

---

# **coala Documentation**

***Release 0.9.1***

**The coala Developers**

**Dec 28, 2016**



<b>1</b>	<b>coilib package</b>	<b>3</b>
1.1	Subpackages	3
1.1.1	coilib.bearlib package	3
	Subpackages	3
	Module contents	38
1.1.2	coilib.bears package	40
	Subpackages	40
	Submodules	42
	coilib.bears.BEAR_KIND module	42
	coilib.bears.Bear module	42
	coilib.bears.GlobalBear module	46
	coilib.bears.LocalBear module	46
	Module contents	47
1.1.3	coilib.collecting package	47
	Submodules	47
	coilib.collecting.Collectors module	47
	coilib.collecting.Dependencies module	49
	coilib.collecting.Importers module	49
	Module contents	50
1.1.4	coilib.misc package	50
	Submodules	50
	coilib.misc.Annotations module	50
	coilib.misc.BuildManPage module	51
	coilib.misc.Caching module	51
	coilib.misc.CachingUtilities module	53
	coilib.misc.Compatibility module	55
	coilib.misc.Constants module	55
	coilib.misc.ContextManagers module	55
	coilib.misc.DictUtilities module	56
	coilib.misc.Enum module	56
	coilib.misc.Exceptions module	56
	coilib.misc.MutableValue module	56
	coilib.misc.Shell module	57
	Module contents	58
1.1.5	coilib.output package	58
	Subpackages	58
	Submodules	59
	coilib.output.ConfWriter module	59

	coilib.output.ConsoleInteraction module	60
	coilib.output.Interactions module	65
	coilib.output.JSONEncoder module	65
	Module contents	66
1.1.6	coilib.parsing package	66
	Submodules	66
	coilib.parsing.CliParsing module	66
	coilib.parsing.ConfParser module	67
	coilib.parsing.DefaultArgParser module	67
	coilib.parsing.Globbing module	67
	coilib.parsing.LineParser module	69
	Module contents	69
1.1.7	coilib.processes package	69
	Subpackages	69
	Submodules	70
	coilib.processes.BearRunning module	70
	coilib.processes.CONTROL_ELEMENT module	74
	coilib.processes.LogPrinterThread module	74
	coilib.processes.Processing module	74
	Module contents	78
1.1.8	coilib.results package	78
	Subpackages	78
	Submodules	81
	coilib.results.AbsolutePosition module	81
	coilib.results.Diff module	82
	coilib.results.HiddenResult module	85
	coilib.results.LineDiff module	86
	coilib.results.RESULT_SEVERITY module	86
	coilib.results.Result module	86
	coilib.results.ResultFilter module	87
	coilib.results.SourcePosition module	88
	coilib.results.SourceRange module	89
	coilib.results.TextPosition module	89
	coilib.results.TextRange module	90
	Module contents	90
1.1.9	coilib.settings package	90
	Submodules	90
	coilib.settings.ConfigurationGathering module	90
	coilib.settings.DocstringMetadata module	94
	coilib.settings.FunctionMetadata module	94
	coilib.settings.Section module	96
	coilib.settings.SectionFilling module	97
	coilib.settings.Setting module	98
	Module contents	99
1.1.10	coilib.testing package	99
	Submodules	99
	coilib.testing.BearTestHelper module	99
	coilib.testing.LocalBearTestHelper module	99
	Module contents	101
1.2	Submodules	101
1.3	coilib.coala module	101
1.4	coilib.coala_ci module	101
1.5	coilib.coala_delete_orig module	101
1.6	coilib.coala_format module	101

1.7	coilib.coala_json module . . . . .	101
1.8	coilib.coala_main module . . . . .	101
1.9	coilib.coala_modes module . . . . .	102
1.10	Module contents . . . . .	102
<b>Python Module Index</b>		<b>103</b>



Hey there! You're in the right place if you:

- want to develop coala itself!
- want to develop a bear for coala.

If you're trying to **use** coala, you should have a look at our [user documentation](#) instead.





---

**coalib package**


---

## 1.1 Subpackages

### 1.1.1 coalib.bearlib package

#### Subpackages

#### coalib.bearlib.abstractions package

#### Submodules

#### coalib.bearlib.abstractions.ExternalBearWrap module

coalib.bearlib.abstractions.ExternalBearWrap.**external\_bear\_wrap**(*executable:*  
*str, \*\*options*)

#### coalib.bearlib.abstractions.Linter module

coalib.bearlib.abstractions.Linter.**linter**(*executable:* *str*, *use\_stdin:* *bool* = *False*,  
*use\_stdout:* *bool* = *True*, *use\_stderr:*  
*bool* = *False*, *config\_suffix:* *str* = *'*, *executable\_check\_fail\_info:* *str* = *'*, *prerequisite\_check\_command:* *tuple* = *()*, *output\_format:* (*<class 'str'>*, *None*) = *None*,  
*\*\*options*)

Decorator that creates a `LocalBear` that is able to process results from an external linter tool.

The main functionality is achieved through the `create_arguments()` function that constructs the command-line-arguments that get parsed to your executable.

```
>>> @linter("xlint", output_format="regex", output_regex="...")
... class XLintBear:
...     @staticmethod
...     def create_arguments(filename, file, config_file):
...         return "--lint", filename
```

Requiring settings is possible like in `Bear.run()` with supplying additional keyword arguments (and if needed with defaults).

```
>>> @linter("xlint", output_format="regex", output_regex="...")
... class XLintBear:
...     @staticmethod
...     def create_arguments(filename,
...                           file,
...                           config_file,
...                           lintmode: str,
...                           enable_aggressive_lints: bool=False):
...         arguments = ("--lint", filename, "--mode=" + lintmode)
...         if enable_aggressive_lints:
...             arguments += ("--aggressive",)
...         return arguments
```

Sometimes your tool requires an actual file that contains configuration. `linter` allows you to just define the contents the configuration shall contain via `generate_config()` and handles everything else for you.

```
>>> @linter("xlint", output_format="regex", output_regex="...")
... class XLintBear:
...     @staticmethod
...     def generate_config(filename,
...                          file,
...                          lintmode,
...                          enable_aggressive_lints):
...         modestring = ("aggressive"
...                        if enable_aggressive_lints else
...                        "non-aggressive")
...         contents = ("<xlint>",
...                      "    <mode>" + lintmode + "</mode>",
...                      "    <aggressive>" + modestring + "</aggressive>",
...                      "</xlint>")
...         return "\n".join(contents)
...
...     @staticmethod
...     def create_arguments(filename,
...                           file,
...                           config_file):
...         return "--lint", filename, "--config", config_file
```

As you can see you don't need to copy additional keyword-arguments you introduced from `create_arguments()` to `generate_config()` and vice-versa. `linter` takes care of forwarding the right arguments to the right place, so you are able to avoid signature duplication.

If you override `process_output`, you have the same feature like above (auto-forwarding of the right arguments defined in your function signature).

Note when overriding `process_output`: Providing a single output stream (via `use_stdout` or `use_stderr`) puts the according string attained from the stream into parameter `output`, providing both output streams inputs a tuple with `(stdout, stderr)`. Providing `use_stdout=False` and `use_stderr=False` raises a `ValueError`. By default `use_stdout` is `True` and `use_stderr` is `False`.

Documentation: Bear description shall be provided at class level. If you document your additional parameters inside `create_arguments`, `generate_config` and `process_output`, beware that conflicting documentation between them may be overridden. Document duplicated parameters inside `create_arguments` first, then in `generate_config` and after that inside `process_output`.

For the tutorial see: [http://coala.readthedocs.io/en/latest/Developers/Writing\\_Linter\\_Bears.html](http://coala.readthedocs.io/en/latest/Developers/Writing_Linter_Bears.html)

## Parameters

- **executable** – The linter tool.
- **use\_stdin** – Whether the input file is sent via stdin instead of passing it over the command-line-interface.
- **use\_stdout** – Whether to use the stdout output stream.
- **use\_stderr** – Whether to use the stderr output stream.
- **config\_suffix** – The suffix-string to append to the filename of the configuration file created when `generate_config` is supplied. Useful if your executable expects getting a specific file-type with specific file-ending for the configuration file.
- **executable\_check\_fail\_info** – Information that is provided together with the fail message from the normal executable check. By default no additional info is printed.
- **prerequisite\_check\_command** – A custom command to check for when `check_prerequisites` gets invoked (via `subprocess.check_call()`). Must be an Iterable.
- **prerequisite\_check\_fail\_message** – A custom message that gets displayed when `check_prerequisites` fails while invoking `prerequisite_check_command`. Can only be provided together with `prerequisite_check_command`.
- **output\_format** – The output format of the underlying executable. Valid values are
  - None: Define your own format by overriding `process_output`. Overriding `process_output` is then mandatory, not specifying it raises a `ValueError`.
  - 'regex': Parse output using a regex. See parameter `output_regex`.
  - 'corrected': The output is the corrected of the given file. Diffs are then generated to supply patches for results.

Passing something else raises a `ValueError`.

- **output\_regex** – The regex expression as a string that is used to parse the output generated by the underlying executable. It should use as many of the following named groups (via `(?P<name>...)`) to provide a good result:
  - line - The line where the issue starts.
  - column - The column where the issue starts.
  - end\_line - The line where the issue ends.
  - end\_column - The column where the issue ends.
  - severity - The severity of the issue.
  - message - The message of the result.
  - origin - The origin of the issue.
  - additional\_info - Additional info provided by the issue.

The groups `line`, `column`, `end_line` and `end_column` don't have to match numbers only, they can also match nothing, the generated `Result` is filled automatically with `None` then for the appropriate properties.

Needs to be provided if `output_format` is 'regex'.

- **severity\_map** – A dict used to map a severity string (captured from the `output_regex` with the named group `severity`) to an actual

`coala.results.RESULT_SEVERITY` for a result. Severity strings are mapped **case-insensitive!**

- `RESULT_SEVERITY.MAJOR`: Mapped by `critical`, `c`, `fatal`, `fail`, `f`, `“error“`, `err` or `e`.
- `RESULT_SEVERITY.NORMAL`: Mapped by `warning`, `warn` or `w`.
- `RESULT_SEVERITY.INFO``: Mapped by ```information`, `info`, `i`, `note` or `suggestion`.

A `ValueError` is raised when the named group `severity` is not used inside `output_regex` and this parameter is given.

- **diff\_severity** – The severity to use for all results if `output_format` is `‘corrected’`. By default this value is `coala.results.RESULT_SEVERITY.NORMAL`. The given value needs to be defined inside `coala.results.RESULT_SEVERITY`.
- **result\_message** – The message-string to use for all results. Can be used only together with `corrected` or `regex` output format. When using `corrected`, the default value is `“Inconsistency found.”`, while for `regex` this static message is disabled and the message matched by `output_regex` is used instead.
- **diff\_distance** – Number of unchanged lines that are allowed in between two changed lines so they get yielded as one diff if `corrected` output-format is given. If a negative distance is given, every change will be yielded as an own diff, even if they are right beneath each other. By default this value is 1.

#### Raises

- **ValueError** – Raised when invalid options are supplied.
- **TypeError** – Raised when incompatible types are supplied. See parameter documentations for allowed types.

**Returns** A `LocalBear` derivation that lints code using an external tool.

## coala.bearlib.abstractions.SectionCreatable module

**class** `coala.bearlib.abstractions.SectionCreatable.SectionCreatable`

Bases: `object`

A `SectionCreatable` is an object that is creatable out of a section object. Thus this is the class for many helper objects provided by the `bearlib`.

If you want to use an object that inherits from this class the following approach is recommended: Instantiate it via the `from_section` method. You can provide default arguments via the lower case keyword arguments.

Example:

```
SpacingHelper.from_section(section, tabwidth=8)
```

creates a `SpacingHelper` and if the “tabwidth” setting is needed and not contained in section, 8 will be taken.

It is recommended to write the prototype of the `__init__` method according to this example:

```
def __init__(self, setting_one: int, setting_two: bool=False):
    pass # Implementation
```

This way the `get_optional_settings` and the `get_non_optional_settings` method will extract automatically that:

- `setting_one` should be an integer
- `setting_two` should be a bool and defaults to False

If you write a documentation comment, you can use `:param` to add descriptions to your parameters. These will be available too automatically.

**classmethod** `from_section` (*section*, *\*\*kwargs*)

Creates the object from a section object.

#### Parameters

- **section** – A section object containing at least the settings specified by `get_non_optional_settings()`
- **kwargs** – Additional keyword arguments

**classmethod** `get_metadata` ()

**classmethod** `get_non_optional_settings` ()

Retrieves the minimal set of settings that need to be defined in order to use this object.

**Returns** a dictionary of needed settings as keys and help texts as values

**classmethod** `get_optional_settings` ()

Retrieves the settings needed IN ADDITION to the ones of `get_non_optional_settings` to use this object without internal defaults.

**Returns** a dictionary of needed settings as keys and help texts as values

## Module contents

The abstractions package contains classes that serve as interfaces for helper classes in the bearlib.

### coala.bearlib.aspects package

#### Submodules

#### coala.bearlib.aspects.Metadata module

**class** `coala.bearlib.aspects.Metadata.Body` (*language*, *\*\*taste\_values*)

Bases: `coala.bearlib.aspects.Metadata.Body`, `coala.bearlib.aspects.base.aspectbase`

**class** `Existence` (*language*, *\*\*taste\_values*)

Bases: `coala.bearlib.aspects.Metadata.Existence`, `coala.bearlib.aspects.base.aspectbase`

**docs** = <coala.bearlib.aspects.docs.Documentation object>

**parent**

alias of `Body`

**subaspects** = {}

**class** `Body.Length` (*language*, *\*\*taste\_values*)

Bases: `coala.bearlib.aspects.Metadata.Length`, `coala.bearlib.aspects.base.aspectbase`

**docs** = <coala.bearlib.aspects.docs.Documentation object>

```

    parent
        alias of Body

    subspects = {}

Body.docs = <coilib.bearlib.aspects.docs.Documentation object>

Body.parent
    alias of CommitMessage

Body.subspects = {'Length': <aspectclass 'Root.Metadata.CommitMessage.Body.Length'>, 'Existence': <aspectclass 'Root.Metadata.CommitMessage.Body.Existence'>}

class coilib.bearlib.aspects.Metadata.ColonExistence (language, **taste_values)
    Bases:
        coilib.bearlib.aspects.Metadata.ColonExistence,
        coilib.bearlib.aspects.base.aspectbase

    docs = <coilib.bearlib.aspects.docs.Documentation object>

    parent
        alias of Shortlog

    subspects = {}

class coilib.bearlib.aspects.Metadata.CommitMessage (language, **taste_values)
    Bases:
        coilib.bearlib.aspects.Metadata.CommitMessage,
        coilib.bearlib.aspects.base.aspectbase

    class Body (language, **taste_values)
        Bases: coilib.bearlib.aspects.Metadata.Body, coilib.bearlib.aspects.base.aspectbase

        class Existence (language, **taste_values)
            Bases:
                coilib.bearlib.aspects.Metadata.Existence,
                coilib.bearlib.aspects.base.aspectbase

            docs = <coilib.bearlib.aspects.docs.Documentation object>

            parent
                alias of Body

            subspects = {}

        class CommitMessage.Body.Length (language, **taste_values)
            Bases: coilib.bearlib.aspects.Metadata.Length, coilib.bearlib.aspects.base.aspectbase

            docs = <coilib.bearlib.aspects.docs.Documentation object>

            parent
                alias of Body

            subspects = {}

        CommitMessage.Body.docs = <coilib.bearlib.aspects.docs.Documentation object>

        CommitMessage.Body.parent
            alias of CommitMessage

        CommitMessage.Body.subspects = {'Length': <aspectclass 'Root.Metadata.CommitMessage.Body.Length'>, 'Existence': <aspectclass 'Root.Metadata.CommitMessage.Body.Existence'>}

class CommitMessage.Emptyiness (language, **taste_values)
    Bases: coilib.bearlib.aspects.Metadata.Emptyiness, coilib.bearlib.aspects.base.aspectbase

    docs = <coilib.bearlib.aspects.docs.Documentation object>

```

```

parent
    alias of CommitMessage

subaspects = {}

class CommitMessage.Shortlog (language, **taste_values)
    Bases: coolib.bearlib.aspects.Metadata.Shortlog, coalib.bearlib.aspects.base.aspectbase

    docs = <coolib.bearlib.aspects.docs.Documentation object>

    parent
        alias of Shortlog

    subaspects = {}

class CommitMessage.Shortlog.FirstCharacter (language, **taste_values)
    Bases: coolib.bearlib.aspects.Metadata.FirstCharacter, coalib.bearlib.aspects.base.aspectbase

    docs = <coolib.bearlib.aspects.docs.Documentation object>

    parent
        alias of Shortlog

    subaspects = {}

class CommitMessage.Shortlog.Length (language, **taste_values)
    Bases: coolib.bearlib.aspects.Metadata.Length, coalib.bearlib.aspects.base.aspectbase

    docs = <coolib.bearlib.aspects.docs.Documentation object>

    parent
        alias of Shortlog

    subaspects = {}

class CommitMessage.Shortlog.Tense (language, **taste_values)
    Bases: coolib.bearlib.aspects.Metadata.Tense, coalib.bearlib.aspects.base.aspectbase

    docs = <coolib.bearlib.aspects.docs.Documentation object>

    parent
        alias of Shortlog

    subaspects = {}

class CommitMessage.Shortlog.TrailingPeriod (language, **taste_values)
    Bases: coolib.bearlib.aspects.Metadata.TrailingPeriod, coalib.bearlib.aspects.base.aspectbase

    docs = <coolib.bearlib.aspects.docs.Documentation object>

    parent
        alias of Shortlog

    subaspects = {}

CommitMessage.Shortlog.docs = <coolib.bearlib.aspects.docs.Documentation object>

```

```

CommitMessage.Shortlog.parent
    alias of CommitMessage

CommitMessage.Shortlog.subaspects = {'ColonExistence': <aspectclass 'Root.Metadata.CommitMessage.S

CommitMessage.docs = <coilib.bearlib.aspects.docs.Documentation object>

CommitMessage.parent
    alias of Metadata

CommitMessage.subaspects = {'Body': <aspectclass 'Root.Metadata.CommitMessage.Body'>, 'Shortlog': <aspectcl

class coilib.bearlib.aspects.Metadata.Emptyiness (language, **taste_values)
    Bases: coilib.bearlib.aspects.Metadata.Emptyiness, coilib.bearlib.aspects.base.aspectbase

    docs = <coilib.bearlib.aspects.docs.Documentation object>

    parent
        alias of CommitMessage

    subaspects = {}

class coilib.bearlib.aspects.Metadata.Existence (language, **taste_values)
    Bases: coilib.bearlib.aspects.Metadata.Existence, coilib.bearlib.aspects.base.aspectbase

    docs = <coilib.bearlib.aspects.docs.Documentation object>

    parent
        alias of Body

    subaspects = {}

class coilib.bearlib.aspects.Metadata.FirstCharacter (language, **taste_values)
    Bases: coilib.bearlib.aspects.Metadata.FirstCharacter,
coilib.bearlib.aspects.base.aspectbase

    docs = <coilib.bearlib.aspects.docs.Documentation object>

    parent
        alias of Shortlog

    subaspects = {}

class coilib.bearlib.aspects.Metadata.Length (language, **taste_values)
    Bases: coilib.bearlib.aspects.Metadata.Length, coilib.bearlib.aspects.base.aspectbase

    docs = <coilib.bearlib.aspects.docs.Documentation object>

    parent
        alias of Body

    subaspects = {}

class coilib.bearlib.aspects.Metadata.Metadata (language, **taste_values)
    Bases: coilib.bearlib.aspects.Metadata.Metadata, coilib.bearlib.aspects.base.aspectbase

class CommitMessage (language, **taste_values)
    Bases: coilib.bearlib.aspects.Metadata.CommitMessage,
coilib.bearlib.aspects.base.aspectbase

