

Regular expression matching correct class constant names.

Overrides class-const-naming-style. If left empty, class constant names will be checked with

the set naming style. (default: None)

**--const-naming-style <style>**

Naming style matching correct constant names. (default: UPPER\_CASE)

**--const-rgx <regexp>** Regular expression matching correct constant names. Overrides const-naming-style.

If left empty, constant names will be checked with the set naming style. (default: None)

**--function-naming-style <style>**

Naming style matching correct function names. (default: snake\_case)

**--function-rgx <regexp>**

Regular expression matching correct function names. Overrides function-naming-style.

If left empty, function names will be checked with the set naming style. (default: None)

**--inlinevar-naming-style <style>**

Naming style matching correct inline iteration names. (default: any)

**--inlinevar-rgx <regexp>**

Regular expression matching correct inline iteration names. Overrides inlinevar-naming-style.

If left empty, inline iteration names will be checked with the set naming style. (default: None)

**--method-naming-style <style>**

Naming style matching correct method names. (default: snake\_case)

**--method-rgx <regexp>**

Regular expression matching correct method names. Overrides method-naming-style.

If left empty, method names will be checked with the set naming style. (default: None)

**--module-naming-style <style>**

Naming style matching correct module names. (default: snake\_case)

**--module-rgx <regexp>**

Regular expression matching correct module names. Overrides module-naming-style.

If left empty, module names will be checked with the set naming style. (default: None)

**--typealias-rgx <regexp>**

Regular expression matching correct type alias names.

If left empty, type alias names will be checked with the set naming style. (default: None)

**--typevar-rgx <regexp>**

Regular expression matching correct type variable names.

If left empty, type variable names will be checked with the set naming style. (default: None)

**--variable-naming-style <style>**

Naming style matching correct variable names. (default: snake\_case)

**--variable-rgx <regexp>**

Regular expression matching correct variable names. Overrides variable-naming-style.

If left empty, variable names will be checked with the set naming style. (default: None)

**--no-docstring-rgx <regexp>**

Regular expression which should only match function or class names that do not require a docstring.

(default: re.compile('^\_'))

**--docstring-min-length <int>**

Minimum line length for functions/classes that require docstrings, shorter ones are exempt.

(default: -1)

**Classes:**

Checker for class nodes.

**--defining-attr-methods <method names>**

List of method names used to declare (i.e. assign) instance attributes.

(default: ('\_\_init\_\_', '\_\_new\_\_', 'setUp', 'asyncSetUp', '\_\_post\_init\_\_'))

**--valid-classmethod-first-arg <argument names>**

List of valid names for the first argument in a class method. (default: ('cls',))

**--valid-metaclass-classmethod-first-arg <argument names>**

List of valid names for the first argument in a metaclass class method. (default: ('mcs',))

**--exclude-protected <protected access exclusions>**

List of member names, which should be excluded from the protected access warning.

(default: ('\_asdict', '\_fields', '\_replace', '\_source', '\_make', 'os.\_exit'))

**--check-protected-access-in-special-methods <y or n>**

Warn about protected attribute access inside special methods (default: False)

**Design:**

Checker of potential misdesigns.

**--max-args <int>** Maximum number of arguments for function / method. (default: 5)

**--max-locals <int>** Maximum number of locals for function / method body. (default: 15)

**--max-returns <int>** Maximum number of return / yield for function / method body. (default: 6)

**--max-branches <int>** Maximum number of branch for function / method body. (default: 12)

**--max-statements <int>** Maximum number of statements in function / method body. (default: 50)

--max-parents <num> Maximum number of parents for a class (see R0901). (default: 7)

**--ignored-parents <comma separated list of class names>**

List of qualified class names to ignore when counting class parents (see R0901) (default: ())

**--max-attributes <num>** Maximum number of attributes for a class (see R0902). (default: 7)

**--min-public-methods <num>** Minimum number of public methods for a class (see R0903). (default: 2)

**--max-public-methods <num>** Maximum number of public methods for a class (see R0904). (default: 20)

**--max-bool-expr <num>** Maximum number of boolean expressions in an if statement (see R0916). (default: 5)

**--exclude-too-few-public-methods <pattern>[,<pattern>...]**

List of regular expressions of class ancestor names to ignore when counting public methods

(see R0903) (default: [])

**Exceptions:**

Exception related checks.

