WSCheck Documentation

Release 1.3.0

Andras Tim

Nov 09, 2018

Contents

WSCheck is a static analysis tool for whitespaces.

CHAPTER 1

Installation

```
$ pip install wscheck
```

CHAPTER 2

Usage

Check multiple files:

```
$ wscheck orange.sh pineapple.xml kiwi.js
```

Exclude rules:

```
$ wscheck --exclude WSC002 --exclude WSC003 orange.sh
```

Get list of available rules:

```
$ wscheck --list-rules
```

For details about rules, see Rules

Write CheckStyle output too:

```
$ wscheck --checkstyle results.xml pineapple.xml
```

CHAPTER 3

Example

```
$ wscheck examples/multiple_problems.py
```

```
In examples/multiple_problems.py line 2:
class LabelPrinter:
^-- WSC007: File begins with newline

In examples/multiple_problems.py line 6:
        self.print_to_pdf()
                            ^-- WSC002: Tailing whitespace

In examples/multiple_problems.py line 9:
   def __generate_pdf(self):
   ^-- WSC003: Indentation is not multiple of 2

In examples/multiple_problems.py line 10:
        pdf_generator = _LabelPdfGenerator()
                                             ^-- WSC001: Bad line ending '\r\n'

In examples/multiple_problems.py line 16:
--->--->os.makedirs(self.__cache_dir, exist_ok=True)
^-- WSC004: Indentation with non-space character

In examples/multiple_problems.py line 22:
        return os.path.join(self.__cache_dir, pdf_name)
                                                        ^-- WSC006: Too many␣
↪newline at end of file (+1)
```

CHAPTER 4

Bugs

Bugs or suggestions? Visit the issue tracker.

CHAPTER 5

Benchmark

- You can run a quick benchmark:

```
tox -- tests/performance --quick-benchmark
```

- You can run benchmarks and generate histogram for compare calls to each other:

```
tox -- tests/performance --benchmark-histogram
```

- You can run benchmarks and save results for later compare:

```
tox -- tests/performance --benchmark-save=foo
```

- You can run benchmarks and compare with the last saved result with fail treshold:

```
tox -- tests/performance --benchmark-histogram --benchmark-compare --
↪benchmark-compare-fail=mean:5% --benchmark-sort=name
```

- You can run benchmarks and compare with the last saved result by groups:

```
tox -- tests/performance --benchmark-histogram --benchmark-compare --
↪benchmark-group-by=func

tox -- tests/performance --benchmark-histogram --benchmark-compare --
↪benchmark-group-by=name
```

CHAPTER 6

Topics

6.1 Rules

6.1.1 WSC001: Bad line ending

This rule enforces Linux style (`\n`) line ending, and alerting for Windows (`\r\n`) and Osx (`\r`) style.

Example

Input file: `examples/WSC001_bad_eol.py`

```
class LabelPrinter:
    def __generate_pdf(self):
        pdf_generator = _LabelPdfGenerator()
        pdf_generator.generate_label(
            self.__title, self.__data, self.__logo_path, config.App.LABEL_BORDER,
            output_path=self.__pdf_path)
```

Command:

```
$ wscheck 'examples/WSC001_bad_eol.py'
```

```
In examples/WSC001_bad_eol.py line 3:
        pdf_generator = _LabelPdfGenerator()
                                           ^-- WSC001: Bad line ending '\r\n'

In examples/WSC001_bad_eol.py line 5:
            self.__title, self.__data, self.__logo_path, config.App.LABEL_BORDER,
                                                                                ^-
↪- WSC001: Bad line ending '\r'
```

6.1.2 WSC002: Tailing whitespace

Alerting for left spaces, tabulators at end of lines.

Example

Input file: `examples/WSC002_tailing_ws.py`

```
class LabelPrinter:
    def print(self, printer: (Printer, None)=None, copies: int=1):
        printer = printer or Printer(config.App.LABEL_PRINTER)

        self.print_to_pdf()
        printer.print_pdf(self.__pdf_path, options={'copies': str(copies)})
```

Command:

```
$ wscheck 'examples/WSC002_tailing_ws.py'
```

```
In examples/WSC002_tailing_ws.py line 3:
        printer = printer or Printer(config.App.LABEL_PRINTER)
                                                              ^-- WSC002: Tailing␣
↪whitespace

In examples/WSC002_tailing_ws.py line 5:
        self.print_to_pdf()
                             ^-- WSC002: Tailing whitespace
```

6.1.3 WSC003: Indentation is not multiple of 2

This rule alerting for indentation (whitespaces before the line) must be multiple of 2 spaces (or zero).

Example

Input file: `examples/WSC003_bad_indentation.py`

```
class LabelPrinter:
   def __generate_pdf(self):
        pdf_generator = _LabelPdfGenerator()
        pdf_generator.generate_label(
            self.__title, self.__data, self.__logo_path, config.App.LABEL_BORDER,
            output_path=self.__pdf_path)
```

Command:

```
$ wscheck 'examples/WSC003_bad_indentation.py'
```

```
In examples/WSC003_bad_indentation.py line 2:
   def __generate_pdf(self):
   ^-- WSC003: Indentation is not multiple of 2
```

6.1.4 WSC004: Indentation with non-space character

This rule enforces the space indentation (whitespaces before the line can be spaces only).