```



```

class Body (language, **taste_values)
    Bases: coolib.bearlib.aspects.Metadata.Body, coalib.bearlib.aspects.base.aspectbase

class Existence (language, **taste_values)
    Bases: coolib.bearlib.aspects.Metadata.Existence, coalib.bearlib.aspects.base.aspectbase
    docs = <coolib.bearlib.aspects.docs.Documentation object>
    parent
        alias of Body
    subspects = {}

class Metadata.CommitMessage.Body.Length (language, **taste_values)
    Bases: coolib.bearlib.aspects.Metadata.Length, coalib.bearlib.aspects.base.aspectbase
    docs = <coolib.bearlib.aspects.docs.Documentation object>
    parent
        alias of Body
    subspects = {}

Metadata.CommitMessage.Body.docs = <coolib.bearlib.aspects.docs.Documentation object>
Metadata.CommitMessage.Body.parent
    alias of CommitMessage
Metadata.CommitMessage.Body.subspects = {'Length': <aspectclass 'Root.Metadata.CommitMessage.Length'>}

class Metadata.CommitMessage.Emptyiness (language, **taste_values)
    Bases: coolib.bearlib.aspects.Metadata.Emptyiness, coalib.bearlib.aspects.base.aspectbase
    docs = <coolib.bearlib.aspects.docs.Documentation object>
    parent
        alias of CommitMessage
    subspects = {}

class Metadata.CommitMessage.Shortlog (language, **taste_values)
    Bases: coolib.bearlib.aspects.Metadata.Shortlog, coalib.bearlib.aspects.base.aspectbase

class ColonExistence (language, **taste_values)
    Bases: coolib.bearlib.aspects.Metadata.ColonExistence, coalib.bearlib.aspects.base.aspectbase
    docs = <coolib.bearlib.aspects.docs.Documentation object>
    parent
        alias of Shortlog
    subspects = {}

class Metadata.CommitMessage.Shortlog.FirstCharacter (language, **taste_values)
    Bases: coolib.bearlib.aspects.Metadata.FirstCharacter, coalib.bearlib.aspects.base.aspectbase
    docs = <coolib.bearlib.aspects.docs.Documentation object>

```

```

    parent
        alias of Shortlog

    subaspects = {}

class Metadata.CommitMessage.Shortlog.Length (language, **taste_values)
    Bases:
        coalib.bearlib.aspects.Metadata.Length,
        coalib.bearlib.aspects.base.aspectbase

    docs = <coilib.bearlib.aspects.docs.Documentation object>

    parent
        alias of Shortlog

    subaspects = {}

class Metadata.CommitMessage.Shortlog.Tense (language, **taste_values)
    Bases:
        coalib.bearlib.aspects.Metadata.Tense,
        coalib.bearlib.aspects.base.aspectbase

    docs = <coilib.bearlib.aspects.docs.Documentation object>

    parent
        alias of Shortlog

    subaspects = {}

class Metadata.CommitMessage.Shortlog.TrailingPeriod (language,
                                                         **taste_values)
    Bases:
        coalib.bearlib.aspects.Metadata.TrailingPeriod,
        coalib.bearlib.aspects.base.aspectbase

    docs = <coilib.bearlib.aspects.docs.Documentation object>

    parent
        alias of Shortlog

    subaspects = {}

Metadata.CommitMessage.Shortlog.docs = <coilib.bearlib.aspects.docs.Documentation object>

Metadata.CommitMessage.Shortlog.parent
    alias of CommitMessage

Metadata.CommitMessage.Shortlog.subaspects = {'ColonExistence': <aspectclass 'Root.Metadata.C

Metadata.CommitMessage.docs = <coilib.bearlib.aspects.docs.Documentation object>

Metadata.CommitMessage.parent
    alias of Metadata

Metadata.CommitMessage.subaspects = {'Body': <aspectclass 'Root.Metadata.CommitMessage.Body'>, 'SH

Metadata.docs = <coilib.bearlib.aspects.docs.Documentation object>

Metadata.parent
    alias of Root

Metadata.subaspects = {'CommitMessage': <aspectclass 'Root.Metadata.CommitMessage'>}

class coilib.bearlib.aspects.Metadata.Shortlog (language, **taste_values)
    Bases: coalib.bearlib.aspects.Metadata.Shortlog, coalib.bearlib.aspects.base.aspectbase

```

```

class ColonExistence (language, **taste_values)
    Bases: coalib.bearlib.aspects.Metadata.ColonExistence,
           coalib.bearlib.aspects.base.aspectbase
    docs = <coala.bearlib.aspects.docs.Documentation object>
    parent
        alias of Shortlog
    subspects = {}

class Shortlog.FirstCharacter (language, **taste_values)
    Bases: coalib.bearlib.aspects.Metadata.FirstCharacter,
           coalib.bearlib.aspects.base.aspectbase
    docs = <coala.bearlib.aspects.docs.Documentation object>
    parent
        alias of Shortlog
    subspects = {}

class Shortlog.Length (language, **taste_values)
    Bases: coalib.bearlib.aspects.Metadata.Length, coalib.bearlib.aspects.base.aspectbase
    docs = <coala.bearlib.aspects.docs.Documentation object>
    parent
        alias of Shortlog
    subspects = {}

class Shortlog.Tense (language, **taste_values)
    Bases: coalib.bearlib.aspects.Metadata.Tense, coalib.bearlib.aspects.base.aspectbase
    docs = <coala.bearlib.aspects.docs.Documentation object>
    parent
        alias of Shortlog
    subspects = {}

class Shortlog.TrailingPeriod (language, **taste_values)
    Bases: coalib.bearlib.aspects.Metadata.TrailingPeriod,
           coalib.bearlib.aspects.base.aspectbase
    docs = <coala.bearlib.aspects.docs.Documentation object>
    parent
        alias of Shortlog
    subspects = {}

Shortlog.docs = <coala.bearlib.aspects.docs.Documentation object>

Shortlog.parent
    alias of CommitMessage

Shortlog.subaspects = {'ColonExistence': <aspectclass 'Root.Metadata.CommitMessage.Shortlog.ColonExistence'>}

class coalib.bearlib.aspects.Metadata.Tense (language, **taste_values)
    Bases: coalib.bearlib.aspects.Metadata.Tense, coalib.bearlib.aspects.base.aspectbase

```

```

docs = <coala.bearlib.aspects.docs.Documentation object>

parent
    alias of Shortlog

subaspects = {}

class coala.bearlib.aspects.Metadata.TrailingPeriod (language, **taste_values)
    Bases: coala.bearlib.aspects.Metadata.TrailingPeriod,
           coala.bearlib.aspects.base.aspectbase

    docs = <coala.bearlib.aspects.docs.Documentation object>

    parent
        alias of Shortlog

    subaspects = {}

```

### coala.bearlib.aspects.Redundancy module

```

class coala.bearlib.aspects.Redundancy.Clone (language, **taste_values)
    Bases: coala.bearlib.aspects.Redundancy.Clone, coala.bearlib.aspects.base.aspectbase

    docs = <coala.bearlib.aspects.docs.Documentation object>

    parent
        alias of Redundancy

    subaspects = {}

class coala.bearlib.aspects.Redundancy.Redundancy (language, **taste_values)
    Bases: coala.bearlib.aspects.Redundancy.Redundancy, coala.bearlib.aspects.base.aspectbase

class Clone (language, **taste_values)
    Bases: coala.bearlib.aspects.Redundancy.Clone, coala.bearlib.aspects.base.aspectbase

    docs = <coala.bearlib.aspects.docs.Documentation object>

    parent
        alias of Redundancy

    subaspects = {}

class Redundancy.UnreachableCode (language, **taste_values)
    Bases: coala.bearlib.aspects.Redundancy.UnreachableCode,
           coala.bearlib.aspects.base.aspectbase

    class UnreachableStatement (language, **taste_values)
        Bases: coala.bearlib.aspects.Redundancy.UnreachableStatement,
               coala.bearlib.aspects.base.aspectbase

        docs = <coala.bearlib.aspects.docs.Documentation object>

        parent
            alias of UnreachableCode

        subaspects = {}

```

```

class Redundancy.UnreachableCode.UnusedFunction (language, **taste_values)
    Bases:
        coalib.bearlib.aspects.Redundancy.UnusedFunction,
        coalib.bearlib.aspects.base.aspectbase

    docs = <coalib.bearlib.aspects.docs.Documentation object>

    parent
        alias of UnreachableCode

    subaspects = {}

Redundancy.UnreachableCode.docs = <coalib.bearlib.aspects.docs.Documentation object>

Redundancy.UnreachableCode.parent
    alias of Redundancy

Redundancy.UnreachableCode.subaspects = {'UnusedFunction': <aspectclass 'Root.Redundancy.Unreacha

class Redundancy.UnusedImport (language, **taste_values)
    Bases:
        coalib.bearlib.aspects.Redundancy.UnusedImport,
        coalib.bearlib.aspects.base.aspectbase

    docs = <coalib.bearlib.aspects.docs.Documentation object>

    parent
        alias of Redundancy

    subaspects = {}

class Redundancy.UnusedVariable (language, **taste_values)
    Bases:
        coalib.bearlib.aspects.Redundancy.UnusedVariable,
        coalib.bearlib.aspects.base.aspectbase

    class UnusedGlobalVariable (language, **taste_values)
        Bases:
            coalib.bearlib.aspects.Redundancy.UnusedGlobalVariable,
            coalib.bearlib.aspects.base.aspectbase

        docs = <coalib.bearlib.aspects.docs.Documentation object>

        parent
            alias of UnusedVariable

        subaspects = {}

    class Redundancy.UnusedVariable.UnusedLocalVariable (language,
                                                            **taste_values)
        Bases:
            coalib.bearlib.aspects.Redundancy.UnusedLocalVariable,
            coalib.bearlib.aspects.base.aspectbase

        docs = <coalib.bearlib.aspects.docs.Documentation object>

        parent
            alias of UnusedVariable

        subaspects = {}

    class Redundancy.UnusedVariable.UnusedParameter (language, **taste_values)
        Bases:
            coalib.bearlib.aspects.Redundancy.UnusedParameter,
            coalib.bearlib.aspects.base.aspectbase

        docs = <coalib.bearlib.aspects.docs.Documentation object>

        parent
            alias of UnusedVariable

        subaspects = {}

```

```

Redundancy.UnusedVariable.docs = <coolib.bearlib.aspects.docs.Documentation object>

Redundancy.UnusedVariable.parent
    alias of Redundancy

Redundancy.UnusedVariable.subaspects = {'UnusedGlobalVariable': <aspectclass 'Root.Redundancy.UnusedGlobalVariable'>, 'Clone': <aspectclass 'Root.Redundancy.UnusedClone'>}

Redundancy.docs = <coolib.bearlib.aspects.docs.Documentation object>

Redundancy.parent
    alias of Root

Redundancy.subaspects = {'UnusedImport': <aspectclass 'Root.Redundancy.UnusedImport'>, 'Clone': <aspectclass 'Root.Redundancy.UnusedClone'>}

class coalib.bearlib.aspects.Redundancy.UnreachableCode (language, **taste_values)
    Bases:
        coalib.bearlib.aspects.Redundancy.UnreachableCode,
        coalib.bearlib.aspects.base.aspectbase

class UnreachableStatement (language, **taste_values)
    Bases:
        coalib.bearlib.aspects.Redundancy.UnreachableStatement,
        coalib.bearlib.aspects.base.aspectbase

    docs = <coolib.bearlib.aspects.docs.Documentation object>

    parent
        alias of UnreachableCode

    subaspects = {}

class UnreachableCode.UnusedFunction (language, **taste_values)
    Bases:
        coalib.bearlib.aspects.Redundancy.UnusedFunction,
        coalib.bearlib.aspects.base.aspectbase

    docs = <coolib.bearlib.aspects.docs.Documentation object>

    parent
        alias of UnreachableCode

    subaspects = {}

UnreachableCode.docs = <coolib.bearlib.aspects.docs.Documentation object>

UnreachableCode.parent
    alias of Redundancy

UnreachableCode.subaspects = {'UnusedFunction': <aspectclass 'Root.Redundancy.UnreachableCode.UnusedFunction'>, 'Clone': <aspectclass 'Root.Redundancy.UnreachableCode.UnusedClone'>}

class coalib.bearlib.aspects.Redundancy.UnreachableStatement (language,
                                                                **taste_values)
    Bases:
        coalib.bearlib.aspects.Redundancy.UnreachableStatement,
        coalib.bearlib.aspects.base.aspectbase

    docs = <coolib.bearlib.aspects.docs.Documentation object>

    parent
        alias of UnreachableCode

    subaspects = {}

class coalib.bearlib.aspects.Redundancy.UnusedFunction (language, **taste_values)
    Bases:
        coalib.bearlib.aspects.Redundancy.UnusedFunction,
        coalib.bearlib.aspects.base.aspectbase

    docs = <coolib.bearlib.aspects.docs.Documentation object>

```

```

    parent
        alias of UnreachableCode

    subaspects = {}

class coalib.bearlib.aspects.Redundancy.UnusedGlobalVariable (language,
                                                                **taste_values)
    Bases:
        coalib.bearlib.aspects.Redundancy.UnusedGlobalVariable,
        coalib.bearlib.aspects.base.aspectbase
    docs = <coala.bearlib.aspects.docs.Documentation object>
    parent
        alias of UnusedVariable
    subaspects = {}

class coalib.bearlib.aspects.Redundancy.UnusedImport (language, **taste_values)
    Bases:
        coalib.bearlib.aspects.Redundancy.UnusedImport,
        coalib.bearlib.aspects.base.aspectbase
    docs = <coala.bearlib.aspects.docs.Documentation object>
    parent
        alias of Redundancy
    subaspects = {}

class coalib.bearlib.aspects.Redundancy.UnusedLocalVariable (language,
                                                                **taste_values)
    Bases:
        coalib.bearlib.aspects.Redundancy.UnusedLocalVariable,
        coalib.bearlib.aspects.base.aspectbase
    docs = <coala.bearlib.aspects.docs.Documentation object>
    parent
        alias of UnusedVariable
    subaspects = {}

class coalib.bearlib.aspects.Redundancy.UnusedParameter (language, **taste_values)
    Bases:
        coalib.bearlib.aspects.Redundancy.UnusedParameter,
        coalib.bearlib.aspects.base.aspectbase
    docs = <coala.bearlib.aspects.docs.Documentation object>
    parent
        alias of UnusedVariable
    subaspects = {}

class coalib.bearlib.aspects.Redundancy.UnusedVariable (language, **taste_values)
    Bases:
        coalib.bearlib.aspects.Redundancy.UnusedVariable,
        coalib.bearlib.aspects.base.aspectbase
    class UnusedGlobalVariable (language, **taste_values)
        Bases:
            coalib.bearlib.aspects.Redundancy.UnusedGlobalVariable,
            coalib.bearlib.aspects.base.aspectbase
        docs = <coala.bearlib.aspects.docs.Documentation object>
        parent
            alias of UnusedVariable
        subaspects = {}

```

```

class UnusedVariable.UnusedLocalVariable (language, **taste_values)
    Bases:
        coalib.bearlib.aspects.Redundancy.UnusedLocalVariable,
        coalib.bearlib.aspects.base.aspectbase
    docs = <coalib.bearlib.aspects.docs.Documentation object>
    parent
        alias of UnusedVariable
    subaspects = {}

class UnusedVariable.UnusedParameter (language, **taste_values)
    Bases:
        coalib.bearlib.aspects.Redundancy.UnusedParameter,
        coalib.bearlib.aspects.base.aspectbase
    docs = <coalib.bearlib.aspects.docs.Documentation object>
    parent
        alias of UnusedVariable
    subaspects = {}

UnusedVariable.docs = <coalib.bearlib.aspects.docs.Documentation object>

UnusedVariable.parent
    alias of Redundancy

UnusedVariable.subaspects = {'UnusedGlobalVariable': <aspectclass 'Root.Redundancy.UnusedVariable.UnusedGlobalVariable'>}
```

## coala.bearlib.aspects.base module

```

class coalib.bearlib.aspects.base.aspectbase (language, **taste_values)
    Bases: object

    Base class for aspectclasses with common features for their instances.

    Derived classes must use coalib.bearlib.aspectclasses.meta.aspectclass as metaclass. This
    is automatically handled by coalib.bearlib.aspectclasses.meta.aspectclass.subaspect()
    decorator.

    tastes
        Get a dictionary of all taste names mapped to their specific values, including parent tastes.
```

## coala.bearlib.aspects.docs module

```

class coalib.bearlib.aspects.docs.Documentation (definition: str = '', example: str =
                                                    '', example_language: str = '', importance_reason: str = '', fix_suggestions: str
                                                    = '')

    Bases: object

    This class contains documentation about an aspectclass. The documentation is consistent if all members are
    given:
```

```

>>> Documentation('defined').check_consistency()
False
>>> Documentation('definition', 'example',
...               'example_language', 'importance',
...               'fix').check_consistency()
True
```



`check_consistency()`

## coala.bearlib.aspects.meta module

**class** `coala.bearlib.aspects.meta.aspectclass` (*clsname*, *bases*, *clsattrs*)

Bases: `type`

Metaclass for aspectclasses.

Root aspectclass is `coala.bearlib.aspectclasses.Root`.

**subaspect** (*subcls*)

The sub-aspectclass decorator.

See `coala.bearlib.aspectclasses.Root` for description and usage.

**tastes**

Get a dictionary of all taste names mapped to their `coala.bearlib.aspectclasses.Taste` instances.

## coala.bearlib.aspects.taste module

`coala.bearlib.aspects.taste.Taste`

Defines tastes in aspectclass definitions.

Tastes can be made only available for certain languages by providing a tuple of language identifiers on instantiation:

```
>>> Taste[bool] (
...     'Ignore ``using`` directives in C#.',
...     (True, False), default=False,
...     languages=('CSharp', )
... ).languages
(C#,)
```

If no *languages* are given, they will be available for any language. See `coala.bearlib.aspectclasses.Root` for further usage.

**exception** `coala.bearlib.aspects.taste.TasteError`

Bases: `AttributeError`

A taste is not allowed to be accessed.

**class** `coala.bearlib.aspects.taste.TasteMeta`

Bases: `type`

Metaclass for `coala.bearlib.aspectclasses.Taste`

Allows defining taste cast type via `__getitem__()`, like:

```
>>> Taste[int]().cast_type
<class 'int'>
```

## Module contents

**class** `coala.bearlib.aspects.Root` (*language*, *\*\*taste\_values*)

Bases: `coala.bearlib.aspects.base.aspectbase`

The root aspectclass.