**--overgeneral-exceptions <comma-separated class names>**

Exceptions that will emit a warning when caught.

(default: ('builtins.BaseException', 'builtins.Exception'))

**Format:**

Formatting checker.

**--max-line-length <int>**

Maximum number of characters on a single line. (default: 100)

**--ignore-long-lines <regexp>**

Regexp for a line that is allowed to be longer than the limit. (default: ^\s\*(# )?<?https?://\S+>?$)

**--single-line-if-stmt <y or n>**

Allow the body of an if to be on the same line as the test if there is no else. (default: False)

**--single-line-class-stmt <y or n>**

Allow the body of a class to be on the same line as the declaration if body contains single statement.

(default: False)

**--max-module-lines <int>**

Maximum number of lines in a module. (default: 1000)

**--indent-string <string>**

String used as indentation unit. This is usually " " (4 spaces) or "\t" (1 tab). (default: )

**--indent-after-paren <int>**

Number of spaces of indent required inside a hanging or continued line. (default: 4)

**--expected-line-ending-format <empty or LF or CRLF>**

Expected format of line ending, e.g. empty (any line ending), LF or CRLF. (default: )

**Imports:**

BaseChecker for import statements.

**--deprecated-modules <modules>**

Deprecated modules which should not be used, separated by a comma. (default: ())

**--preferred-modules <module:preferred-module>**

Couples of modules and preferred modules, separated by a comma. (default: ())

**--import-graph <file.gv>**

Output a graph (.gv or any supported image format) of all (i.e. internal and external)

dependencies to the given file (report RP0402 must not be disabled). (default: )

**--ext-import-graph <file.gv>**

Output a graph (.gv or any supported image format) of external dependencies to the given file

(report RP0402 must not be disabled). (default: )

**--int-import-graph <file.gv>**

Output a graph (.gv or any supported image format) of internal dependencies to the given file

(report RP0402 must not be disabled). (default: )

**--known-standard-library <modules>**

Force import order to recognize a module as part of the standard compatibility libraries. (default: ())

**--known-third-party <modules>**

Force import order to recognize a module as part of a third party library. (default: ('enchant',))

**--allow-any-import-level <modules>**

List of modules that can be imported at any level, not just the top level one. (default: ())

**--allow-wildcard-with-all <y or n>**

Allow wildcard imports from modules that define \_\_all\_\_. (default: False)

**--allow-reexport-from-package <y or n>**

Allow explicit reexports by alias from a package \_\_init\_\_. (default: False)

**Logging:**

Checks use of the logging module.

**--logging-modules <comma separated list>**

Logging modules to check that the string format arguments are in logging function parameter format.

(default: ('logging',))

**--logging-format-style <old (%) or new ({)>**

The type of string formatting that logging methods do. `old` means using % formatting,

`new` is for `{}` formatting. (default: old)

**Method\_args:**

BaseChecker for method\_args.

**--timeout-methods <comma separated list>**

List of qualified names (i.e., library.method)

which require a timeout parameter e.g. 'requests.api.get,requests.api.post' (default:

('requests.api.delete', 'requests.api.get', 'requests.api.head', 'requests.api.options',

'requests.api.patch', 'requests.api.post', 'requests.api.put', 'requests.api.request'))

**Miscellaneous:**

BaseChecker for encoding issues.

**--notes <comma separated values>**

List of note tags to take in consideration, separated by a comma. (default: ('FIXME', 'XXX', 'TODO'))

**--notes-rgx <regexp>** Regular expression of note tags to take in consideration. (default: )

**Refactoring:**

Looks for code which can be refactored.

**--max-nested-blocks <int>**

Maximum number of nested blocks for function / method body (default: 5)

**--never-returning-functions <members names>**

Complete name of functions that never returns. When checking for inconsistent-return-statements

if a never returning function is called then it will be

considered as an explicit return statement and no message will be printed.

(default: ('sys.exit', 'argparse.parse\_error'))

**Similarities:**

Checks for similarities and duplicated code.