Example

Input file: `examples/WSC004_tab_indentation.py`

```
class LabelPrinter:
    def __prepare_print_cache_dir(self):
                os.makedirs(self.__print_cache_dir, exist_ok=True)
```

Command:

```
$ wscheck 'examples/WSC004_tab_indentation.py'
```

```
In examples/WSC004_tab_indentation.py line 3:
--->--->os.makedirs(self.__print_cache_dir, exist_ok=True)
^-- WSC004: Indentation with non-space character
```

6.1.5 WSC005: No newline at end of file

This rule enforces one `\n` at end of file.

Example

Input file: `examples/WSC005_no_new_line_at_eof.py`

```
class LabelPrinter:
    def print(self, printer: (Printer, None)=None, copies: int=1):
        printer = printer or Printer(config.App.LABEL_PRINTER)

        self.print_to_pdf()
        printer.print_pdf(self.__pdf_path, options={'copies': str(copies)})
```

Command:

```
$ wscheck 'examples/WSC005_no_new_line_at_eof.py'
```

```
In examples/WSC005_no_new_line_at_eof.py line 6:
        printer.print_pdf(self.__pdf_path, options={'copies': str(copies)})
                                                                          ^--␣
↪WSC005: No newline at end of file
```

6.1.6 WSC006: Too many newlines at the end of file

This rule enforces one `\n` at end of file.

Example

Input file: `examples/WSC006_too_many_new_lines_at_eof.py`

```
class LabelPrinter:
    def __get_pdf_path(self) -> str:
        pdf_name = self.__pdf_name_template.format(
            data=self.__data,
            title_hash=hashlib.sha1(self.__title.encode()).hexdigest())
        return os.path.join(self.__print_cache_dir, pdf_name)
```

Command:

```
$ wscheck 'examples/WSC006_too_many_new_lines_at_eof.py'
```

```
In examples/WSC006_too_many_new_lines_at_eof.py line 6:
        return os.path.join(self.__print_cache_dir, pdf_name)
                                                              ^-- WSC006: Too many␣
↪newline at end of file (+1)
```

6.1.7 WSC007: File begins with newline

Check empty lines before the first non-empty line.

Example

Input file: `examples/WSC007_new_line_at_bof.py`

```
class LabelPrinter:
    def print(self, printer: (Printer, None)=None, copies: int=1):
        printer = printer or Printer(config.App.LABEL_PRINTER)

        self.print_to_pdf()
        printer.print_pdf(self.__pdf_path, options={'copies': str(copies)})
```

Command:

```
$ wscheck 'examples/WSC007_new_line_at_bof.py'
```

```
In examples/WSC007_new_line_at_bof.py line 2:
class LabelPrinter:
^-- WSC007: File begins with newline
```

6.2 Contributing

6.2.1 Extending rules

Checklist

1. Extend the **RULES** list in `wscheck/checker.py` file with the next ID
2. Write unit tests and production code with TDD
   (a) Extend the `tests/unit/checker/test_rules.py` file with specific unit tests.
   (b) Write the checker in `wscheck/checker.py` file.
   (c) Extend the complex cases with the new rule related things.
3. Extend performance tests in `tests/performance/test_checker_performance.py`
   (a) With a rule specific suite.
   (b) Extend the complex case too.
   (c) Run all performance tests for check performance degradation!
4. Extend documentation
   (a) Create `docs/rules/WSC000.rst` file for describing the rule.
   (b) Write example into `examples/WSC000_foo` and use it in the `.rst`.
   (c) Extend `examples/multiple_problems.py` file with a typical wrong line for demonstrate.

(d) Refresh the output in `README.rst` too.

5. Update changelog
   (a) Extend the link list of rules at the bottom of `CHANGELOG.md`.
   (b) Update the **Unreleased** section of `CHANGELOG.md`, where refers to the rule.

6.2.2 Release

Checklist for release a new version

1. Update the `CHANGELOG.md`
   (a) Move all notes from **Unreleased** section to a new one with version and date.
   (b) Copy and update(!) the diff link for the specified version and the **Unreleaed** too.
2. Update version in `wscheck/version.py` file
3. Run tests with CI
   (a) Push changes of `devel` to remote
   (b) wait for the results of CI
4. Check the package building
   (a) Remove up the `build` directory
   (b) Build a package with `setup.py build`
   (c) Check the package contains in the new `build` directory
5. If all tests are green, lets merge
   (a) Merge `devel` branch with `--no-ff`
   (b) Tag the merge patch
   (c) Push them all
6. Publish
   (a) Check do you are on the tagged patch
   (b) Build and upload package to pypi with `setup.py release`
   (c) Update tag description on GitHub

CHAPTER 7

Indices and tables

- genindex

Index