Define sub-aspectclasses with class-bound `.subaspect` decorator. Definition string is taken from doc-string of decorated class. Remaining docs are taken from a nested `docs` class. Tastes are defined as class attributes that are instances of `coalib.bearlib.aspectclasses.Taste`.

```
>>> @Root.subaspect
... class Formatting:
...     """
...     A parent aspect for code formatting aspects...
...     """
```

We can now create subaspects like this:

```
>>> @Formatting.subaspect
... class LineLength:
...     """
...     This aspect controls the length of a line...
...     """
...     class docs:
...         example = "... "
...         example_language = "... "
...         importance_reason = "... "
...         fix_suggestions = "... "
...
...     max_line_length = Taste[int](
...         "Maximum length allowed for a line.",
...         (80, 90, 120), default=80)
```

The representation will show the full “path” to the leaf of the tree:

```
>>> Root.Formatting.LineLength
<aspectclass 'Root.Formatting.LineLength'>
```

We can see, which settings are availables:

```
>>> Formatting.tastes
{}
>>> LineLength.tastes
{'max_line_length': <....Taste[int] object at ...>}
```

And instantiate the aspect with the values, they will be automatically converted:

```
>>> Formatting('Python')
<coalib.bearlib.aspects.Root.Formatting object at 0x...>
>>> LineLength('Python', max_line_length="100").tastes
{'max_line_length': 100}
```

If no settings are given, the defaults will be taken>>> LineLength('Python').tastes {'max\_line\_length': 80}

Tastes can also be made available for only specific languages:

```
>>> from coalib.bearlib.languages import Language
>>> @Language
... class GreaterTrumpScript:
...     pass
```

```
>>> @Formatting.subaspect
... class Greatness:
...     """
...     This aspect controls the greatness of a file...
...     """
...
...     min_greatness = Taste[int](
...         "Minimum greatness factor needed for a TrumpScript file. "
...         "This is fact.",
...         (1000000, 10000000000, 10000000000000), default=1000000,
...         languages=('GreaterTrumpScript' ,))
```

```
>>> Greatness.tastes
{'min_greatness': <...Taste[int] object at ...>}
>>> Greatness('GreaterTrumpScript').tastes
{'min_greatness': 1000000}
>>> Greatness('GreaterTrumpScript', min_greatness=1000000000000).tastes
{'min_greatness': 1000000000000}
```

```
>>> Greatness('Python').tastes
{}
```

```
>>> Greatness('Python', min_greatness=10000000000)
...
Traceback (most recent call last):
...
coala.bearlib.aspects.taste.TasteError:
Root.Formatting.Greatness.min_greatness is not available ...
```

```
>>> Greatness('Python').min_greatness
...
Traceback (most recent call last):
...
coala.bearlib.aspects.taste.TasteError:
Root.Formatting.Greatness.min_greatness is not available ...
```

**class Metadata** (*language*, *\*\*taste\_values*)

Bases: *coala.bearlib.aspects.Metadata.Metadata*, *coala.bearlib.aspects.base.aspectbase*

**class CommitMessage** (*language*, *\*\*taste\_values*)

Bases: *coala.bearlib.aspects.Metadata.CommitMessage*, *coala.bearlib.aspects.base.aspectbase*

**class Body** (*language*, *\*\*taste\_values*)

Bases: *coala.bearlib.aspects.Metadata.Body*, *coala.bearlib.aspects.base.aspectbase*

**class Existence** (*language*, *\*\*taste\_values*)

Bases: *coala.bearlib.aspects.Metadata.Existence*, *coala.bearlib.aspects.base.aspectbase*

**docs** = <coala.bearlib.aspects.docs.Documentation object>

**parent**

alias of Body

**subaspects** = {}

```

class Root.Metadata.CommitMessage.Body.Length (language, **taste_values)
    Bases:
        coalib.bearlib.aspects.Metadata.Length,
        coalib.bearlib.aspects.base.aspectbase
    docs = <coalib.bearlib.aspects.docs.Documentation object>
    parent
        alias of Body
    subspects = {}
Root.Metadata.CommitMessage.Body.docs = <coalib.bearlib.aspects.docs.Documentation object>
Root.Metadata.CommitMessage.Body.parent
    alias of CommitMessage
Root.Metadata.CommitMessage.Body.subspects = {'Length': <aspectclass 'Root.Metadata.Com

class Root.Metadata.CommitMessage.Emptyiness (language, **taste_values)
    Bases:
        coalib.bearlib.aspects.Metadata.Emptyiness,
        coalib.bearlib.aspects.base.aspectbase
    docs = <coalib.bearlib.aspects.docs.Documentation object>
    parent
        alias of CommitMessage
    subspects = {}

class Root.Metadata.CommitMessage.Shortlog (language, **taste_values)
    Bases:
        coalib.bearlib.aspects.Metadata.Shortlog,
        coalib.bearlib.aspects.base.aspectbase
    class ColonExistence (language, **taste_values)
        Bases:
            coalib.bearlib.aspects.Metadata.ColonExistence,
            coalib.bearlib.aspects.base.aspectbase
        docs = <coalib.bearlib.aspects.docs.Documentation object>
        parent
            alias of Shortlog
        subspects = {}
    class Root.Metadata.CommitMessage.Shortlog.FirstCharacter (language,
                                                                **taste_values)
        Bases:
            coalib.bearlib.aspects.Metadata.FirstCharacter,
            coalib.bearlib.aspects.base.aspectbase
        docs = <coalib.bearlib.aspects.docs.Documentation object>
        parent
            alias of Shortlog
        subspects = {}
    class Root.Metadata.CommitMessage.Shortlog.Length (language,
                                                         **taste_values)
        Bases:
            coalib.bearlib.aspects.Metadata.Length,
            coalib.bearlib.aspects.base.aspectbase
        docs = <coalib.bearlib.aspects.docs.Documentation object>
        parent
            alias of Shortlog

```

```

    subaspects = {}

class Root.Metadata.CommitMessage.Shortlog.Tense (language,
                                                **taste_values)
    Bases:
        coalib.bearlib.aspects.Metadata.Tense,
        coalib.bearlib.aspects.base.aspectbase
    docs = <coalib.bearlib.aspects.docs.Documentation object>
    parent
        alias of Shortlog
    subaspects = {}

class Root.Metadata.CommitMessage.Shortlog.TrailingPeriod (language,
                                                            **taste_values)
    Bases:
        coalib.bearlib.aspects.Metadata.TrailingPeriod,
        coalib.bearlib.aspects.base.aspectbase
    docs = <coalib.bearlib.aspects.docs.Documentation object>
    parent
        alias of Shortlog
    subaspects = {}

Root.Metadata.CommitMessage.Shortlog.docs = <coalib.bearlib.aspects.docs.Documentation obj

Root.Metadata.CommitMessage.Shortlog.parent
    alias of CommitMessage

Root.Metadata.CommitMessage.Shortlog.subaspects = {'ColonExistence': <aspectclass 'Root.M

Root.Metadata.CommitMessage.docs = <coalib.bearlib.aspects.docs.Documentation object>

Root.Metadata.CommitMessage.parent
    alias of Metadata

Root.Metadata.CommitMessage.subaspects = {'Body': <aspectclass 'Root.Metadata.CommitMessage.

Root.Metadata.docs = <coalib.bearlib.aspects.docs.Documentation object>

Root.Metadata.parent
    alias of Root

Root.Metadata.subaspects = {'CommitMessage': <aspectclass 'Root.Metadata.CommitMessage'>}

class Root.Redundancy (language, **taste_values)
    Bases:
        coalib.bearlib.aspects.Redundancy.Redundancy,
        coalib.bearlib.aspects.base.aspectbase
    class Clone (language, **taste_values)
        Bases:
            coalib.bearlib.aspects.Redundancy.Clone,
            coalib.bearlib.aspects.base.aspectbase
        docs = <coalib.bearlib.aspects.docs.Documentation object>
        parent
            alias of Redundancy
        subaspects = {}
    class Root.Redundancy.UnreachableCode (language, **taste_values)
        Bases:
            coalib.bearlib.aspects.Redundancy.UnreachableCode,
            coalib.bearlib.aspects.base.aspectbase

```

```

class UnreachableStatement (language, **taste_values)
    Bases:          coalib.bearlib.aspects.Redundancy.UnreachableStatement,
                  coalib.bearlib.aspects.base.aspectbase
    docs = <coalib.bearlib.aspects.docs.Documentation object>
    parent
        alias of UnreachableCode
    subspects = {}

class Root.Redundancy.UnreachableCode.UnusedFunction (language,
**taste_values)
    Bases:          coalib.bearlib.aspects.Redundancy.UnusedFunction,
                  coalib.bearlib.aspects.base.aspectbase
    docs = <coalib.bearlib.aspects.docs.Documentation object>
    parent
        alias of UnreachableCode
    subspects = {}

Root.Redundancy.UnreachableCode.docs = <coalib.bearlib.aspects.docs.Documentation object>
Root.Redundancy.UnreachableCode.parent
    alias of Redundancy

Root.Redundancy.UnreachableCode.subspects = {'UnusedFunction': <aspectclass 'Root.Redundan

class Root.Redundancy.UnusedImport (language, **taste_values)
    Bases:          coalib.bearlib.aspects.Redundancy.UnusedImport,
                  coalib.bearlib.aspects.base.aspectbase
    docs = <coalib.bearlib.aspects.docs.Documentation object>
    parent
        alias of Redundancy
    subspects = {}

class Root.Redundancy.UnusedVariable (language, **taste_values)
    Bases:          coalib.bearlib.aspects.Redundancy.UnusedVariable,
                  coalib.bearlib.aspects.base.aspectbase
    class UnusedGlobalVariable (language, **taste_values)
        Bases:          coalib.bearlib.aspects.Redundancy.UnusedGlobalVariable,
                      coalib.bearlib.aspects.base.aspectbase
        docs = <coalib.bearlib.aspects.docs.Documentation object>
        parent
            alias of UnusedVariable
        subspects = {}
    class Root.Redundancy.UnusedVariable.UnusedLocalVariable (language,
**taste_values)
        Bases:          coalib.bearlib.aspects.Redundancy.UnusedLocalVariable,
                      coalib.bearlib.aspects.base.aspectbase
        docs = <coalib.bearlib.aspects.docs.Documentation object>
        parent
            alias of UnusedVariable

```

```

    subaspects = {}

class Root.Redundancy.UnusedVariable.UnusedParameter (language,
                                                         **taste_values)
    Bases: coalib.bearlib.aspects.Redundancy.UnusedParameter,
           coalib.bearlib.aspects.base.aspectbase
    docs = <coala.bearlib.aspects.docs.Documentation object>
    parent
        alias of UnusedVariable
    subaspects = {}

Root.Redundancy.UnusedVariable.docs = <coala.bearlib.aspects.docs.Documentation object>
Root.Redundancy.UnusedVariable.parent
    alias of Redundancy

Root.Redundancy.UnusedVariable.subaspects = {'UnusedGlobalVariable': <aspectclass 'Root.Redundancy.UnusedGlobalVariable'>, 'UnusedImport': <aspectclass 'Root.Redundancy.UnusedImport'>, 'Clone': <aspectclass 'Root.Redundancy.UnusedClone'>}

Root.Redundancy.docs = <coala.bearlib.aspects.docs.Documentation object>
Root.Redundancy.parent
    alias of Root

Root.Redundancy.subaspects = {'UnusedImport': <aspectclass 'Root.Redundancy.UnusedImport'>, 'Clone': <aspectclass 'Root.Redundancy.UnusedClone'>, 'UnusedGlobalVariable': <aspectclass 'Root.Redundancy.UnusedGlobalVariable'>, 'UnusedParameter': <aspectclass 'Root.Redundancy.UnusedParameter'>, 'UnusedVariable': <aspectclass 'Root.Redundancy.UnusedVariable'>, 'Redundancy': <aspectclass 'Root.Redundancy'>, 'Metadata': <aspectclass 'Root.Metadata'>}, 'UnusedGlobalVariable': <aspectclass 'Root.Redundancy.UnusedGlobalVariable'>, 'UnusedImport': <aspectclass 'Root.Redundancy.UnusedImport'>, 'Clone': <aspectclass 'Root.Redundancy.UnusedClone'>, 'UnusedParameter': <aspectclass 'Root.Redundancy.UnusedParameter'>, 'UnusedVariable': <aspectclass 'Root.Redundancy.UnusedVariable'>, 'Redundancy': <aspectclass 'Root.Redundancy'>, 'Metadata': <aspectclass 'Root.Metadata'>}

Root.parent = None

Root.subaspects = {'Redundancy': <aspectclass 'Root.Redundancy'>, 'Metadata': <aspectclass 'Root.Metadata'>}, 'UnusedGlobalVariable': <aspectclass 'Root.Redundancy.UnusedGlobalVariable'>, 'UnusedImport': <aspectclass 'Root.Redundancy.UnusedImport'>, 'Clone': <aspectclass 'Root.Redundancy.UnusedClone'>, 'UnusedParameter': <aspectclass 'Root.Redundancy.UnusedParameter'>, 'UnusedVariable': <aspectclass 'Root.Redundancy.UnusedVariable'>, 'Redundancy': <aspectclass 'Root.Redundancy'>, 'Metadata': <aspectclass 'Root.Metadata'>}

```

**coala.bearlib.aspects.Taste**

Defines tastes in aspectclass definitions.

Tastes can be made only available for certain languages by providing a tuple of language identifiers on instantiation:

```

>>> Taste[bool] (
...     'Ignore ``using`` directives in C#.',
...     (True, False), default=False,
...     languages=('CSharp', )
... ).languages
(C#,)

```

If no *languages* are given, they will be available for any language. See `coala.bearlib.aspectclasses.Root` for further usage.

**exception coalib.bearlib.aspects.TasteError**

Bases: `AttributeError`

A taste is not allowed to be accessed.

**class coalib.bearlib.aspects.aspectclass (clsname, bases, clsattrs)**

Bases: `type`

Metaclass for aspectclasses.

Root aspectclass is `coala.bearlib.aspectclasses.Root`.

**subaspect (subcls)**

The sub-aspectclass decorator.

See `coala.bearlib.aspectclasses.Root` for description and usage.

#### **tastes**

Get a dictionary of all taste names mapped to their `coala.bearlib.aspectclasses.Taste` instances.

### **coala.bearlib.languages package**

#### **Subpackages**

#### **coala.bearlib.languages.definitions package**

#### **Submodules**

#### **coala.bearlib.languages.definitions.C module**

#### **coala.bearlib.languages.definitions.CPP module**

#### **coala.bearlib.languages.definitions.CSS module**

#### **coala.bearlib.languages.definitions.CSharp module**

#### **coala.bearlib.languages.definitions.Java module**

#### **coala.bearlib.languages.definitions.JavaScript module**

#### **coala.bearlib.languages.definitions.Python module**

#### **coala.bearlib.languages.definitions.Unknown module**

#### **coala.bearlib.languages.definitions.Vala module**

#### **Module contents**

This directory holds language definitions.

Language definitions hold expressions that help defining specific syntax elements for a programming language.

Currently defined keys are:

names extensions comment\_delimiter multiline\_comment\_delimiters string\_delimiters multi-  
line\_string\_delimiters keywords special\_chars



## coala.bearlib.languages.documentation package

### Submodules

#### coala.bearlib.languages.documentation.DocstyleDefinition module

```
class coala.bearlib.languages.documentation.DocstyleDefinition.DocstyleDefinition (language:
                                                                    str,
                                                                    doc-
                                                                    style:
                                                                    str,
                                                                    mark-
                                                                    ers:
                                                                    (<class
                                                                    'col-
                                                                    lec-
                                                                    tions.abc.Itera
                                                                    <class
                                                                    'str'>),
                                                                    meta-
                                                                    data:
                                                                    coala.bearlib.
```

Bases: object

The DocstyleDefinition class holds values that identify a certain type of documentation comment (for which language, documentation style/tool used etc.).

**class Metadata** (*param\_start, param\_end, return\_sep*)

Bases: tuple

**param\_end**

Alias for field number 1

**param\_start**

Alias for field number 0

**return\_sep**

Alias for field number 2

DocstyleDefinition.**docstyle**

The documentation style/tool used to document code.

**Returns** A lower-case string defining the docstyle (i.e. “default” or “doxygen”).

**static DocstyleDefinition.get\_available\_definitions()**

Returns a sequence of pairs with (docstyle, language) which are available when using load().

**Returns** A sequence of pairs with (docstyle, language).

DocstyleDefinition.**language**

The programming language.

**Returns** A lower-case string defining the programming language (i.e. “cpp” or “python”).

**classmethod DocstyleDefinition.load** (*language: str, docstyle: str, coalang\_dir=None*)

Loads a DocstyleDefinition from the coala docstyle definition files.

This function considers all settings inside the according coalang-files as markers, except param\_start, param\_end and return\_sep which are considered as special metadata markers.

---

**Note:** When placing new coala docstyle definition files, these must consist of only lowercase letters and end with `.coalang!`

---

### Parameters

- **language** – The case insensitive programming language of the documentation comment as a string.
- **docstyle** – The case insensitive documentation style/tool used to document code, e.g. "default" or "doxygen".
- **coalang\_dir** – Path to directory with coalang docstyle definition files. This replaces the default path if given.

### Raises

- **FileNotFoundError** – Raised when the given docstyle was not found.
- **KeyError** – Raised when the given language is not defined for given docstyle.

**Returns** The `DocstyleDefinition` for given language and docstyle.

### `DocstyleDefinition.markers`

A tuple of marker sets that identify a documentation comment.

Marker sets consist of 3 entries where the first is the start-marker, the second one the each-line marker and the last one the end-marker. For example a marker tuple with a single marker set `(( "/*", "*", "*/" ), )` would match following documentation comment:

```
/**
 * This is documentation.
 */
```

It's also possible to supply an empty each-line marker `(( "/*", "", "*/" ))`:

```
/**
 This is more documentation.
 */
```

Markers are matched “greedy”, that means it will match as many each-line markers as possible. I.e. for `(( "///", "///", "///" ))`:

```
/// Brief documentation.
///
/// Detailed documentation.
```

**Returns** A tuple of marker/delimiter string tuples that identify a documentation comment.

### `DocstyleDefinition.metadata`

A namedtuple of certain attributes present in the documentation.

These attributes are used to define parts of the documentation.

## coala.bearlib.languages.documentation.DocumentationComment module

**class** coalib.bearlib.languages.documentation.DocumentationComment.**DocumentationComment** (*documentation\_definition, docstyle\_definition, indent, marker, range*)

Bases: object

The DocumentationComment holds information about a documentation comment inside source-code, like position etc.

**class** **Description** (*desc*)

Bases: tuple

**desc**

Alias for field number 0

**class** DocumentationComment.**Parameter** (*name, desc*)

Bases: tuple

**desc**

Alias for field number 1

**name**

Alias for field number 0

**class** DocumentationComment.**ReturnValue** (*desc*)

Bases: tuple

**desc**

Alias for field number 0

DocumentationComment.**assemble** ()

Assembles parsed documentation to the original documentation.

This function assembles the whole documentation comment, with the given markers and indentation.

DocumentationComment.**docstyle**

**classmethod** DocumentationComment.**from\_metadata** (*doccomment, docstyle\_definition, marker, indent, range*)

Assembles a list of parsed documentation comment metadata.

This function just assembles the documentation comment itself, without the markers and indentation.

```
>>> from coalib.bearlib.languages.documentation.DocumentationComment \
...     import DocumentationComment
>>> from coalib.bearlib.languages.documentation.DocstyleDefinition \
...     import DocstyleDefinition
>>> from coalib.results.TextRange import TextRange
>>> Description = DocumentationComment.Description
>>> Parameter = DocumentationComment.Parameter
>>> python_default = DocstyleDefinition.load("python3", "default")
>>> parsed_doc = [Description(desc='\nDescription\n'),
...               Parameter(name='age', desc=' Age\n')]
>>> str(DocumentationComment.from_metadata(
...     parsed_doc, python_default,
...     python_default.markers[0], 4,
```

```
...         TextRange.from_values(0, 0, 0, 0)))
'\nDescription\n:param age: Age\n'
```

#### Parameters

- **doccomment** – The list of parsed documentation comment metadata.
- **docstyle\_definition** – The `DocstyleDefinition` instance that defines what docstyle is being used in a documentation comment.
- **marker** – The markers to be used in the documentation comment.
- **indent** – The indentation to be used in the documentation comment.
- **range** – The range of the documentation comment.

**Returns** A `DocumentationComment` instance of the assembled documentation.

`DocumentationComment.language`

`DocumentationComment.metadata`

`DocumentationComment.parse()`

Parses documentation independent of language and docstyle.

**Returns** The list of all the parsed sections of the documentation. Every section is a namedtuple of either `Description` or `Parameter` or `ReturnValue`.

**Raises** **NotImplementedError** – When no parsing method is present for the given language and docstyle.

### coala.bearlib.languages.documentation.DocumentationExtraction module

`coala.bearlib.languages.documentation.DocumentationExtraction.extract_documentation` (*content, language, docstyle*)

Extracts all documentation texts inside the given source-code-string using the coala docstyle definition files.

The documentation texts are sorted by their order appearing in `content`.

For more information about how documentation comments are identified and extracted, see `DocstyleDefinition.doctypes` enumeration.

#### Parameters

- **content** – The source-code-string where to extract documentation from. Needs to be a list or tuple where each string item is a single line (including ending whitespaces like `\n`).
- **language** – The programming language used.
- **docstyle** – The documentation style/tool used (e.g. `doxygen`).

#### Raises

- **FileNotFoundError** – Raised when the docstyle definition file was not found.
- **KeyError** – Raised when the given language is not defined in given docstyle.
- **ValueError** – Raised when a docstyle definition setting has an invalid format.

**Returns** An iterator returning each `DocumentationComment` found in the content.

`coilib.bearlib.languages.documentation.DocumentationExtraction.extract_documentation_with_r`

Extracts all documentation texts inside the given source-code-string.

#### Parameters

- **content** – The source-code-string where to extract documentation from. Needs to be a list or tuple where each string item is a single line (including ending whitespaces like `\n`).
- **markers** – The list/tuple of marker-sets that identify a documentation-comment. Low-index markers have higher priority than high-index markers.

**Returns** An iterator returning each `DocumentationComment` found in the content.

## Module contents

Provides facilities to extract, parse and assemble documentation comments for different languages and documentation tools.

## Submodules

### `coilib.bearlib.languages.Language` module

**class** `coilib.bearlib.languages.Language.Language(*versions)`

Bases: object

This class defines programming languages and their versions.