**--min-similarity-lines <int>** Minimum lines number of a similarity. (default: 4)

**--ignore-comments <y or n>** Comments are removed from the similarity computation (default: True)

**--ignore-docstrings <y or n>** Docstrings are removed from the similarity computation (default: True)

**--ignore-imports <y or n>** Imports are removed from the similarity computation (default: True)

**--ignore-signatures <y or n>** Signatures are removed from the similarity computation (default: True)

**Spelling:**

Check spelling in comments and docstrings.

**--spelling-dict <d**ict name> Spelling dictionary name. No available dictionaries :

You need to install both the python package and the system dependency for enchant to work.

(default: )

**--spelling-ignore-words <comma separated words>**

List of comma separated words that should not be checked. (default: )

**--spelling-private-dict-file <path to file>**

A path to a file that contains the private dictionary; one word per line. (default: )

**--spelling-store-unknown-words <y or n>**

Tells whether to store unknown words to the private dictionary

(see the --spelling-private-dict-file option) instead of raising a message. (default: n)

**--max-spelling-suggestions N**

Limits count of emitted suggestions for spelling mistakes. (default: 4)

**--spelling-ignore-comment-directives <comma separated words>**

List of comma separated words that should be considered directives if they appear

at the beginning of a comment and should not be checked.

(default: fmt: on,fmt: off,noqa:,noqa,nosec,isort:skip,mypy:)

**String:**

Check string literals.

**--check-str-concat-over-line-jumps <y or n>**

This flag controls whether the implicit-str-concat should generate a warning on

implicit string concatenation in sequences defined over several lines. (default: False)

**--check-quote-consistency <y or n>**

This flag controls whether inconsistent-quotes generates a warning when the character

used as a quote delimiter is used inconsistently within a module. (default: False)

**Typecheck:**

Try to find bugs in the code using type inference.

**--ignore-on-opaque-inference <y or n>**

This flag controls whether pylint should warn about no-member and similar checks whenever

an opaque object is returned when inferring. The inference can

return multiple potential results while evaluating a Python object,

but some branches might not be evaluated, which results in partial inference. In

that case, it might be useful to still emit no-member and other checks for the rest of

the inferred objects. (default: True)

**--mixin-class-rgx <regexp>**

Regex pattern to define which classes are considered mixins. (default: .\*[Mm]ixin)

**--ignore-mixin-members <y or n>**

Tells whether missing members accessed in mixin class should be ignored.

A class is considered mixin if its name matches the mixin-class-rgx option.

(default: True)

**--ignored-checks-for-mixins <list of messages names>**

List of symbolic message names to ignore for Mixin members.

(default: ['no-member', 'not-async-context-manager', 'not-context-manager', 'attribute-

defined-outside-init'])

**--ignore-none <y or n>**

Tells whether to warn about missing members when the owner of the attribute is inferred to be None.

(default: True)

**--ignored-classes <members names>**

List of class names for which member attributes should not be checked

(useful for classes with dynamically set attributes). This supports the use of

qualified names. (default: ('optparse.Values', 'thread.\_local', '\_thread.\_local', 'argparse.Namespace'))

**--generated-members <members names>**

List of members which are set dynamically and missed by pylint inference system,

and so shouldn't trigger E1101 when accessed. Python regular

expressions are accepted. (default: ())

**--contextmanager-decorators <decorator names>**

List of decorators that produce context managers, such as contextlib.contextmanager.

Add to this list to register other decorators that produce valid

context managers. (default: ['contextlib.contextmanager'])

**--missing-member-hint-distance <member hint edit distance>**

The minimum edit distance a name should have in order to be considered a similar match for

a missing member name. (default: 1)

**--missing-member-max-choices <member hint max choices>**

The total number of similar names that should be taken in consideration when showing

a hint for a missing member. (default: 1)

**--missing-member-hint <missing member hint>**

Show a hint with possible names when a member name was not found. The aspect of finding

the hint is based on edit distance. (default: True)

**--signature-mutators <decorator names>**

List of decorators that change the signature of a decorated function. (default: [])

**Variables:**

BaseChecker for variables.

**--init-import <y or n>**

Tells whether we should check for unused import in \_\_init\_\_ files. (default: False)

**--dummy-variables-rgx <regexp>**

A regular expression matching the name of dummy variables (i.e. expected to not be used).

(default:\_+$|(\_[a-zA-Z0-9\_]\*[a-zA-Z0-9]+?$)|dummy|^ignored\_|^unused\_)

**--additional-builtins <comma separated list>**

List of additional names supposed to be defined in builtins.