You can define a new programming language as follows:

```
>>> @Language
... class TrumpScript:
...     __qualname__ = "America is great."
...     aliases = 'ts',
...     versions = 2.7, 3.3, 3.4, 3.5, 3.6
...     comment_delimiter = '#'
...     string_delimiter = {"'": "'"}
... 
```

From a bear, you can simply parse the user given language string to get the instance of the Language you desire:

```
>>> Language['trumpscript']
America is great. 2.7, 3.3, 3.4, 3.5, 3.6
>>> Language['ts 3.4, 3.6']
America is great. 3.4, 3.6
>>> Language['TS 3']
America is great. 3.3, 3.4, 3.5, 3.6
>>> Language['tS 1']
Traceback (most recent call last):
...
ValueError: No versions left
```

The attributes are not accessible unless you have selected one - and only one - version of your language:

```
>>> Language.TrumpScript(3.3, 3.4).comment_delimiter
Traceback (most recent call last):
... 
```

```
AttributeError: You have to specify ONE version ...
>>> Language.TrumpScript(3.3).comment_delimiter
'#'
```

If you don't know which version is the right one, just use this:

```
>>> Language.TrumpScript().get_default_version()
America is great. 3.6
```

To see which attributes are available, use the `attributes` property:

```
>>> Language.TrumpScript(3.3).attributes
['comment_delimiter', 'string_delimiter']
```

You can access a dictionary of the attribute values for every version from the class:

```
>>> Language.TrumpScript.comment_delimiter
OrderedDict([(2.7, '#'), (3.3, '#'), (3.4, '#'), (3.5, '#'), (3.6, '#')])
```

Any nonexistent item will of course not be served:

```
>>> Language.TrumpScript.unknown_delimiter
Traceback (most recent call last):
...
AttributeError
```

**You now know the most important parts for writing a bear using languages. Read ahead if you want to know more about working with multiple versions of programming languages as well as derivative languages!**

We can define derivative languages as follows:

```
>>> @Language
... class TrumpScriptDerivative(Language.TrumpScript):
...     __qualname__ = 'Shorter'
...     comment_delimiter = '//'
...     keywords = None
```

```
>>> Language.TrumpScriptDerivative()
Shorter 2.7, 3.3, 3.4, 3.5, 3.6
```

```
>>> Language.TrumpScriptDerivative().get_default_version().attributes
['comment_delimiter', 'keywords', 'string_delimiter']
>>> Language.TrumpScriptDerivative().get_default_version().keywords
>>> Language.TrumpScriptDerivative().get_default_version().comment_delimiter
'//'
>>> Language.TrumpScriptDerivative().get_default_version().string_delimiter
{'\"': '\"'}
```

We can get an instance via this syntax as well:

```
>>> Language[Language.TrumpScript]
America is great. 2.7, 3.3, 3.4, 3.5, 3.6
>>> Language[Language.TrumpScript(3.6)]
America is great. 3.6
```

As you see, you can use the `__qualname__` property. This will also affect the string representation and work as an implicit alias:

```
>>> str(Language.TrumpScript(3.4))
'America is great. 3.4'
```

We can specify the version by instantiating the `TrumpScript` class now:

```
>>> str(Language.TrumpScript(3.6))
'America is great. 3.6'
```

You can also define ranges of versions of languages:

```
>>> (Language.TrumpScript > 3.3) <= 3.5
America is great. 3.4, 3.5
```

```
>>> Language.TrumpScript == 3
America is great. 3.3, 3.4, 3.5, 3.6
```

Those can be combined by the or operator:

```
>>> (Language.TrumpScript == 3.6) | (Language.TrumpScript == 2)
America is great. 2.7, 3.6
```

The `__contains__` operator of the class is defined as well for strings and instances. This is case insensitive and aliases are allowed:

```
>>> Language.TrumpScript(3.6) in Language.TrumpScript
True
>>> 'ts 3.6, 3.5' in Language.TrumpScript
True
>>> 'TrumpScript 2.6' in Language.TrumpScript
False
>>> 'TrumpScript' in Language.TrumpScript
True
```

This also works on instances:

```
>>> 'ts 3.6, 3.5' in (Language.TrumpScript == 3)
True
>>> 'ts 3.6,3.5' in ((Language.TrumpScript == 2)
...                  | Language.TrumpScript(3.5))
False
>>> Language.TrumpScript(2.7, 3.5) in (Language.TrumpScript == 3)
False
>>> Language.TrumpScript(3.5) in (Language.TrumpScript == 3)
True
```

Any undefined language will obviously not be available:

```
>>> Language.Cobol
Traceback (most recent call last):
...
AttributeError
```

## attributes

Retrieves the names of all attributes that are available for this language.

**get\_default\_version()**

Retrieves the latest version the user would want to choose from the given versions in self.

(At a later point this might also retrieve a default version specifiable by the language definition, so keep using this!)

**class** coalib.bearlib.languages.Language.**LanguageMeta**

Bases: type

Metaclass for *coalib.bearlib.languages.Language.Language*.

Allows it being used as a decorator as well as implements the `__contains__` operation and stores all languages created with the decorator.

The operators are defined on the class as well, so you can do the following:

```
>>> @Language
... class SomeLang:
...     versions = 2.7, 3.3, 3.4, 3.5, 3.6
>>> Language.SomeLang > 3.4
SomeLang 3.5, 3.6
>>> Language.SomeLang < 3.4
SomeLang 2.7, 3.3
>>> Language.SomeLang >= 3.4
SomeLang 3.4, 3.5, 3.6
>>> Language.SomeLang <= 3.4
SomeLang 2.7, 3.3, 3.4
>>> Language.SomeLang == 3.4
SomeLang 3.4
>>> Language.SomeLang != 3.4
SomeLang 2.7, 3.3, 3.5, 3.6
>>> Language.SomeLang == 1.0
Traceback (most recent call last):
...
ValueError: No versions left
```

**class** coalib.bearlib.languages.Language.**LanguageUberMeta**

Bases: type

This class is used to hide the *all* attribute from the Language class.

**all** = [<class 'coalib.bearlib.languages.Language.Unknown'>, <class 'coalib.bearlib.languages.Language.C'>, <class 'coalib.bearlib.languages.Language.C'>]

**class** coalib.bearlib.languages.Language.**Languages**

Bases: tuple

A tuple-based container for *coalib.bearlib.languages.Language* instances. It supports language identifiers in any format accepted by `Language[...]`:

```
>>> Languages(['C#', Language.Python == 3])
(C#, Python 3.3, 3.4, 3.5, 3.6)
```

It provides `__contains__()` for checking if a given language identifier is included:

```
>>> 'Python 2.7, 3.5' in Languages([Language.Python()])
True
>>> 'Py 3.3' in Languages(['Python 2'])
False
>>> 'csharp' in Languages(['C#', Language.Python == 3])
True
```



`coala.bearlib.languages.Language.limit_versions` (*language, limit, operator*)

Limits given languages with the given operator:

**Parameters**

- **language** – A *Language* instance.
- **limit** – A number to limit the versions.
- **operator** – The operator to use for the limiting.

**Returns** A new *Language* instance with limited versions.

**Raises** **ValueError** – If no version is left anymore.

`coala.bearlib.languages.Language.parse_lang_str` (*string*)

Parses any given language string into name and a list of float versions:

```
>>> parse_lang_str("Python")
('Python', [])
>>> parse_lang_str("Python 3.3")
('Python', [3.3])
>>> parse_lang_str("Python 3.6, 3.3")
('Python', [3.6, 3.3])
>>> parse_lang_str("Objective C 3.6, 3.3")
('Objective C', [3.6, 3.3])
>>> parse_lang_str("Cobol, stupid!") # +ELLIPSIS
Traceback (most recent call last):
...
ValueError: Couldn't convert value 'stupid!' ...
```

## coala.bearlib.languages.LanguageDefinition module

**class** `coala.bearlib.languages.LanguageDefinition.LanguageDefinition` (*language:*  
*str,*  
*coalang\_dir=None*)

Bases: `coala.bearlib.abstractions.SectionCreatable.SectionCreatable`

**This class is deprecated!** Use the *Language* class instead.

A Language Definition holds constants which may help parsing the language. If you want to write a bear you'll probably want to use those definitions to keep your bear independent of the semantics of each language.

You can easily get your language definition by just creating it with the name of the language desired:

```
>>> list(LanguageDefinition("cpp")['extensions'])
['.c', '.cpp', '.h', '.hpp']
```

For some languages aliases exist, the name is case insensitive; they will behave just like before and return settings:

```
>>> dict(LanguageDefinition('C++')['comment_delimiter'])
{'//': ''}
>>> dict(LanguageDefinition('C++')['string_delimiters'])
{'\"': '\"'}
```

If no language exists, you will get a `FileNotFoundError`:

```
>>> LanguageDefinition("BULLSHIT!")
Traceback (most recent call last):
...
FileNotFoundError
```

Custom coalangs are no longer supported. You can simply register your languages to the Languages decorator. When giving a custom coalang directory a warning will be emitted and it will attempt to load the given Language anyway through conventional means:

```
>>> LanguageDefinition("custom", coalang_dir='somewhere')
Traceback (most recent call last):
...
FileNotFoundError
```

If you need a custom language, just go like this:

```
>>> @Language
... class MyLittlePony:
...     color = 'green'
...     legs = 5
>>> int(LanguageDefinition('mylittlepony')['legs'])
5
```

But seriously, just use *Language* - and mind that it's already typed:

```
>>> Language['mylittlepony'].get_default_version().legs
5
```

## Module contents

This directory holds means to get generic information for specific languages.

### coala.bearlib.naming\_conventions package

#### Module contents

`coala.bearlib.naming_conventions.to_camelcase(string)`  
Converts the given string to camel-case.

```
>>> to_camelcase('Hello_world')
'helloWorld'
>>> to_camelcase('__Init__file__')
'__initFile__'
>>> to_camelcase('')
''
>>> to_camelcase('alreadyCamelCase')
'alreadyCamelCase'
>>> to_camelcase('    string')
'__string'
```

**Parameters** *string* – The string to convert.

**Returns** The camel-cased string.

`coala.bearlib.naming_conventions.to_pascalcase(string)`  
 Converts the given to string pascal-case.

```
>>> to_pascalcase('hello_world')
'HelloWorld'
>>> to_pascalcase('__init__file__')
'__InitFile__'
>>> to_pascalcase('')
''
>>> to_pascalcase('AlreadyPascalCase')
'AlreadyPascalCase'
>>> to_pascalcase('    string')
'__String'
```

**Parameters** `string` – The string to convert.

**Returns** The pascal-cased string.

`coala.bearlib.naming_conventions.to_snakecase(string)`  
 Converts the given string to snake-case.

```
>>> to_snakecase('HelloWorld')
'hello_world'
>>> to_snakecase('__Init__File__')
'__init_file__'
>>> to_snakecase('')
''
>>> to_snakecase('already_snake_case')
'already_snake_case'
>>> to_snakecase('    string    ')
'__string__'
>>> to_snakecase('ABCde.F.G..H..IH')
'a_b_cde.f.g..h..i_h'
```

**Parameters** `string` – The string to convert.

**Returns** The snake-cased string.

`coala.bearlib.naming_conventions.to_spacecase(string)`  
 Converts the given string to space-case.

```
>>> to_spacecase('helloWorld')
'Hello World'
>>> to_spacecase('__Init__File__')
'Init File'
>>> to_spacecase('')
''
>>> to_spacecase('Already Space Case')
'Already Space Case'
>>> to_spacecase('    string    ')
'String'
```

**Parameters** `string` – The string to convert.

**Returns** The space-cased string.

## coala.bearlib.spacing package

### Submodules

### coala.bearlib.spacing.SpacingHelper module

```
class coala.bearlib.spacing.SpacingHelper.SpacingHelper(tab_width: int = 4)
    Bases: coala.bearlib.abstractions.SectionCreatable.SectionCreatable

    DEFAULT_TAB_WIDTH = 4

    get_indentation(line: str)
        Checks the lines indentation.

        Parameters line – A string to check for indentation.

        Returns The indentation count in spaces.

    replace_spaces_with_tabs(line: str)
        Replaces spaces with tabs where possible. However in no case only one space will be replaced by a tab.
        Example: " a_text another" will be converted to "a_text another", assuming the tab_width is set to 4.

        Parameters line – The string with spaces to replace.

        Returns The converted string.

    replace_tabs_with_spaces(line: str)
        Replaces tabs in this line with the appropriate number of spaces.
        Example: "" will be converted to "", assuming the tab_width is set to 4.

        Parameters line – The string with tabs to replace.

        Returns A string with no tabs.

    yield_tab_lengths(input: str)
        Yields position and size of tabs in a input string.

        Parameters input – The string with tabs.
```

### Module contents

### Module contents

The bearlib is an optional library designed to ease the task of any Bear. Just as the rest of coala the bearlib is designed to be as easy to use as possible while offering the best possible flexibility.

```
coala.bearlib.deprecate_bear(bear)
    Use this to deprecate a bear. Say we have a bear:
```

```
>>> class SomeBear:
...     def run(*args):
...         print("I'm running!")
```

To change the name from SomeOldBear to SomeBear you can keep the SomeOldBear.py around with those contents:

```
>>> @deprecate_bear
... class SomeOldBear(SomeBear): pass
```

Now let's run the bear:

```
>>> import sys
>>> logging.basicConfig(stream=sys.stdout, level=logging.DEBUG)
>>> SomeOldBear().run()
WARNING:root:The bear SomeOldBear is deprecated. Use SomeBear instead!
I'm running!
```

**Parameters** **bear** – An old bear class that inherits from the new one (so it gets its methods and can just contain a pass.)

**Returns** A bear class that warns about deprecation on use.

`coilib.bearlib.deprecate_settings(**depr_args)`

The purpose of this decorator is to allow passing old settings names to bears due to the heavy changes in their names.

```
>>> @deprecate_settings(new='old')
... def run(new):
...     print(new)
```

Now we can simply call the bear with the deprecated setting, we'll get a warning - but it still works!

```
>>> import sys
>>> logging.basicConfig(stream=sys.stdout, level=logging.DEBUG)
>>> run(old="Hello world!")
WARNING:root:The setting `old` is deprecated. Please use `new` instead.
Hello world!
>>> run(new="Hello world!")
Hello world!
```

This example represents the case where the old setting name needs to be modified to match the new one.

```
>>> @deprecate_settings(new=('old', lambda a: a + 'coala!'))
... def func(new):
...     print(new)
```

```
>>> func(old="Welcome to ")
WARNING:root:The setting `old` is deprecated. Please use `new` instead.
Welcome to coala!
>>> func(new='coala!')
coala!
```

This example represents the case where the old and new settings are provided to the function.

```
>>> @deprecate_settings(new='old')
... def run(new):
...     print(new)
>>>
... run(old="Hello!", new='coala is always written with lowercase `c`.')
WARNING:root:The setting `old` is deprecated. Please use `new` instead.
WARNING:root:The value of `old` and `new` are conflicting. `new` will...
coala is always written with lowercase `c`.
>>> run(old='Hello!', new='Hello!')
WARNING:root:The setting `old` is deprecated. Please use `new` instead.
Hello!
```

The metadata for coala has been adjusted as well:

```
>>> list(run.__metadata__.non_optional_params.keys())
['new']
>>> list(run.__metadata__.optional_params.keys())
['old']
```

You cannot deprecate an already deprecated setting. Don't try. It will introduce non-deterministic errors in your program.

**Parameters** `depr_args` – A dictionary of settings as keys and their deprecated names as values.

## 1.1.2 coalib.bears package

### Subpackages

#### coala.bears.requirements package

### Submodules

#### coala.bears.requirements.DistributionRequirement module

**class** `coala.bears.requirements.DistributionRequirement.DistributionRequirement` (*\*\*manager\_comm*  
*Bases: coalib.bears.requirements.PackageRequirement.PackageRequirement*

This class is a subclass of `PackageRequirement`. It specifies the proper type automatically.

#### coala.bears.requirements.GemRequirement module

**class** `coala.bears.requirements.GemRequirement.GemRequirement` (*package, version='',*  
*require=''*)  
*Bases: coalib.bears.requirements.PackageRequirement.PackageRequirement*

This class is a subclass of `PackageRequirement`. It specifies the proper type for ruby packages automatically and provide a function to check for the requirement.

**is\_installed()**  
 Checks if the dependency is installed.

**Parameters** `return` – True if dependency is installed, false otherwise.

#### coala.bears.requirements.GoRequirement module

**class** `coala.bears.requirements.GoRequirement.GoRequirement` (*package, version='',*  
*flag=''*)  
*Bases: coalib.bears.requirements.PackageRequirement.PackageRequirement*

This class is a subclass of `PackageRequirement`. It specifies the proper type for go packages automatically and provide a function to check for the requirement.

**is\_installed()**  
 Checks if the dependency is installed.

**Parameters** `return` – True if dependency is installed, false otherwise.

## coala.bears.requirements.JuliaRequirement module

```
class coalib.bears.requirements.JuliaRequirement.JuliaRequirement (package,
                                                                    version='')
    Bases: coalib.bears.requirements.PackageRequirement.PackageRequirement
```

This class is a subclass of `PackageRequirement`. It specifies the proper type for julia packages automatically and provide a function to check for the requirement.

```
is_installed()
    Checks if the dependency is installed.

    Returns True if dependency is installed, False otherwise.
```

## coala.bears.requirements.NpmRequirement module

```
class coalib.bears.requirements.NpmRequirement.NpmRequirement (package, version='')
    Bases: coalib.bears.requirements.PackageRequirement.PackageRequirement
```

This class is a subclass of `PackageRequirement`. It specifies the proper type for npm packages automatically and provide a function to check for the requirement.

```
is_installed()
    Checks if the dependency is installed.

    Parameters return – True if dependency is installed, false otherwise.
```

## coala.bears.requirements.PackageRequirement module

```
class coalib.bears.requirements.PackageRequirement.PackageRequirement (type: str,
                                                                    package:
                                                                    str, ver-
                                                                    sion='')
    Bases: object
```

This class helps keeping track of bear requirements. It should simply be appended to the `REQUIREMENTS` tuple inside the Bear class.

Two `PackageRequirements` should always be equal if they have the same type, package and version:

```
>>> pr1 = PackageRequirement('pip', 'coala_decorators', '0.1.0')
>>> pr2 = PackageRequirement('pip', 'coala_decorators', '0.1.0')
>>> pr1 == pr2
True
```

```
is_installed()
    Check if the requirement is satisfied.
```

```
>>> PackageRequirement('pip', 'coala_decorators', '0.2.1').is_installed()
↪
Traceback (most recent call last):
...
NotImplementedError
```

**Returns** Returns True if satisfied, False if not.

## coala.bears.requirements.PipRequirement module

**class** `coala.bears.requirements.PipRequirement.PipRequirement` (*package*, *version=''*)  
 Bases: `coala.bears.requirements.PackageRequirement.PackageRequirement`

This class is a subclass of `PackageRequirement`. It specifies the proper type for python packages automatically and provide a function to check for the requirement.

**is\_installed()**  
 Checks if the dependency is installed.

**Parameters** **return** – True if dependency is installed, false otherwise.

## coala.bears.requirements.RscriptRequirement module

**class** `coala.bears.requirements.RscriptRequirement.RscriptRequirement` (*package*,  
*ver-*  
*sion=''*,  
*flag=''*,  
*repo=''*)  
 Bases: `coala.bears.requirements.PackageRequirement.PackageRequirement`

This class is a subclass of `PackageRequirement`. It specifies the proper type for R packages automatically and provide a function to check for the requirement.

**is\_installed()**  
 Checks if the dependency is installed.

**Parameters** **return** – True if dependency is installed, false otherwise.

## Module contents

### Submodules

#### coala.bears.BEAR\_KIND module

#### coala.bears.Bear module

**class** `coala.bears.Bear.Bear` (*section: coala.settings.Section.Section*, *message\_queue*, *timeout=0*)  
 Bases: `pyprint.Printer.Printer`, `coala.output.printers.LogPrinter.LogPrinterMixin`

A bear contains the actual subroutine that is responsible for checking source code for certain specifications. However it can actually do whatever it wants with the files it gets. If you are missing some Result type, feel free to contact us and/or help us extending the coala.

This is the base class for every bear. If you want to write an bear, you will probably want to look at the `GlobalBear` and `LocalBear` classes that inherit from this class. In any case you'll want to overwrite at least the `run` method. You can send debug/warning/error messages through the `debug()`, `warn()`, `err()` functions. These will send the appropriate messages so that they are outputted. Be aware that if you use `err()`, you are expected to also terminate the bear run-through immediately.

If you need some setup or teardown for your bear, feel free to overwrite the `set_up()` and `tear_down()` functions. They will be invoked before/after every run invocation.

Settings are available at all times through `self.section`.



To indicate which languages your bear supports, just give it the `LANGUAGES` value which should be a set of string(s):

```
>>> class SomeBear(Bear):
...     LANGUAGES = {'C', 'CPP', 'C#', 'D'}
```

To indicate the requirements of the bear, assign `REQUIREMENTS` a set with instances of `PackageRequirements`.

```
>>> class SomeBear(Bear):
...     REQUIREMENTS = {
...         PackageRequirement('pip', 'coala_decorators', '0.2.1')}
```

If your bear uses requirements from a manager we have a subclass from, you can use the subclass, such as `PipRequirement`, without specifying manager:

```
>>> class SomeBear(Bear):
...     REQUIREMENTS = {PipRequirement('coala_decorators', '0.2.1')}
```

To specify additional attributes to your bear, use the following:

```
>>> class SomeBear(Bear):
...     AUTHORS = {'Jon Snow'}
...     AUTHORS_EMAILS = {'jon_snow@gmail.com'}
...     MAINTAINERS = {'Catelyn Stark'}
...     MAINTAINERS_EMAILS = {'catelyn_stark@gmail.com'}
...     LICENSE = 'AGPL-3.0'
...     ASCIINEMA_URL = 'https://asciinema.org/a/80761'
```

If the maintainers are the same as the authors, they can be omitted:

```
>>> class SomeBear(Bear):
...     AUTHORS = {'Jon Snow'}
...     AUTHORS_EMAILS = {'jon_snow@gmail.com'}
>>> SomeBear.maintainers
{'Jon Snow'}
>>> SomeBear.maintainers_emails
{'jon_snow@gmail.com'}
```

If your bear needs to include local files, then specify it giving strings containing relative file paths to the `INCLUDE_LOCAL_FILES` set:

```
>>> class SomeBear(Bear):
...     INCLUDE_LOCAL_FILES = {'checkstyle.jar', 'google_checks.xml'}
```

To keep track easier of what a bear can do, simply tell it to the `CAN_FIX` and the `CAN_DETECT` sets. Possible values:

```
>>> CAN_DETECT = {'Syntax', 'Formatting', 'Security', 'Complexity', 'Smell',
... 'Unused Code', 'Redundancy', 'Variable Misuse', 'Spelling',
... 'Memory Leak', 'Documentation', 'Duplication', 'Commented Code',
... 'Grammar', 'Missing Import', 'Unreachable Code', 'Undefined Element',
... 'Code Simplification'}
>>> CAN_FIX = {'Syntax', ...}
```

Specifying something to `CAN_FIX` makes it obvious that it can be detected too, so it may be omitted:

```
>>> class SomeBear(Bear):
...     CAN_DETECT = {'Syntax', 'Security'}
...     CAN_FIX = {'Redundancy'}
>>> list(sorted(SomeBear.can_detect))
['Redundancy', 'Security', 'Syntax']
```

Every bear has a data directory which is unique to that particular bear:

```
>>> class SomeBear(Bear): pass
>>> class SomeOtherBear(Bear): pass
>>> SomeBear.data_dir == SomeOtherBear.data_dir
False
```

BEAR\_DEPS contains bear classes that are to be executed before this bear gets executed. The results of these bears will then be passed to the run method as a dict via the dependency\_results argument. The dict will have the name of the Bear as key and the list of its results as results:

```
>>> class SomeBear(Bear): pass
>>> class SomeOtherBear(Bear):
...     BEAR_DEPS = {SomeBear}
>>> SomeOtherBear.BEAR_DEPS
{<class 'coilib.bears.Bear.SomeBear'>}
```

**ASCIINEMA\_URL** = ‘

**AUTHORS** = set()

**AUTHORS\_EMAILS** = set()

**BEAR\_DEPS** = set()

**CAN\_DETECT** = set()

**CAN\_FIX** = set()

**INCLUDE\_LOCAL\_FILES** = set()

**LANGUAGES** = set()

**LICENSE** = ‘

**MAINTAINERS** = set()

**MAINTAINERS\_EMAILS** = set()

**PLATFORMS** = {‘any’}

**REQUIREMENTS** = set()

**can\_detect** = set()

**classmethod check\_prerequisites()**

Checks whether needed runtime prerequisites of the bear are satisfied.

This function gets executed at construction and returns True by default.

Section value requirements shall be checked inside the run method.

**Returns** True if prerequisites are satisfied, else False or a string that serves a more detailed description of what’s missing.

**data\_dir** = ‘/home/docs/.local/share/coala-bears/Bear’

**download\_cached\_file** (*url*, *filename*)

Downloads the file if needed and caches it for the next time. If a download happens, the user will be informed.

Take a sane simple bear:

```
>>> from queue import Queue
>>> bear = Bear(Section("a section"), Queue())
```

We can now carelessly query for a neat file that doesn't exist yet:

```
>>> from os import remove
>>> if exists(join(bear.data_dir, "a_file")):
...     remove(join(bear.data_dir, "a_file"))
>>> file = bear.download_cached_file("https://github.com/", "a_file")
```

If we download it again, it'll be much faster as no download occurs:

```
>>> newfile = bear.download_cached_file("https://github.com/", "a_file")
>>> newfile == file
True
```

**Parameters**

- **url** – The URL to download the file from.
- **filename** – The filename it should get, e.g. "test.txt".

**Returns** A full path to the file ready for you to use!

**execute** (\**args*, \*\**kwargs*)**get\_config\_dir** ()

Gives the directory where the configuration file is

**Returns** Directory of the config file

**classmethod get\_metadata** ()

**Returns** Metadata for the run function. However parameters like `self` or parameters implicitly used by coala (e.g. filename for local bears) are already removed.

**classmethod get\_non\_optional\_settings** ()

This method has to determine which settings are needed by this bear. The user will be prompted for needed settings that are not available in the settings file so don't include settings where a default value would do.

**Returns** A dictionary of needed settings as keys and a tuple of help text and annotation as values

**static kind** ()

**Returns** The kind of the bear

**log\_message** (*log\_message*, *timestamp=None*, \*\**kwargs*)**maintainers = set** ()**maintainers\_emails = set** ()**classmethod missing\_dependencies** (*lst*)

Checks if the given list contains all dependencies.

**Parameters** **lst** – A list of all already resolved bear classes (not instances).

**Returns** A set of missing dependencies.

**name** = 'Bear'

**new\_result**

Returns a partial for creating a result with this bear already bound.

**run** (\*args, \*, *dependency\_results=None*, \*\*kwargs)

**run\_bear\_from\_section** (args, kwargs)

**static setup\_dependencies** ()

This is a user defined function that can download and set up dependencies (via `download_cached_file` or arbitrary other means) in an OS independent way.

## coala.bears.GlobalBear module

**class** `coala.bears.GlobalBear.GlobalBear` (*file\_dict*, *section*, *message\_queue*, *timeout=0*)

Bases: `coala.bears.Bear.Bear`

A GlobalBear is able to analyze semantic facts across several file.

The results of a GlobalBear will be presented grouped by the origin Bear. Therefore Results spanning above multiple files are allowed and will be handled right.

If you only look at one file at once anyway a LocalBear is better for your needs. (And better for performance and usability for both user and developer.)

**static kind** ()

**run** (\*args, \*, *dependency\_results=None*, \*\*kwargs)

Handles all files in *file\_dict*.

**Returns** A list of Result type.

## coala.bears.LocalBear module

**class** `coala.bears.LocalBear.LocalBear` (*section*: `coala.settings.Section.Section`, *message\_queue*, *timeout=0*)

Bases: `coala.bears.Bear.Bear`

A LocalBear is a Bear that analyzes only one file at once. It therefore can not analyze semantical facts over multiple files.

This has the advantage that it can be highly parallelized. In addition, the results from multiple bears for one file can be shown together for that file, which is better to grasp for the user. coala takes care of all that.

Examples for LocalBear's could be:

- A SpaceConsistencyBear that checks every line for trailing whitespaces, tabs, etc.
- A VariableNameBear that checks variable names and constant names for certain conditions

**classmethod** `get_metadata` ()

**static kind** ()

**run** (*filename*, *file*, \*args, \*, *dependency\_results=None*, \*\*kwargs)

Handles the given file.

**Parameters**

- **filename** – The filename of the file
- **file** – The file contents as string array

**Returns** A list of Result

## Module contents

### 1.1.3 coalib.collecting package

#### Submodules

#### coailb.collecting.Collectors module

`coailb.collecting.Collectors.collect_all_bears_from_sections` (*sections*,  
*log\_printer*)

Collect all kinds of bears from bear directories given in the sections.

##### Parameters

- **sections** – List of sections so bear\_dirs are taken into account
- **log\_printer** – Log\_printer to handle logging

**Returns** Tuple of dictionaries of local and global bears. The dictionary key is section class and dictionary value is a list of Bear classes

`coailb.collecting.Collectors.collect_bears` (*bear\_dirs*, *bear\_globs*, *kinds*, *log\_printer*,  
*warn\_if\_unused\_glob=True*)

Collect all bears from bear directories that have a matching kind matching the given globs.

##### Parameters

- **bear\_dirs** – Directory name or list of such that can contain bears.
- **bear\_globs** – Globs of bears to collect.
- **kinds** – List of bear kinds to be collected.
- **log\_printer** – log\_printer to handle logging.
- **warn\_if\_unused\_glob** – True if warning message should be shown if a glob didn't give any bears.

**Returns** Tuple of list of matching bear classes based on kind. The lists are in the same order as kinds.

`coailb.collecting.Collectors.collect_dirs` (*dir\_paths*, *ignored\_dir\_paths=None*)

Evaluate globs in directory paths and return all matching directories

##### Parameters

- **dir\_paths** – File path or list of such that can include globs
- **ignored\_dir\_paths** – List of globs that match to-be-ignored dirs

**Returns** List of paths of all matching directories

`coailb.collecting.Collectors.collect_files` (*file\_paths*, *log\_printer*, *ig-*  
*nored\_file\_paths=None*,  
*limit\_file\_paths=None*)

Evaluate globs in file paths and return all matching files

##### Parameters

- **file\_paths** – File path or list of such that can include globs
- **ignored\_file\_paths** – List of globs that match to-be-ignored files

- **limit\_file\_paths** – List of globs that the files are limited to

**Returns** List of paths of all matching files

`coolib.collecting.Collectors.collect_registered_bears_dirs(entrpoint)`

Searches setup tools for the entripoint and returns the bear directories given by the module.

**Parameters** **entrpoint** – The entripoint to find packages with.

**Returns** List of bear directories.

`coolib.collecting.Collectors.filter_capabilities_by_languages(bears, languages)`

Filters the bears capabilities by languages.

**Parameters**

- **bears** – Dictionary with sections as keys and list of bears as values.
- **languages** – Languages that bears are being filtered on.

**Returns** New dictionary with languages as keys and their bears capabilities as values. The capabilities are stored in a tuple of two elements where the first one represents what the bears can detect, and the second one what they can fix.

`coolib.collecting.Collectors.filter_section_bears_by_languages(bears, languages)`

Filters the bears by languages.

**Parameters**

- **bears** – The dictionary of the sections as keys and list of bears as values.
- **languages** – Languages that bears are being filtered on.

**Returns** New dictionary with filtered out bears that don't match any language from languages.

`coolib.collecting.Collectors.get_all_bears_names()`

`coolib.collecting.Collectors.icollect(file_paths, ignored_globs=None)`

Evaluate globs in file paths and return all matching files.

**Parameters**

- **file\_paths** – File path or list of such that can include globs
- **ignored\_globs** – List of globs to ignore when matching files

**Returns** Iterator that yields tuple of path of a matching file, the glob where it was found

`coolib.collecting.Collectors.icollect_bears(bear_dir_glob, bear_globs, kinds, log_printer)`

Collect all bears from bear directories that have a matching kind.

**Parameters**

- **bear\_dir\_glob** – Directory globs or list of such that can contain bears
- **bear\_globs** – Globs of bears to collect
- **kinds** – List of bear kinds to be collected
- **log\_printer** – Log\_printer to handle logging

**Returns** Iterator that yields a tuple with bear class and which bear\_glob was used to find that bear class.

## coala.collecting.Dependencies module

**exception** `coala.collecting.Dependencies.CircularDependencyError`

Bases: `Exception`

**classmethod** `for_bears(bears)`

Creates the `CircularDependencyError` with a helpful message about the dependency.

`coala.collecting.Dependencies.resolve(bears)`

Collects all dependencies of the given bears. This will also remove duplicates.

**Parameters** `bears` – The given bears. Will not be modified.

**Returns** The new list of bears, sorted so that it can be executed sequentially without dependency issues.

## coala.collecting.Importers module

`coala.collecting.Importers.iimport_objects(file_paths, names=None, types=None, supers=None, attributes=None, local=False, suppress_output=False)`

Import all objects from the given modules that fulfill the requirements

### Parameters

- **file\_paths** – File path(s) from which objects will be imported.
- **names** – Name(s) an objects need to have one of.
- **types** – Type(s) an objects need to be out of.
- **supers** – Class(es) objects need to be a subclass of.
- **attributes** – Attribute(s) an object needs to (all) have.
- **local** – If True: Objects need to be defined in the file they appear in to be collected.
- **suppress\_output** – Whether console output from stdout shall be suppressed or not.

**Returns** An iterator that yields all matching python objects.

**Raises** **Exception** – Any exception that is thrown in module code or an `ImportError` if paths are erroneous.

`coala.collecting.Importers.import_objects(file_paths, names=None, types=None, supers=None, attributes=None, local=False, verbose=False)`

Import all objects from the given modules that fulfill the requirements

### Parameters

- **file\_paths** – File path(s) from which objects will be imported
- **names** – Name(s) an objects need to have one of
- **types** – Type(s) an objects need to be out of
- **supers** – Class(es) objects need to be a subclass of
- **attributes** – Attribute(s) an object needs to (all) have
- **local** – if True: Objects need to be defined in the file they appear in to be collected

**Returns** list of all matching python objects

**Raises `Exception`** – Any exception that is thrown in module code or an `ImportError` if paths are erroneous.

`coala.collecting.Importers.object_defined_in(obj, file_path)`

Check if the object is defined in the given file.

```
>>> object_defined_in(object_defined_in, __file__)
True
>>> object_defined_in(object_defined_in, "somewhere else")
False
```

Builtins are always defined outside any given file:

```
>>> object_defined_in(False, __file__)
False
```

#### Parameters

- **obj** – The object to check.
- **file\_path** – The path it might be defined in.

**Returns** True if the object is defined in the file.

## Module contents

### 1.1.4 coala.misc package

#### Submodules

#### coala.misc.Annotations module

`coala.misc.Annotations.typechain(*args)`

Returns function which applies the first transformation it can from args and returns transformed value, or the value itself if it is in args.

```
>>> function = typechain(int, 'a', ord, None)
>>> function("10")
10
>>> function("b")
98
>>> function("a")
'a'
>>> function(int)
<class 'int'>
>>> function(None) is None
True
>>> function("str")
Traceback (most recent call last):
...
ValueError: Couldn't convert value 'str' to any specified type or find it in
↳specified values.
```

**Raises `TypeError`** – Raises when either no functions are specified for checking.



## coolib.misc.BuildManPage module

```
class coalib.misc.BuildManPage.BuildManPage (dist)
    Bases: distutils.cmd.Command
```

Add a `build_manpage` command to your `setup.py`. To use this Command class add a command to call this class:

```
# For setuptools
setup(
    entry_points={
        "distutils.commands": [
            "build_manpage = coalib.misc.BuildManPage:BuildManPage"
        ]
    }
)

# For distutils
from coalib.misc.BuildManPage import BuildManPage
setup(
    cmdclass={'build_manpage': BuildManPage}
)
```

You can then use the following setup command to produce a man page:

```
$ python setup.py build_manpage --output=coala.1 --parser=coolib.
  ↳ parsing.DefaultArgParser:default_arg_parser
```

If automatically want to build the man page every time you invoke your build, add to your `setup.cfg` the following:

```
[build_manpage]
output = <appname>.1
parser = <path_to_your_parser>
```

```
finalize_options()
```

```
initialize_options()
```

```
run()
```

```
user_options = [('output=', 'O', 'output file'), ('parser=', None, 'module path to an ArgumentParser instance(e.g. my
```

```
class coalib.misc.BuildManPage.ManPageFormatter (prog,
                                                    indent_increment=2,
                                                    max_help_position=24, width=None,
                                                    desc=None, long_desc=None,
                                                    ext_sections=None, parser=None)

    Bases: argparse.HelpFormatter
```

```
format_man_page()
```

## coolib.misc.Caching module

```
class coalib.misc.Caching.FileCache (log_printer: coalib.output.printers.LogPrinter.LogPrinterMixin,
                                       project_dir: str, flush_cache: bool = False)
```

```
    Bases: object
```

This object is a file cache that helps in collecting only the changed and new files since the last run. Example/Tutorial:

```
>>> from pyprint.NullPrinter import NullPrinter
>>> from coalib.output.printers.LogPrinter import LogPrinter
>>> import logging
>>> import copy, time
>>> log_printer = LogPrinter()
>>> log_printer.log_level = logging.CRITICAL
```

To initialize the cache create an instance for the project:

```
>>> cache = FileCache(log_printer, "test", flush_cache=True)
```

Now we can track new files by running:

```
>>> cache.track_files(["a.c", "b.c"])
```

Since all cache operations are lazy (for performance), we need to explicitly write the cache to disk for persistence in future uses: (Note: The cache will automatically figure out the write location)

```
>>> cache.write()
```

Let's go into the future:

```
>>> time.sleep(1)
```

Let's create a new instance to simulate a separate run:

```
>>> cache = FileCache(log_printer, "test", flush_cache=False)
```

```
>>> old_data = copy.deepcopy(cache.data)
```

We can mark a file as changed by doing:

```
>>> cache.untrack_files({"a.c"})
```

Again write to disk after calculating the new cache times for each file:

```
>>> cache.write()
>>> new_data = cache.data
```

Since we marked 'a.c' as a changed file:

```
>>> "a.c" not in cache.data
True
>>> "a.c" in old_data
True
```

Since 'b.c' was untouched after the second run, its time was updated to the latest value:

```
>>> old_data["b.c"] < new_data["b.c"]
True
```

**flush\_cache()**

Flushes the cache and deletes the relevant file.

**get\_uncached\_files(files)**

Returns the set of files that are not in the cache yet or have been untracked.

**Parameters files** – The list of collected files.

**Returns** A set of files that are uncached.

**track\_files** (*files*)

Start tracking files given in *files* by adding them to the database.

**Parameters** *files* – A set of files that need to be tracked. These files are initialized with their last modified tag as -1.

**untrack\_files** (*files*)

Removes the given files from the cache so that they are no longer considered cached for this and the next run.

**Parameters** *files* – A set of files to remove from cache.

**write** ()

Update the last run time on the project for each file to the current time. Using this object as a contextmanager is preferred (that will automatically call this method on exit).

## coolib.misc.CachingUtilities module

`coolib.misc.CachingUtilities.delete_files(log_printer, identifiers)`

Delete the given identifiers from the user's coala data directory.

**Parameters**

- **log\_printer** – A LogPrinter object to use for logging.
- **identifiers** – The list of files to be deleted.

**Returns** True if all the given files were successfully deleted. False otherwise.

`coolib.misc.CachingUtilities.get_data_path(log_printer, identifier)`

Get the full path of *identifier* present in the user's data directory.

**Parameters**

- **log\_printer** – A LogPrinter object to use for logging.
- **identifier** – The file whose path needs to be expanded.

**Returns** Full path of the file, assuming it's present in the user's config directory. Returns None if there is a `PermissionError` in creating the directory.

`coolib.misc.CachingUtilities.get_settings_hash(sections, targets=[], ignore_settings: list = ['disable_caching'])`

Compute and return a unique hash for the settings.

**Parameters**

- **sections** – A dict containing the settings for each section.
- **targets** – The list of sections that are enabled.
- **ignore\_settings** – Setting keys to remove from sections before hashing.

**Returns** A MD5 hash that is unique to the settings used.

`coolib.misc.CachingUtilities.hash_id(text)`

Hashes the given text.

**Parameters** *text* – String to to be hashed

**Returns** A MD5 hash of the given string

`coala.lib.misc.CachingUtilities.pickle_dump(log_printer, identifier, data)`

Write data into the file `filename` present in the user config directory.

#### Parameters

- **log\_printer** – A LogPrinter object to use for logging.
- **identifier** – The name of the file present in the user config directory.
- **data** – Data to be serialized and written to the file using pickle.

**Returns** True if the write was successful. False if there was a permission error in writing.

`coala.lib.misc.CachingUtilities.pickle_load(log_printer, identifier, fallback=None)`

Get the data stored in `filename` present in the user config directory. Example usage:

```
>>> from pyprint.NullPrinter import NullPrinter
>>> from coala.lib.output.printers.LogPrinter import LogPrinter
>>> log_printer = LogPrinter(NullPrinter())
>>> test_data = {"answer": 42}
>>> pickle_dump(log_printer, "test_project", test_data)
True
>>> pickle_load(log_printer, "test_project")
{'answer': 42}
>>> pickle_load(log_printer, "nonexistent_project")
>>> pickle_load(log_printer, "nonexistent_project", fallback=42)
42
```

#### Parameters

- **log\_printer** – A LogPrinter object to use for logging.
- **identifier** – The name of the file present in the user config directory.
- **fallback** – Return value to fallback to in case the file doesn't exist.

**Returns** Data that is present in the file, if the file exists. Otherwise the `default` value is returned.

`coala.lib.misc.CachingUtilities.settings_changed(log_printer, settings_hash)`

Determine if the settings have changed since the last run with caching.

#### Parameters

- **log\_printer** – A LogPrinter object to use for logging.
- **settings\_hash** – A MD5 hash that is unique to the settings used.

**Returns** Return True if the settings hash has changed Return False otherwise.

`coala.lib.misc.CachingUtilities.update_settings_db(log_printer, settings_hash)`

Update the config file last modification date.

#### Parameters

- **log\_printer** – A LogPrinter object to use for logging.
- **settings\_hash** – A MD5 hash that is unique to the settings used.