Remember that you should avoid defining new builtins when possible. (default: ())

**--callbacks <callbacks>**

List of strings which can identify a callback function by name.

A callback name must start or end with one of those strings. (default: ('cb\_', '\_cb'))

**--redefining-builtins-modules <comma separated list>**

List of qualified module names which can have objects that can redefine builtins.

(default: ('six.moves', 'past.builtins', 'future.builtins',

'builtins', 'io'))

**--ignored-argument-names <regexp>**

Argument names that match this expression will be ignored. (default: re.compile('\_.\*|^ignored\_|^unused\_'))

**--allow-global-unused-variables <y or n>**

Tells whether unused global variables should be treated as a violation. (default: True)

**--allowed-redefined-builtins <comma separated list>**

List of names allowed to shadow builtins (default: ())

### Suppress certain checks

#### Command Line

The --disable <msg ids>, -d <msg ids> option is used to specify suppressions on the command line.

#### In-File

Check session [Import Configuration File](#_Import_Configuration_File).

#### In-Line

Suppressions can also be added directly in the code by adding comments that contain special keywords.

**On a single line:**

a, b = ... # pylint: disable=unbalanced-tuple-unpacking

**On the following line:**

# pylint: disable-next=unbalanced-tuple-unpacking

a, b = ...

**In a single scope:**

def test():

    # Disable all the no-member violations in this function

    # pylint: disable=no-member

    ...

**In a bloc**k:

def func(self):

    if self.bar:

        # pylint: enable=no-member

        # enable all no-members for this block

        print(self.foo)

    else:

        # This is affected by the scope disable

        print(self.foo)

Details: <https://pylint.readthedocs.io/en/latest/user_guide/messages/message_control.html>

### Import Configuration File

If you try of writing lengthy commands again and again, you can create a config file called .pylintrc.

About where to put the file, there are multiple options:

* /etc/.pylintrc for default global configuration
* ~/.pylintrc for default user configuration
* <your\_project>/.pylintrc for default project configuration (used when you'll run $ pylint <your-project>)
* Wherever you want, then use $ pylint --rcfile=<wherever\_I\_want>

**Tips**:

* Sample configuration file: <https://www.codeac.io/documentation/pylint-configuration.html>
* You can **generate configuration files** with --generate-rcfile or --generate-toml-config. Every option present on the command line before this will be included in the file. For example:

$ pylint --disable=bare-except,invalid-name --class-rgx='[A-Z][a-z]+' --generate-toml-config

* My simple configuration file:

[MESSAGES CONTROL]

# Disable the message, report, category or checker with the given id(s)

disable = missing-module-docstring,

          missing-class-docstring,

          missing-function-docstring,

          too-few-public-methods,

          format,

          fixme,

          broad-exception-raised,

          broad-exception-caught,

          invalid-name,

[FORMAT]

# Maximum number of characters on a single line.

max-line-length=160

# Maximum number of lines in a module

max-module-lines=1000

# String used as indentation unit. This is usually "    " (4 spaces) or "\t" (1 tab).

indent-string='    '

### Install Plugins

## Tips

### VS Code Extension

Microsoft has an [official extension for Pylint](https://marketplace.visualstudio.com/items?itemName=ms-python.pylint).

After installing the extension, you need to modify VSCode’s User Settings so that the extension can work correctly as your configigurations. Basically, following items are considered:

// Arguments passed to Pylint for linting Python files. Each argument should be provided as a separate string in the array.

//  Examples:

// - `"pylint.args": ["--rcfile=<file>"]`       # Path to .pylintrc file

"pylint.args": ["--rcfile=D:/MYGIT/Personal/ShopeeSalesHunter/src/.pylintrc"],

// Enable linting Python files with Pylint as you type.

"pylint.lintOnChange": false,

#### Notes

* **Used with Virtual Environment:**

The Pylint extension may assume your project modules were standard library modules, and refusing to perform any linting.

You can resolve this issue following one of these solutions:

1. Make sure you installed pylint in your virtual environment. In User Setting, set pylint.interpreter to the default Python interpreter (not the Virtual Environment one).
2. From Command Palette, choose *Python: Select Interpreter*, and select the default Python interpreter.