## coolib.misc.Compatibility module

## coolib.misc.Constants module

`coolib.misc.Constants.configure_logging()`  
Configures the logging with hard coded dictionary.

## coolib.misc.ContextManagers module

`coolib.misc.ContextManagers.change_directory(path)`

`coolib.misc.ContextManagers.make_temp(suffix='', prefix='tmp', dir=None)`  
Creates a temporary file with a closed stream and deletes it when done.

**Returns** A contextmanager retrieving the file path.

`coolib.misc.ContextManagers.prepare_file(lines, filename, force_linebreaks=True, create_tempfile=True, tempfile_kwargs={})`  
Can create a temporary file (if filename is None) with the lines. Can also add a trailing newline to each line specified if needed.

### Parameters

- **lines** – The lines from the file. (list or tuple of strings)
- **filename** – The filename to be prepared.
- **force\_linebreaks** – Whether to append newlines at each line if needed.
- **create\_tempfile** – Whether to save lines in tempfile if needed.
- **tempfile\_kwargs** – Kwargs passed to `tempfile.mkstemp()`.

`coolib.misc.ContextManagers.replace_stderr(replacement)`  
Replaces stderr with the replacement, yields back to the caller and then reverts everything back.

`coolib.misc.ContextManagers.replace_stdout(replacement)`  
Replaces stdout with the replacement, yields back to the caller and then reverts everything back.

`coolib.misc.ContextManagers.retrieve_stderr()`  
Yields a StringIO object from which one can read everything that was printed to stderr. (It won't be printed to the real stderr!)

Example usage:

```
with retrieve_stderr() as stderr: print("something") # Won't print to the console
what_was_printed = stderr.getvalue() # Save the value
```

`coolib.misc.ContextManagers.retrieve_stdout()`  
Yields a StringIO object from which one can read everything that was printed to stdout. (It won't be printed to the real stdout!)

Example usage:

```
with retrieve_stdout() as stdout: print("something") # Won't print to the console
what_was_printed = stdout.getvalue() # Save the value
```

`coolib.misc.ContextManagers.simulate_console_inputs(*inputs)`  
Does some magic to simulate the given inputs to any calls to the `input` builtin. This yields back an Input-Generator object so you can check which input was already used and append any additional inputs you want.  
Example:

```
with simulate_console_inputs(0, 1, 2) as generator: assert(input() == 0) as-
sert(generator.last_input == 0) generator.inputs.append(3) assert(input() == 1) assert(input() ==
2) assert(input() == 3) assert(generator.last_input == 3)
```

**Parameters** `inputs` – Any inputs to simulate.

**Raises** `ValueError` – Raised when was asked for more input but there's no more provided.

`coolib.misc.ContextManagers.subprocess_timeout(sub_process, seconds, kill_pg=False)`  
Kill subprocess if the sub process takes more the than the timeout.

**Parameters**

- **sub\_process** – The sub process to run.
- **seconds** – The number of seconds to allow the test to run for. If set to 0 or a negative value, it waits indefinitely. Floats can be used to specify units smaller than seconds.
- **kill\_pg** – Boolean whether to kill the process group or only this process. (not applicable for windows)

`coolib.misc.ContextManagers.suppress_stdout()`  
Suppresses everything going to stdout.

## coolib.misc.DictUtilities module

`coolib.misc.DictUtilities.add_pair_to_dict(key, value, dictionary)`  
Add (key, value) pair to the dictionary. The value is added to a list of values for the key.

`coolib.misc.DictUtilities.inverse_dicts(*dicts)`  
Inverts the dicts, e.g. {1: 2, 3: 4} and {2: 3, 4: 4} will be inverted {2: [1], 3: [2], 4: [3, 4]}. This also handles dictionaries with Iterable items as values e.g. {1: [1, 2, 3], 2: [3, 4, 5]} and {2: [1], 3: [2], 4: [3, 4]} will be inverted to {1: [1, 2], 2: [1, 3], 3: [1, 2, 4], 4: [2, 4], 5: [2]}. No order is preserved.

**Parameters** `dicts` – The dictionaries to invert.

**Returns** The inversed dictionary which merges all dictionaries into one.

`coolib.misc.DictUtilities.update_ordered_dict_key(dictionary, old_key, new_key)`

## coolib.misc.Enum module

`coolib.misc.Enum.enum(*sequential, **named)`

## coolib.misc.Exceptions module

`coolib.misc.Exceptions.get_exitcode(exception, log_printer=None)`

## coolib.misc.MutableValue module

`class coolib.misc.MutableValue.MutableValue(val=None)`  
Bases: object

## coolib.misc.Shell module

`coolib.misc.Shell.call_without_output = functools.partial(<function call>, stderr=-3, stdout=-3)`  
 Uses `subprocess.call` to execute a command, but suppresses the output and the errors.

`coolib.misc.Shell.get_shell_type()`  
 Finds the current shell type based on the outputs of common pre-defined variables in them. This is useful to identify which sort of escaping is required for strings.

**Returns** The shell type. This can be either “powershell” if Windows Powershell is detected, “cmd” if command prompt is been detected or “sh” if it’s neither of these.

`coolib.misc.Shell.run_interactive_shell_command(command, **kwargs)`  
 Runs a single command in shell and provides stdout, stderr and stdin streams.

This function creates a context manager that sets up the process (using `subprocess.Popen()`), returns to caller and waits for process to exit on leaving.

By default the process is opened in `universal_newlines` mode and creates pipes for all streams (stdout, stderr and stdin) using `subprocess.PIPE` special value. These pipes are closed automatically, so if you want to get the contents of the streams you should retrieve them before the context manager exits.

```
>>> with run_interactive_shell_command(["echo", "TEXT"]) as p:
...     stdout = p.stdout
...     stdout_text = stdout.read()
>>> stdout_text
'TEXT\n'
>>> stdout.closed
True
```

Custom streams provided are not closed except of `subprocess.PIPE`.

```
>>> from tempfile import TemporaryFile
>>> stream = TemporaryFile()
>>> with run_interactive_shell_command(["echo", "TEXT"],
...                                   stdout=stream) as p:
...     stderr = p.stderr
>>> stderr.closed
True
>>> stream.closed
False
```

### Parameters

- **command** – The command to run on shell. This parameter can either be a sequence of arguments that are directly passed to the process or a string. A string gets split beforehand using `shlex.split()`. If providing `shell=True` as a keyword-argument, no `shlex.split()` is performed and the command string goes directly to `subprocess.Popen()`.
- **kwargs** – Additional keyword arguments to pass to `subprocess.Popen` that are used to spawn the process.

**Returns** A context manager yielding the process started from the command.

`coolib.misc.Shell.run_shell_command(command, stdin=None, **kwargs)`  
 Runs a single command in shell and returns the read stdout and stderr data.

This function waits for the process (created using `subprocess.Popen()`) to exit. Effectively it wraps `run_interactive_shell_command()` and uses `communicate()` on the process.

See also `run_interactive_shell_command()`.

#### Parameters

- **command** – The command to run on shell. This parameter can either be a sequence of arguments that are directly passed to the process or a string. A string gets splitted beforehand using `shlex.split()`.
- **stdin** – Initial input to send to the process.
- **kwargs** – Additional keyword arguments to pass to `subprocess.Popen` that is used to spawn the process.

**Returns** A tuple with `(stdoutstring, stderrstring)`.

## Module contents

### 1.1.5 coalib.output package

#### Subpackages

#### coala.output.printers package

#### Submodules

#### coala.output.printers.LOG\_LEVEL module

#### coala.output.printers.ListLogPrinter module

```
class coalib.output.printers.ListLogPrinter.ListLogPrinter(log_level=30,      times-
                                                         tamp_format='%X')
    Bases: pyprint.Printer.Printer, coalib.output.printers.LogPrinter.LogPrinterMixin

    A ListLogPrinter is a log printer which collects all LogMessages to a list so that the logs can be used at a later
    time.

    log_message(log_message, **kwargs)
```

#### coala.output.printers.LogPrinter module

```
class coalib.output.printers.LogPrinter.LogPrinter(printer=None, log_level=10, times-
                                                         tamp_format='%X')
    Bases: coalib.output.printers.LogPrinter.LogPrinterMixin

    This class is deprecated and will be soon removed. To get logger use logging.getLogger(__name__). Make sure
    that you're getting it when the logging configuration is loaded.

    The LogPrinter class allows to print log messages to an underlying Printer.

    This class is an adapter, means you can create a LogPrinter from every existing Printer instance.

    log_level
        Returns current log_level used in logger.

    log_message(log_message, **kwargs)
```



## **printer**

Returns the underlying printer where logs are printed to.

**class** `coolib.output.printers.LogPrinter.LogPrinterMixin`

Bases: `object`

Provides access to the logging interfaces (e.g. `err`, `warn`, `info`) by routing them to the `log_message` method, which should be implemented by descendants of this class.

**debug** (*\*messages*, *\**, *delimiter*=' ', *timestamp*=None, *\*\*kwargs*)

**err** (*\*messages*, *\**, *delimiter*=' ', *timestamp*=None, *\*\*kwargs*)

**info** (*\*messages*, *\**, *delimiter*=' ', *timestamp*=None, *\*\*kwargs*)

**log** (*log\_level*, *message*, *timestamp*=None, *\*\*kwargs*)

**log\_exception** (*message*, *exception*, *log\_level*=40, *timestamp*=None, *\*\*kwargs*)

If the `log_level` of the printer is greater than `DEBUG`, it prints only the message. If it is `DEBUG` or lower, it shows the message along with the traceback of the exception.

### **Parameters**

- **message** – The message to print.
- **exception** – The exception to print.
- **log\_level** – The `log_level` of this message (not used when logging the traceback. Tracebacks always have a level of `DEBUG`).
- **timestamp** – The time at which this log occurred. Defaults to the current time.
- **kwargs** – Keyword arguments to be passed when logging the message (not used when logging the traceback).

**log\_message** (*log\_message*, *\*\*kwargs*)

It is your responsibility to implement this method, if you're using this mixin.

**warn** (*\*messages*, *\**, *delimiter*=' ', *timestamp*=None, *\*\*kwargs*)

## **Module contents**

This package holds printer objects. Printer objects are general purpose and not tied to `coala`.

If you need logging capabilities please take a look at the `LogPrinter` object which adds logging capabilities “for free” if used as base class for any other printer.

## **Submodules**

### **coolib.output.ConfWriter module**

**class** `coolib.output.ConfWriter.ConfWriter` (*file\_name*, *key\_value\_delimiters*=('=', ), *comment\_separators*=(' ', ), *key\_delimiters*=(' ', ' '), *section\_name\_surroundings*=mappingproxy({'I': 'I'}), *section\_override\_delimiters*=(' ', ), *unsavable\_keys*=('save', ))

Bases: `pyprint.ClosableObject.ClosableObject`

**static is\_comment** (*key*)

**write\_section** (*section*)

**write\_sections** (*sections*)

## coala.output.ConsoleInteraction module

**class** coalib.output.ConsoleInteraction.**BackgroundMessageStyle**

Bases: `pygments.style.Style`

**styles** = {Token.Keyword.Pseudo: “, Token.Literal.String.Heredoc: “, Token.Other: “, Token.Comment: “, Token.Na

**class** coalib.output.ConsoleInteraction.**BackgroundSourceRangeStyle**

Bases: `pygments.style.Style`

**styles** = {Token.Keyword.Pseudo: “, Token.Literal.String.Heredoc: “, Token.Other: “, Token.Comment: “, Token.Na

**class** coalib.output.ConsoleInteraction.**NoColorStyle**

Bases: `pygments.style.Style`

**styles** = {Token.Keyword.Pseudo: “, Token.Literal.String.Heredoc: “, Token.Other: “, Token.Comment: “, Token.Na

**coala.output.ConsoleInteraction.acquire\_actions\_and\_apply** (*console\_printer*, *section*, *file\_diff\_dict*, *result*, *file\_dict*, *cli\_actions=None*)

Acquires applicable actions and applies them.

### Parameters

- **console\_printer** – Object to print messages on the console.
- **section** – Name of section to which the result belongs.
- **file\_diff\_dict** – Dictionary containing filenames as keys and Diff objects as values.
- **result** – A derivative of Result.
- **file\_dict** – A dictionary containing all files with filename as key.
- **cli\_actions** – The list of cli actions available.

**coala.output.ConsoleInteraction.acquire\_settings** (*log\_printer*, *settings\_names\_dict*, *section*)

This method prompts the user for the given settings.

### Parameters

- **log\_printer** – Printer responsible for logging the messages. This is needed to comply with the interface.
- **settings\_names\_dict** – A dictionary with the settings name as key and a list containing a description in [0] and the name of the bears who need this setting in [1] and following.

Example:

```
{ "UseTabs": [ "describes whether tabs should be used instead of spaces",
               "SpaceConsistencyBear",
               "SomeOtherBear" ] }
```

**Parameters** **section** – The section the action corresponds to.

**Returns** A dictionary with the settings name as key and the given value as value.

```
coolib.output.ConsoleInteraction.ask_for_action_and_apply(console_printer,
                                                         section,           meta-
                                                         data_list,   action_dict,
                                                         failed_actions, result,
                                                         file_diff_dict,file_dict)
```

Asks the user for an action and applies it.

#### Parameters

- **console\_printer** – Object to print messages on the console.
- **section** – Currently active section.
- **metadata\_list** – Contains metadata for all the actions.
- **action\_dict** – Contains the action names as keys and their references as values.
- **failed\_actions** – A set of all actions that have failed. A failed action remains in the list until it is successfully executed.
- **result** – Result corresponding to the actions.
- **file\_diff\_dict** – If it is an action which applies a patch, this contains the diff of the patch to be applied to the file with filename as keys.
- **file\_dict** – Dictionary with filename as keys and its contents as values.

**Returns** Returns a boolean value. True will be returned, if it makes sense that the user may choose to execute another action, False otherwise.

```
coolib.output.ConsoleInteraction.choose_action(console_printer, actions)
```

Presents the actions available to the user and takes as input the action the user wants to choose.

#### Parameters

- **console\_printer** – Object to print messages on the console.
- **actions** – Actions available to the user.

**Returns** Return choice of action of user.

```
coolib.output.ConsoleInteraction.format_lines(lines, line_nr='')
```

```
coolib.output.ConsoleInteraction.get_action_info(section, action, failed_actions)
```

Gets all the required Settings for an action. It updates the section with the Settings.

#### Parameters

- **section** – The section the action corresponds to.
- **action** – The action to get the info for.
- **failed\_actions** – A set of all actions that have failed. A failed action remains in the list until it is successfully executed.

**Returns** Action name and the updated section.

```
coolib.output.ConsoleInteraction.highlight_text(no_color,           text,
                                                  lexer=<pygments.lexers.TextLexer>,
                                                  style=None)
```

```
coolib.output.ConsoleInteraction.join_names(values)
```

Produces a string by concatenating the items in *values* with commas, except the last element, which is concatenated with an “and”.

```
>>> join_names(["apples", "bananas", "oranges"])
'apples, bananas and oranges'
>>> join_names(["apples", "bananas"])
'apples and bananas'
>>> join_names(["apples"])
'apples'
```

**Parameters** **values** – A list of strings.

**Returns** The concatenated string.

`coalaib.output.ConsoleInteraction.nothing_done(log_printer)`

Will be called after processing a coafile when nothing had to be done, i.e. no section was enabled/targeted.

**Parameters** **log\_printer** – A LogPrinter object.

`coalaib.output.ConsoleInteraction.print_actions(console_printer, section, actions, failed_actions)`

Prints the given actions and lets the user choose.

**Parameters**

- **console\_printer** – Object to print messages on the console.
- **actions** – A list of FunctionMetadata objects.
- **failed\_actions** – A set of all actions that have failed. A failed action remains in the list until it is successfully executed.

**Returns** A tuple with the name member of the FunctionMetadata object chosen by the user and a Section containing at least all needed values for the action. If the user did choose to do nothing, return (None, None).

`coalaib.output.ConsoleInteraction.print_affected_files(console_printer, log_printer, result, file_dict)`

Prints all the affected files and affected lines within them.

**Parameters**

- **console\_printer** – Object to print messages on the console.
- **log\_printer** – Printer responsible for logging the messages.
- **result** – The result to print the context for.
- **file\_dict** – A dictionary containing all files with filename as key.

`coalaib.output.ConsoleInteraction.print_affected_lines(console_printer, file_dict, sourcerange)`

Prints the lines affected by the bears.

**Parameters**

- **console\_printer** – Object to print messages on the console.
- **file\_dict** – A dictionary containing all files with filename as key.
- **sourcerange** – The SourceRange object referring to the related lines to print.

`coalaib.output.ConsoleInteraction.print_bears(bears, show_description, show_params, console_printer)`

Presents all bears being used in a stylized manner.

**Parameters**

- **bears** – It's a dictionary with bears as keys and list of sections containing those bears as values.
- **show\_description** – True if the main description of the bears should be shown.
- **show\_params** – True if the parameters and their description should be shown.
- **console\_printer** – Object to print messages on the console.

`coolib.output.ConsoleInteraction.print_diffs_info` (*diffs, printer*)

`coolib.output.ConsoleInteraction.print_lines` (*console\_printer, file\_dict, sourcerange*)  
Prints the lines between the current and the result line. If needed they will be shortened.

#### Parameters

- **console\_printer** – Object to print messages on the console.
- **file\_dict** – A dictionary containing all files as values with filenames as key.
- **sourcerange** – The SourceRange object referring to the related lines to print.

`coolib.output.ConsoleInteraction.print_result` (*console\_printer, section, file\_diff\_dict, result, file\_dict, interactive=True*)

Prints the result to console.

#### Parameters

- **console\_printer** – Object to print messages on the console.
- **section** – Name of section to which the result belongs.
- **file\_diff\_dict** – Dictionary containing filenames as keys and Diff objects as values.
- **result** – A derivative of Result.
- **file\_dict** – A dictionary containing all files with filename as key.
- **interactive** – Variable to check whether or not to offer the user actions interactively.

`coolib.output.ConsoleInteraction.print_results` (*log\_printer, section, result\_list, file\_dict, file\_diff\_dict, console\_printer*)

Prints all the results in a section.

#### Parameters

- **log\_printer** – Printer responsible for logging the messages.
- **section** – The section to which the results belong to.
- **result\_list** – List containing the results
- **file\_dict** – A dictionary containing all files with filename as key.
- **file\_diff\_dict** – A dictionary that contains filenames as keys and diff objects as values.
- **console\_printer** – Object to print messages on the console.

`coolib.output.ConsoleInteraction.print_results_formatted` (*log\_printer, section, result\_list, \*args*)

`coolib.output.ConsoleInteraction.print_results_no_input` (*log\_printer, section, result\_list, file\_dict, file\_diff\_dict, console\_printer*)

Prints all non interactive results in a section

#### Parameters

- **log\_printer** – Printer responsible for logging the messages.
- **section** – The section to which the results belong to.
- **result\_list** – List containing the results
- **file\_dict** – A dictionary containing all files with filename as key.
- **file\_diff\_dict** – A dictionary that contains filenames as keys and diff objects as values.
- **console\_printer** – Object to print messages on the console.

`coolib.output.ConsoleInteraction.print_section_beginning(console_printer, section)`  
 Will be called after initialization current\_section in begin\_section()

#### Parameters

- **console\_printer** – Object to print messages on the console.
- **section** – The section that will get executed now.

`coolib.output.ConsoleInteraction.require_setting(setting_name, arr, section)`  
 This method is responsible for prompting a user about a missing setting and taking its value as input from the user.

#### Parameters

- **setting\_name** – Name of the setting missing
- **arr** – A list containing a description in [0] and the name of the bears who need this setting in [1] and following.
- **section** – The section the action corresponds to.

`coolib.output.ConsoleInteraction.show_bear(bear, show_description, show_params, console_printer)`

Displays all information about a bear.

#### Parameters

- **bear** – The bear to be displayed.
- **show\_description** – True if the main description should be shown.
- **show\_params** – True if the details should be shown.
- **console\_printer** – Object to print messages on the console.

`coolib.output.ConsoleInteraction.show_bears(local_bears, global_bears, show_description, show_params, console_printer)`

Extracts all the bears from each enabled section or the sections in the targets and passes a dictionary to the show\_bears\_callback method.

#### Parameters

- **local\_bears** – Dictionary of local bears with section names as keys and bear list as values.
- **global\_bears** – Dictionary of global bears with section names as keys and bear list as values.
- **show\_description** – True if the main description of the bears should be shown.
- **show\_params** – True if the parameters and their description should be shown.
- **console\_printer** – Object to print messages on the console.

```
coolib.output.ConsoleInteraction.show_enumeration(console_printer, title, items, inden-
                                                    tation, no_items_text)
```

This function takes as input an iterable object (preferably a list or a dict) and prints it in a stylized format. If the iterable object is empty, it prints a specific statement given by the user. An e.g :

```
<indentation>Title: <indentation> * Item 1 <indentation> * Item 2
```

#### Parameters

- **console\_printer** – Object to print messages on the console.
- **title** – Title of the text to be printed
- **items** – The iterable object.
- **indentation** – Number of spaces to indent every line by.
- **no\_items\_text** – Text printed when iterable object is empty.

```
coolib.output.ConsoleInteraction.show_language_bears_capabilities(language_bears_capabilities,
                                                                    con-
                                                                    sole_printer)
```

Displays what the bears can detect and fix.

#### Parameters

- **language\_bears\_capabilities** – Dictionary with languages as keys and their bears' capabilities as values. The capabilities are stored in a tuple of two elements where the first one represents what the bears can detect, and the second one what they can fix.
- **console\_printer** – Object to print messages on the console.

### coolib.output.Interactions module

```
coolib.output.Interactions.fail_acquire_settings(log_printer, settings_names_dict,
                                                    section)
```

This method throws an exception if any setting needs to be acquired.

#### Parameters

- **log\_printer** – Printer responsible for logging the messages.
- **settings** – A dictionary with the settings name as key and a list containing a description in [0] and the name of the bears who need this setting in [1] and following.

#### Raises

- **AssertionError** – If any setting is required.
- **TypeError** – If `settings_names_dict` is not a dictionary.

### coolib.output.JSONEncoder module

```
coolib.output.JSONEncoder.create_json_encoder(**kwargs)
```

## Module contents

### 1.1.6 coalib.parsing package

#### Submodules

#### coala.parsing.CliParsing module

`coala.parsing.CliParsing.check_conflicts` (*sections*)

Checks if there are any conflicting arguments passed.

**Parameters** **sections** – The {`section_name`: `section_object`} dictionary to check conflicts for.

**Returns** True if no conflicts occur.

**Raises** **SystemExit** – If there are conflicting arguments (exit code: 2)

`coala.parsing.CliParsing.parse_cli` (*arg\_list=None, origin='/home/docs/checkouts/readthedocs.org/user\_builds/coala/api/checkouts/0.9.1/docs', arg\_parser=None, key\_value\_delimiters=('=', ':'), comment\_seperators=(), key\_delimiters=(' ', ), section\_override\_delimiters=('.', ))*)

Parses the CLI arguments and creates sections out of it.

#### Parameters

- **arg\_list** – The CLI argument list.
- **origin** – Directory used to interpret relative paths given as argument.
- **arg\_parser** – Instance of ArgParser that is used to parse none-setting arguments.
- **key\_value\_delimiters** – Delimiters to separate key and value in setting arguments.
- **comment\_seperators** – Allowed prefixes for comments.
- **key\_delimiters** – Delimiter to separate multiple keys of a setting argument.
- **section\_override\_delimiters** – The delimiter to delimit the section from the key name (e.g. the '.' in `sect.key = value`).

**Returns** A dictionary holding section names as keys and the sections themselves as value.

`coala.parsing.CliParsing.parse_custom_settings` (*sections, custom\_settings\_list, origin, line\_parser*)

Parses the custom settings given to coala via `-S something=value`.

#### Parameters

- **sections** – The Section dictionary to add to (mutable).
- **custom\_settings\_list** – The list of settings strings.
- **origin** – The originating directory.
- **line\_parser** – The LineParser to use.



## coolib.parsing.ConfParser module

```
class coalib.parsing.ConfParser.ConfParser(key_value_delimiters=('=', ),
                                             comment_seperators=(' ', ), key_delimiters=(' ', ),
                                             section_name_surroundings=mappingproxy({'[':
                                             '']}), remove_empty_iter_elements=True)
```

Bases: object

**get\_section** (name, create\_if\_not\_exists=False)

**parse** (input\_data, overwrite=False)

Parses the input and adds the new data to the existing.

### Parameters

- **input\_data** – The filename to parse from.
- **overwrite** – If True, wipes all existing Settings inside this instance and adds only the newly parsed ones. If False, adds the newly parsed data to the existing one (and overwrites already existing keys with the newly parsed values).

**Returns** A dictionary with (lowercase) section names as keys and their Setting objects as values.

## coolib.parsing.DefaultArgParser module

```
class coalib.parsing.DefaultArgParser.CustomFormatter(prog,
                                                        indent_increment=2,
                                                        max_help_position=24,
                                                        width=None)
```

Bases: argparse.RawDescriptionHelpFormatter

A Custom Formatter that will keep the metavar in the usage but remove them in the more detailed arguments section.

```
coolib.parsing.DefaultArgParser.default_arg_parser (formatter_class=None)
```

This function creates an ArgParser to parse command line arguments.

**Parameters** **formatter\_class** – Formatting the arg\_parser output into a specific form. For example: In the manpage format.

## coolib.parsing.Globbing module

```
coolib.parsing.Globbing.fnmatch (name, globs)
```

Tests whether name matches one of the given globs.

### Parameters

- **name** – File or directory name
- **globs** – Glob string with wildcards or list of globs

**Returns** Boolean: Whether or not name is matched by glob

Glob Syntax:

- **'[seq]'**: Matches any character in seq. Cannot be empty. Any special character loses its special meaning in a set.
- **'[!seq]'**: Matches any character not in seq. Cannot be empty. Any special character loses its special meaning in a set.

- **‘(seq\_alseq\_b)’**: Matches either *sequence\_a* or *sequence\_b* as a whole. More than two or just one sequence can be given.
- **‘?’**: Matches any single character.
- **‘\*’**: Matches everything but *os.sep*.
- **‘\*\*’**: Matches everything.

`coala.lib.parsing.Globbing.glob(pattern)`

Iterates all filesystem paths that get matched by the glob pattern. Syntax is equal to that of `fnmatch`.

**Parameters** *pattern* – Glob pattern with wildcards

**Returns** List of all file names that match pattern

`coala.lib.parsing.Globbing.glob_escape(input_string)`

Escapes the given string with `[c]` pattern. Examples:

```
>>> from coala.lib.parsing.Globbing import glob_escape
>>> glob_escape('test (1)')
'test [(1)]'
>>> glob_escape('test folder?')
'test folder[?]'
>>> glob_escape('test*folder')
'test[*]folder'
```

**Parameters** *input\_string* – String that is to be escaped with `[ ]`.

**Returns** Escaped string in which all the special glob characters `()[]|?*` are escaped.

`coala.lib.parsing.Globbing.has_wildcard(pattern)`

Checks whether pattern has any wildcards.

**Parameters** *pattern* – Glob pattern that may contain wildcards

**Returns** Boolean: Whether or not there are wildcards in pattern

`coala.lib.parsing.Globbing.iglob(pattern)`

Iterates all filesystem paths that get matched by the glob pattern. Syntax is equal to that of `fnmatch`.

**Parameters** *pattern* – Glob pattern with wildcards

**Returns** Iterator that yields all file names that match pattern

`coala.lib.parsing.Globbing.relative_flat_glob(dirname, basename)`

Non-recursive glob for one directory. Does not accept wildcards.

**Parameters**

- **dirname** – Directory name
- **basename** – Basename of a file in dir of *dirname*

**Returns** List containing Basename if the file exists

`coala.lib.parsing.Globbing.relative_recursive_glob(dirname, pattern)`

Recursive Glob for one directory and all its (nested) subdirectories. Accepts only `‘**’` as pattern.

**Parameters**

- **dirname** – Directory name
- **pattern** – The recursive wildcard `‘**’`

**Returns** Iterator that yields all the (nested) subdirectories of the given dir

`coala.lib.parsing.Globbing.relative_wildcard_glob` (*dirname, pattern*)

Non-recursive glob for one directory. Accepts wildcards.

**Parameters**

- **dirname** – Directory name
- **pattern** – Glob pattern with wildcards

**Returns** List of files in the dir of *dirname* that match the pattern

`coala.lib.parsing.Globbing.translate` (*pattern*)

Translates a pattern into a regular expression.

**Parameters** **pattern** – Glob pattern with wildcards

**Returns** Regular expression with the same meaning

## coala.lib.parsing.LineParser module

```
class coala.lib.parsing.LineParser.LineParser(key_value_delimiters=('=', ), comment_separators=(' ', ), key_delimiters=(' ', ), section_name_surroundings=None, section_override_delimiters=('.', ))
```

Bases: object

**parse** (*line*)

Note that every value in the returned tuple *besides the value* is unescaped. This is so since the value is meant to be put into a Setting later thus the escapes may be needed there.

**Parameters** **line** – The line to parse.

**Returns** section\_name (empty string if it's no section name), [(section\_override, key), ...], value, comment

## Module contents

The StringProcessing module contains various functions for extracting information out of strings.

Most of them support regexes for advanced pattern matching.

## 1.1.7 coala.lib.processes package

### Subpackages

**coala.lib.processes.communication package**

### Submodules

**coala.lib.processes.communication.LogMessage module**

```
class coala.lib.processes.communication.LogMessage.LogMessage(log_level, *messages, *, delimiter=' ', timestamp=None)
```

Bases: object

**to\_string\_dict()**

Makes a dictionary which has all keys and values as strings and contains all the data that the LogMessage has.

**Returns** Dictionary with keys and values as string.

## Module contents

## Submodules

### coala.processes.BearRunning module

`coala.processes.BearRunning.get_global_dependency_results` (*global\_result\_dict*, *bear\_instance*)

This method gets all the results originating from the dependencies of a `bear_instance`. Each `bear_instance` may or may not have dependencies.

**Parameters** `global_result_dict` – The list of results out of which the dependency results are picked.

**Returns** None if bear has no dependencies, False if dependencies are not met, the dependency dict otherwise.

`coala.processes.BearRunning.get_local_dependency_results` (*local\_result\_list*, *bear\_instance*)

This method gets all the results originating from the dependencies of a `bear_instance`. Each `bear_instance` may or may not have dependencies.

#### Parameters

- **local\_result\_list** – The list of results out of which the dependency results are picked.
- **bear\_instance** – The instance of a local bear to get the dependencies from.

**Returns** Return none if there are no dependencies for the bear. Else return a dictionary containing dependency results.

`coala.processes.BearRunning.get_next_global_bear` (*timeout*, *global\_bear\_queue*, *global\_bear\_list*, *global\_result\_dict*)

Retrieves the next global bear.

#### Parameters

- **timeout** – The queue blocks at most timeout seconds for a free slot to execute the put operation on. After the timeout it returns queue Full exception.
- **global\_bear\_queue** – queue (read, write) of indexes of global bear instances in the `global_bear_list`.
- **global\_bear\_list** – A list containing all global bears to be executed.
- **global\_result\_dict** – A `Manager.dict` that will be used to store global results. The list of results of one global bear will be stored with the bear name as key.

**Returns** (bear, bearname, dependency\_results)

```
coilib.processes.BearRunning.run(file_name_queue, local_bear_list, global_bear_list,
                                global_bear_queue, file_dict, local_result_dict,
                                global_result_dict, message_queue, control_queue, time-
                                out=0)
```

This is the method that is actually runs by processes.

If parameters type is ‘queue (read)’ this means it has to implement the get(timeout=TIMEOUT) method and it shall raise queue.Empty if the queue is empty up until the end of the timeout. If the queue has the (optional!) task\_done() attribute, the run method will call it after processing each item.

If parameters type is ‘queue (write)’ it shall implement the put(object, timeout=TIMEOUT) method.

If the queues raise any exception not specified here the user will get an ‘unknown error’ message. So beware of that.

### Parameters

- **file\_name\_queue** – queue (read) of file names to check with local bears. Each invocation of the run method needs one such queue which it checks with all the local bears. The queue could be empty. (Repeat until queue empty.)
- **local\_bear\_list** – List of local bear instances.
- **global\_bear\_list** – List of global bear instances.
- **global\_bear\_queue** – queue (read, write) of indexes of global bear instances in the global\_bear\_list.
- **file\_dict** – dict of all files as {filename:file}, file as in file.readlines().
- **local\_result\_dict** – A Manager.dict that will be used to store local results. A list of all local results. will be stored with the filename as key.
- **global\_result\_dict** – A Manager.dict that will be used to store global results. The list of results of one global bear will be stored with the bear name as key.
- **message\_queue** – queue (write) for debug/warning/error messages (type LogMessage)
- **control\_queue** – queue (write). If any result gets written to the result\_dict a tuple containing a CONTROL\_ELEMENT (to indicate what kind of event happened) and either a bear name (for global results) or a file name to indicate the result will be put to the queue. If the run method finished all its local bears it will put (CONTROL\_ELEMENT.LOCAL\_FINISHED, None) to the queue, if it finished all global ones, (CONTROL\_ELEMENT.GLOBAL\_FINISHED, None) will be put there.
- **timeout** – The queue blocks at most timeout seconds for a free slot to execute the put operation on. After the timeout it returns queue Full exception.

```
coilib.processes.BearRunning.run_bear(message_queue, timeout, bear_instance, *args,
                                     **kwargs)
```

This method is responsible for executing the instance of a bear. It also reports or logs errors if any occur during the execution of that bear instance.

### Parameters

- **message\_queue** – A queue that contains messages of type errors/warnings/debug statements to be printed in the Log.
- **timeout** – The queue blocks at most timeout seconds for a free slot to execute the put operation on. After the timeout it returns queue Full exception.
- **bear\_instance** – The instance of the bear to be executed.
- **args** – The arguments that are to be passed to the bear.

- **kwargs** – The keyword arguments that are to be passed to the bear.

**Returns** Returns a valid list of objects of the type Result if the bear executed successfully. None otherwise.

```
coolib.processes.BearRunning.run_global_bear(message_queue,          timeout,
                                             global_bear_instance,    depen-
                                             dency_results)
```

Runs an instance of a global bear. Checks if bear\_instance is of type GlobalBear and then passes it to the run\_bear to execute.

#### Parameters

- **message\_queue** – A queue that contains messages of type errors/warnings/debug statements to be printed in the Log.
- **timeout** – The queue blocks at most timeout seconds for a free slot to execute the put operation on. After the timeout it returns queue Full exception.
- **global\_bear\_instance** – Instance of GlobalBear to run.
- **dependency\_results** – The results of all the bears on which the instance of the passed bear to be run depends on.

**Returns** Returns a list of results generated by the passed bear\_instance.

```
coolib.processes.BearRunning.run_global_bears(message_queue,          timeout,
                                              global_bear_queue,    global_bear_list,
                                              global_result_dict, control_queue)
```

Run all global bears.

#### Parameters

- **message\_queue** – A queue that contains messages of type errors/warnings/debug statements to be printed in the Log.
- **timeout** – The queue blocks at most timeout seconds for a free slot to execute the put operation on. After the timeout it returns queue Full exception.
- **global\_bear\_queue** – queue (read, write) of indexes of global bear instances in the global\_bear\_list.
- **global\_bear\_list** – list of global bear instances
- **global\_result\_dict** – A Manager.dict that will be used to store global results. The list of results of one global bear will be stored with the bear name as key.
- **control\_queue** – If any result gets written to the result\_dict a tuple containing a CONTROL\_ELEMENT (to indicate what kind of event happened) and either a bear name(for global results) or a file name to indicate the result will be put to the queue.

```
coolib.processes.BearRunning.run_local_bear(message_queue, timeout, local_result_list,
                                             file_dict, bear_instance, filename)
```

Runs an instance of a local bear. Checks if bear\_instance is of type LocalBear and then passes it to the run\_bear to execute.

#### Parameters

- **message\_queue** – A queue that contains messages of type errors/warnings/debug statements to be printed in the Log.
- **timeout** – The queue blocks at most timeout seconds for a free slot to execute the put operation on. After the timeout it returns queue Full exception.
- **local\_result\_list** – Its a list that stores the results of all local bears.

- **file\_dict** – Dictionary containing contents of file.
- **bear\_instance** – Instance of LocalBear the run.
- **filename** – Name of the file to run it on.

**Returns** Returns a list of results generated by the passed bear\_instance.

```
coilib.processes.BearRunning.run_local_bears(filename_queue, message_queue, timeout,
                                             file_dict, local_bear_list, local_result_dict,
                                             control_queue)
```

Run local bears on all the files given.

#### Parameters

- **filename\_queue** – queue (read) of file names to check with local bears.
- **message\_queue** – A queue that contains messages of type errors/warnings/debug statements to be printed in the Log.
- **timeout** – The queue blocks at most timeout seconds for a free slot to execute the put operation on. After the timeout it returns queue Full exception.
- **file\_dict** – Dictionary that contains contents of files.
- **local\_bear\_list** – List of local bears to run.
- **local\_result\_dict** – A Manager.dict that will be used to store local bear results. A list of all local bear results will be stored with the filename as key.
- **control\_queue** – If any result gets written to the result\_dict a tuple containing a CONTROL\_ELEMENT (to indicate what kind of event happened) and either a bear name(for global results) or a file name to indicate the result will be put to the queue.

```
coilib.processes.BearRunning.run_local_bears_on_file(message_queue, timeout,
                                                    file_dict, local_bear_list, local_result_dict, control_queue,
                                                    filename)
```

This method runs a list of local bears on one file.

#### Parameters

- **message\_queue** – A queue that contains messages of type errors/warnings/debug statements to be printed in the Log.
- **timeout** – The queue blocks at most timeout seconds for a free slot to execute the put operation on. After the timeout it returns queue Full exception.
- **file\_dict** – Dictionary that contains contents of files.
- **local\_bear\_list** – List of local bears to run on file.
- **local\_result\_dict** – A Manager.dict that will be used to store local bear results. A list of all local bear results will be stored with the filename as key.
- **control\_queue** – If any result gets written to the result\_dict a tuple containing a CONTROL\_ELEMENT (to indicate what kind of event happened) and either a bear name(for global results) or a file name to indicate the result will be put to the queue.
- **filename** – The name of file on which to run the bears.

```
coilib.processes.BearRunning.send_msg(message_queue, timeout, log_level, *args, *, delimiter=' ', end='')
```

Puts message into message queue for a LogPrinter to present to the user.

#### Parameters

- **message\_queue** – The queue to put the message into and which the LogPrinter reads.
- **timeout** – The queue blocks at most timeout seconds for a free slot to execute the put operation on. After the timeout it returns queue Full exception.
- **log\_level** – The log\_level i.e Error, Debug or Warning. It is sent to the LogPrinter depending on the message.
- **args** – This includes the elements of the message.
- **delimiter** – It is the value placed between each arg. By default it is a ‘ ‘.
- **end** – It is the value placed at the end of the message.

`coolib.processes.BearRunning.task_done(obj)`

Invokes task\_done if the given queue provides this operation. Otherwise passes silently.

**Parameters** `obj` – Any object.

`coolib.processes.BearRunning.validate_results(message_queue, timeout, result_list, name, args, kwargs)`

Validates if the result\_list passed to it contains valid set of results. That is the result\_list must itself be a list and contain objects of the instance of Result object. If any irregularity is found a message is put in the message\_queue to present the irregularity to the user. Each result\_list belongs to an execution of a bear.

**Parameters**

- **message\_queue** – A queue that contains messages of type errors/warnings/debug statements to be printed in the Log.
- **timeout** – The queue blocks at most timeout seconds for a free slot to execute the put operation on. After the timeout it returns queue Full exception.
- **result\_list** – The list of results to validate.
- **name** – The name of the bear executed.
- **args** – The args with which the bear was executed.
- **kwargs** – The kwargs with which the bear was executed.

**Returns** Returns None if the result\_list is invalid. Else it returns the result\_list itself.

## coolib.processes.CONTROL\_ELEMENT module

## coolib.processes.LogPrinterThread module

`class coolib.processes.LogPrinterThread.LogPrinterThread(message_queue, log_printer)`

Bases: `threading.Thread`

This is the Thread object that outputs all log messages it gets from its message\_queue. Setting `obj.running = False` will stop within the next 0.1 seconds.

**run()**

## coolib.processes.Processing module

`coolib.processes.Processing.autoapply_actions(results, file_dict, file_diff_dict, section, log_printer)`

Auto-applies actions like defined in the given section.

**Parameters**



- **results** – A list of results.
- **file\_dict** – A dictionary containing the name of files and its contents.
- **file\_diff\_dict** – A dictionary that contains filenames as keys and diff objects as values.
- **section** – The section.
- **log\_printer** – A log printer instance to log messages on.

**Returns** A list of unprocessed results.

`coilib.processes.Processing.check_result_ignore(result, ignore_ranges)`  
Determines if the result has to be ignored.

#### Parameters

- **result** – The result that needs to be checked.
- **ignore\_ranges** – A list of tuples, each containing a list of lower cased affected bear-names and a `SourceRange` to ignore. If any of the bearname lists is empty, it is considered an ignore range for all bears. This may be a list of globbed bear wildcards.

**Returns** True if the result has to be ignored.

`coilib.processes.Processing.create_process_group(command_array, **kwargs)`  
`coilib.processes.Processing.execute_section(section, global_bear_list, local_bear_list, print_results, cache, log_printer, console_printer)`

Executes the section with the given bears.

The `execute_section` method does the following things:

- 1.Prepare a Process - Load files - Create queues
- 2.Spawn up one or more Processes
- 3.Output results from the Processes
- 4.Join all processes

#### Parameters

- **section** – The section to execute.
- **global\_bear\_list** – List of global bears belonging to the section. Dependencies are already resolved.
- **local\_bear\_list** – List of local bears belonging to the section. Dependencies are already resolved.
- **print\_results** – Prints all given results appropriate to the output medium.
- **cache** – An instance of `misc.Caching.FileCache` to use as a file cache buffer.
- **log\_printer** – The `log_printer` to warn to.
- **console\_printer** – Object to print messages on the console.

**Returns** Tuple containing a bool (True if results were yielded, False otherwise), a `Manager.dict` containing all local results (filenames are key) and a `Manager.dict` containing all global bear results (bear names are key) as well as the file dictionary.

`coilib.processes.Processing.fill_queue(queue_fill, any_list)`  
Takes element from a list and populates a queue with those elements.

#### Parameters

- **queue\_fill** – The queue to be filled.
- **any\_list** – List containing the elements.

`coala.lib.processes.Processing.filter_raising_callables` (*it, exception, \*args, \*\*kwargs*)

Filters all callable items inside the given iterator that raise the given exceptions.

#### Parameters

- **it** – The iterator to filter.
- **exception** – The (tuple of) exception(s) to filter for.
- **args** – Positional arguments to pass to the callable.
- **kwargs** – Keyword arguments to pass to the callable.

`coala.lib.processes.Processing.get_cpu_count` ()

`coala.lib.processes.Processing.get_default_actions` (*section*)

Parses the key `default_actions` in the given section.

**Parameters** **section** – The section where to parse from.

**Returns** A dict with the bename as keys and their default actions as values and another dict that contains bears and invalid action names.

`coala.lib.processes.Processing.get_file_dict` (*filename\_list, log\_printer*)

Reads all files into a dictionary.

#### Parameters

- **filename\_list** – List of names of paths to files to get contents of.
- **log\_printer** – The logger which logs errors.

**Returns** Reads the content of each file into a dictionary with filenames as keys.

`coala.lib.processes.Processing.get_file_list` (*results*)

Get the set of files that are affected in the given results.

**Parameters** **results** – A list of results from which the list of files is to be extracted.

**Returns** A set of file paths containing the mentioned list of files.

`coala.lib.processes.Processing.get_ignore_scope` (*line, keyword*)

Retrieves the bears that are to be ignored defined in the given line.

#### Parameters

- **line** – The line containing the ignore declaration.
- **keyword** – The keyword that was found. Everything after the rightmost occurrence of it will be considered for the scope.

**Returns** A list of lower cased bearnames or an empty list (-> “all”)

`coala.lib.processes.Processing.get_running_processes` (*processes*)

`coala.lib.processes.Processing.instantiate_bears` (*section, local\_bear\_list, global\_bear\_list, file\_dict, message\_queue, console\_printer*)

Instantiates each bear with the arguments it needs.

#### Parameters

- **section** – The section the bears belong to.

- **local\_bear\_list** – List of local bear classes to instantiate.
- **global\_bear\_list** – List of global bear classes to instantiate.
- **file\_dict** – Dictionary containing filenames and their contents.
- **message\_queue** – Queue responsible to maintain the messages delivered by the bears.
- **console\_printer** – Object to print messages on the console.

**Returns** The local and global bear instance lists.

`coolib.processes.Processing.instantiate_processes` (*section*, *local\_bear\_list*,  
*global\_bear\_list*, *job\_count*, *cache*,  
*log\_printer*, *console\_printer*)

Instantiate the number of processes that will run bears which will be responsible for running bears in a multi-processing environment.

#### Parameters

- **section** – The section the bears belong to.
- **local\_bear\_list** – List of local bears belonging to the section.
- **global\_bear\_list** – List of global bears belonging to the section.
- **job\_count** – Max number of processes to create.
- **cache** – An instance of `misc.Caching.FileCache` to use as a file cache buffer.
- **log\_printer** – The log printer to warn to.
- **console\_printer** – Object to print messages on the console.

**Returns** A tuple containing a list of processes, and the arguments passed to each process which are the same for each object.

`coolib.processes.Processing.print_result` (*results*, *file\_dict*, *retval*, *print\_results*, *section*,  
*log\_printer*, *file\_diff\_dict*, *ignore\_ranges*, *console\_printer*)

Takes the results produced by each bear and gives them to the `print_results` method to present to the user.

#### Parameters

- **results** – A list of results.
- **file\_dict** – A dictionary containing the name of files and its contents.
- **retval** – It is True if no results were yielded ever before. If it is False this function will return False no matter what happens. Else it depends on if this invocation yields results.
- **print\_results** – A function that prints all given results appropriate to the output medium.
- **file\_diff\_dict** – A dictionary that contains filenames as keys and diff objects as values.
- **ignore\_ranges** – A list of `SourceRanges`. Results that affect code in any of those ranges will be ignored.
- **console\_printer** – Object to print messages on the console.

**Returns** Returns False if any results were yielded. Else True.

`coala.lib.processes.Processing.process_queues` (*processes, control\_queue, local\_result\_dict, global\_result\_dict, file\_dict, print\_results, section, cache, log\_printer, console\_printer*)

Iterate the control queue and send the results received to the `print_result` method so that they can be presented to the user.

#### Parameters

- **processes** – List of processes which can be used to run Bears.
- **control\_queue** – Containing control elements that indicate whether there is a result available and which bear it belongs to.
- **local\_result\_dict** – Dictionary containing results respective to local bears. It is modified by the processes i.e. results are added to it by multiple processes.
- **global\_result\_dict** – Dictionary containing results respective to global bears. It is modified by the processes i.e. results are added to it by multiple processes.
- **file\_dict** – Dictionary containing file contents with filename as keys.
- **print\_results** – Prints all given results appropriate to the output medium.
- **cache** – An instance of `misc.Caching.FileCache` to use as a file cache buffer.

**Returns** Return True if all bears execute successfully and Results were delivered to the user. Else False.

`coala.lib.processes.Processing.simplify_section_result` (*section\_result*)

Takes in a section's result from `execute_section` and simplifies it for easy usage in other functions.

**Parameters** **section\_result** – The result of a section which was executed.

**Returns** Tuple containing: - bool - True if results were yielded - bool - True if unfixed results were yielded - list - Results from all bears (local and global)

`coala.lib.processes.Processing.yield_ignore_ranges` (*file\_dict*)

Yields tuples of affected bears and a `SourceRange` that shall be ignored for those.

**Parameters** **file\_dict** – The file dictionary.

## Module contents

### 1.1.8 coala.results package

#### Subpackages

#### coala.results.result\_actions package

#### Submodules

#### coala.results.result\_actions.ApplyPatchAction module

**class** `coala.results.result_actions.ApplyPatchAction.ApplyPatchAction`

Bases: `coala.results.result_actions.ResultAction.ResultAction`

**SUCCESS\_MESSAGE** = 'Patch applied successfully.'

**apply** (*result, original\_file\_dict, file\_diff\_dict, no\_orig: bool = False*)  
Apply patch

Parameters **no\_orig** – Whether or not to create .orig backup files

**static is\_applicable** (*result, original\_file\_dict, file\_diff\_dict*)

### coolib.results.result\_actions.IgnoreResultAction module

**class** coalib.results.result\_actions.IgnoreResultAction.**IgnoreResultAction**

Bases: *coolib.results.result\_actions.ResultAction.ResultAction*

**SUCCESS\_MESSAGE** = 'An ignore comment was added to your source code.'

**apply** (*result, original\_file\_dict, file\_diff\_dict, language: str, no\_orig: bool = False*)

Add ignore comment

**static is\_applicable** (*result, original\_file\_dict, file\_diff\_dict*)

For being applicable, the result has to point to a number of files that have to exist i.e. have not been previously deleted.

### coolib.results.result\_actions.OpenEditorAction module

**class** coalib.results.result\_actions.OpenEditorAction.**OpenEditorAction**

Bases: *coolib.results.result\_actions.ResultAction.ResultAction*

**SUCCESS\_MESSAGE** = 'Changes saved successfully.'

**apply** (*result, original\_file\_dict, file\_diff\_dict, editor: str*)

Open file(s)

Parameters **editor** – The editor to open the file with.

**static is\_applicable** (*result, original\_file\_dict, file\_diff\_dict*)

For being applicable, the result has to point to a number of files that have to exist i.e. have not been previously deleted.

### coolib.results.result\_actions.PrintDebugMessageAction module

**class** coalib.results.result\_actions.PrintDebugMessageAction.**PrintDebugMessageAction**

Bases: *coolib.results.result\_actions.ResultAction.ResultAction*

**apply** (*result, original\_file\_dict, file\_diff\_dict*)

Print debug message

**static is\_applicable** (*result, original\_file\_dict, file\_diff\_dict*)

### coolib.results.result\_actions.PrintMoreInfoAction module

**class** coalib.results.result\_actions.PrintMoreInfoAction.**PrintMoreInfoAction**

Bases: *coolib.results.result\_actions.ResultAction.ResultAction*

**apply** (*result, original\_file\_dict, file\_diff\_dict*)

Print more info

**static is\_applicable** (*result, original\_file\_dict, file\_diff\_dict*)

## coala.results.result\_actions.ResultAction module

A ResultAction is an action that is applicable to at least some results. This file serves the base class for all result actions, thus providing a unified interface for all actions.

**class** `coala.results.result_actions.ResultAction.ResultAction`

Bases: `object`

**SUCCESS\_MESSAGE** = 'The action was executed successfully.'

**apply** (*result*, *original\_file\_dict*, *file\_diff\_dict*, *\*\*kwargs*)

No description. Something went wrong.

**apply\_from\_section** (*result*, *original\_file\_dict*: *dict*, *file\_diff\_dict*: *dict*, *section*: *coala.settings.Section.Section*)

Applies this action to the given results with all additional options given as a section. The file dictionaries are needed for differential results.

### Parameters

- **result** – The result to apply.
- **original\_file\_dict** – A dictionary containing the files in the state where the result was generated.
- **file\_diff\_dict** – A dictionary containing a diff for every file from the state in the `original_file_dict` to the current state. This dict will be altered so you do not need to use the return value.
- **section** – The section where to retrieve the additional information.

**Returns** The modified `file_diff_dict`.

**classmethod** `get_metadata()`

Retrieves metadata for the apply function. The description may be used to advertise this action to the user. The parameters and their help texts are additional information that are needed from the user. You can create a section out of the inputs from the user and use `apply_from_section` to apply

:return A `FunctionMetadata` object.

**static** `is_applicable(result, original_file_dict, file_diff_dict)`

Checks whether the Action is valid for the result type.

Returns `True` by default.

### Parameters

- **result** – The result from the coala run to check if an Action is applicable.
- **original\_file\_dict** – A dictionary containing the files in the state where the result was generated.
- **file\_diff\_dict** – A dictionary containing a diff for every file from the state in the `original_file_dict` to the current state. This dict will be altered so you do not need to use the return value.

## coala.results.result\_actions.ShowPatchAction module

**class** `coala.results.result_actions.ShowPatchAction.ShowPatchAction`

Bases: `coala.results.result_actions.ResultAction.ResultAction`

**SUCCESS\_MESSAGE** = 'Displayed patch successfully.'

**apply** (*result, original\_file\_dict, file\_diff\_dict, colored: bool = True, show\_result\_on\_top: bool = False*)  
 Show patch

#### Parameters

- **colored** – Whether or not to use colored output.
- **show\_result\_on\_top** – Set this to True if you want to show the result info on top. (Useful for e.g. coala.ci.)

**static is\_applicable** (*result, original\_file\_dict, file\_diff\_dict*)

```
coolib.results.result_actions.ShowPatchAction.format_line(line,      real_nr='',
                                                         sign='|',  mod_nr='',
                                                         symbol='')
coolib.results.result_actions.ShowPatchAction.print_beautified_diff(difflines,
                                                                    printer)
coolib.results.result_actions.ShowPatchAction.print_from_name(printer, line)
coolib.results.result_actions.ShowPatchAction.print_to_name(printer, line)
```

## Module contents

The result\_actions package holds objects deriving from ResultAction. A ResultAction represents an action that can be applied to a result.

## Submodules

### coolib.results.AbsolutePosition module

**class** coolib.results.AbsolutePosition.**AbsolutePosition** (*text: (<class 'tuple'>, <class 'list'>, None) = None, position: (<class 'int'>, None) = None)*

Bases: *coolib.results.TextPosition.TextPosition*

#### position

coolib.results.AbsolutePosition.**calc\_line\_col** (*text, position*)

Creates a tuple containing (line, column) by calculating line number and column in the text, from position.

The position represents the index of a character. In the following example 'a' is at position '0' and its corresponding line and column are:

```
>>> calc_line_col(('a\n',), 0)
(1, 1)
```

All special characters(including the newline character) belong in the same line, and have their own position. A line is an item in the tuple:

```
>>> calc_line_col(('a\n', 'b\n'), 1)
(1, 2)
>>> calc_line_col(('a\n', 'b\n'), 2)
(2, 1)
```

#### Parameters

- **text** – A tuple/list of lines in which position is to be calculated.

- **position** – Position (starting from 0) of character to be found in the (line, column) form.

**Returns** A tuple of the form (line, column), where both line and column start from 1.

## coala.results.Diff module

**class** coalib.results.Diff.**Diff** (*file\_list*, *rename=False*, *delete=False*)

Bases: object

A Diff result represents a difference for one file.

**add\_lines** (*line\_nr\_before*, *lines*)

Adds lines after the given line number.

### Parameters

- **line\_nr\_before** – Line number of the line before the additions. Use 0 for insert lines before everything.
- **lines** – A list of lines to add.

**affected\_code** (*filename*)

Creates a list of SourceRange objects which point to the related code. Changes on continuous lines will be put into one SourceRange.

**Parameters** **filename** – The filename to associate the SourceRange's to.

**Returns** A list of all related SourceRange objects.

**change\_line** (*line\_nr*, *original\_line*, *replacement*)

Changes the given line with the given line number. The replacement will be there instead.

Given an empty diff object:

```
>>> diff = Diff(['Hey there! Gorgeous.\n',
...              "It's nice that we're here.\n"])
```

We can change a line easily:

```
>>> diff.change_line(1,
...                  'Hey there! Gorgeous.\n',
...                  'Hey there! This is sad.\n')
>>> diff.modified
['Hey there! This is sad.\n', "It's nice that we're here.\n"]
```

We can even merge changes within one line:

```
>>> diff.change_line(1,
...                  'Hey there! Gorgeous.\n',
...                  'Hello. :( Gorgeous.\n')
>>> diff.modified
['Hello. :( This is sad.\n', "It's nice that we're here.\n"]
```

However, if we change something that has been changed before, we'll get a conflict:

```
>>> diff.change_line(1, # +ELLIPSIS
...                  'Hey there! Gorgeous.\n',
...                  'Hello. This is not ok. Gorgeous.\n')
Traceback (most recent call last):
...
coalib.results.LineDiff.ConflictError: ...
```



**delete**

**Returns** True if file is set to be deleted.

**delete\_line** (*line\_nr*)

Mark the given line nr as deleted. The first line is line number 1.

**delete\_lines** (*line\_nr\_start*, *line\_nr\_end*)

Delete lines in a specified range, inclusively.

**classmethod from\_clang\_fixit** (*fixit*, *file*)

Creates a Diff object from a given clang fixit and the file contents.

**Parameters**

- **fixit** – A cindex.Fixit object.
- **file** – A list of lines in the file to apply the fixit to.

**Returns** The corresponding Diff object.

**classmethod from\_string\_arrays** (*file\_array\_1*, *file\_array\_2*, *rename=False*)

Creates a Diff object from two arrays containing strings.

If this Diff is applied to the original array, the second array will be created.

**Parameters**

- **file\_array\_1** – Original array
- **file\_array\_2** – Array to compare
- **rename** – False or str containing new name of file.

**insert** (*position*, *text*)

Inserts (multiline) text at arbitrary position.

```
>>> from coalib.results.TextPosition import TextPosition
>>> test_text = ['123\n', '456\n', '789\n']
>>> def insert(position, text):
...     diff = Diff(test_text)
...     diff.insert(position, text)
...     return diff.modified
>>> insert(TextPosition(2, 3), 'woopy doopy')
['123\n', '45woopy doopy6\n', '789\n']
>>> insert(TextPosition(1, 1), 'woopy\ndoopy')
['woopy\n', 'doopy123\n', '456\n', '789\n']
>>> insert(TextPosition(2, 4), '\nwoopy\ndoopy\n')
['123\n', '456\n', 'woopy\n', 'doopy\n', '\n', '789\n']
```

**Parameters**

- **position** – The TextPosition where to insert text.
- **text** – The text to insert.

**modified**

Calculates the modified file, after applying the Diff to the original.

**original**

Retrieves the original file.

### **range** (*filename*)

Calculates a SourceRange spanning over the whole Diff. If something is added after the 0th line (i.e. before the first line) the first line will be included in the SourceRange.

The range of an empty diff will only affect the filename:

```
>>> range = Diff([]).range("file")
>>> range.file is None
False
>>> print(range.start.line)
None
```

**Parameters** **filename** – The filename to associate the SourceRange with.

**Returns** A SourceRange object.

### **remove** (*range*)

Removes a piece of text in a given range.

```
>>> from coalib.results.TextRange import TextRange
>>> test_text = ['nice\n', 'try\n', 'bro\n']
>>> def remove(range):
...     diff = Diff(test_text)
...     diff.remove(range)
...     return diff.modified
>>> remove(TextRange.from_values(1, 1, 1, 4))
['e\n', 'try\n', 'bro\n']
>>> remove(TextRange.from_values(1, 5, 2, 1))
['nicetry\n', 'bro\n']
>>> remove(TextRange.from_values(1, 3, 3, 2))
['niro\n']
>>> remove(TextRange.from_values(2, 1, 2, 1))
['nice\n', 'try\n', 'bro\n']
```

**Parameters** **range** – The range to delete.

### **rename**

**Returns** string containing new name of the file.

### **replace** (*range, replacement*)

Replaces a part of text. Allows to span multiple lines.

This function uses `add_lines` and `delete_lines` accordingly, so calls of those functions on lines given range affects after usage or vice versa lead to `ConflictError`.

```
>>> from coalib.results.TextRange import TextRange
>>> test_text = ['hello\n', 'world\n', '4lines\n', 'done\n']
>>> def replace(range, text):
...     diff = Diff(test_text)
...     diff.replace(range, text)
...     return diff.modified
>>> replace(TextRange.from_values(1, 5, 4, 3), '\nyeah\ncool\nno')
['hell\n', 'yeah\n', 'cool\n', 'none\n']
>>> replace(TextRange.from_values(2, 1, 3, 5), 'b')
['hello\n', 'bes\n', 'done\n']
>>> replace(TextRange.from_values(1, 6, 4, 3), '')
['hellone\n']
```

### Parameters

- **range** – The TextRange that gets replaced.
- **replacement** – The replacement string. Can be multiline.

#### **split\_diff** (*distance=1*)

Splits this diff into small pieces, such that several continuously altered lines are still together in one diff. All subdiffs will be yielded.

A diff like this with changes being together closely won't be splitted:

```
>>> diff = Diff.from_string_arrays(['b', 'c', 'e'],
...                               ['a', 'b', 'd', 'f'])
>>> len(list(diff.split_diff()))
1
```

If we set the distance to 0, it will be splitted:

```
>>> len(list(diff.split_diff(distance=0)))
2
```

If a negative distance is given, every change will be yielded as an own diff, even if they are right beneath each other:

```
>>> len(list(diff.split_diff(distance=-1)))
3
```

If a file gets renamed or deleted only, it will be yielded as is:

```
>>> len(list(Diff([], rename='test').split_diff()))
1
```

An empty diff will not yield any diffs:

```
>>> len(list(Diff([]).split_diff()))
0
```

**Parameters distance** – Number of unchanged lines that are allowed in between two changed lines so they get yielded as one diff.

#### **stats** ()

Returns tuple containing number of additions and deletions in the diff.

#### **unified\_diff**

Generates a unified diff corresponding to this patch.

Note that the unified diff is not deterministic and thus not suitable for equality comparison.

## coala.results.HiddenResult module

**class** coalib.results.HiddenResult.**HiddenResult** (*origin, contents*)

Bases: *coala.results.Result.Result*

This is a result that is not meant to be shown to the user. It can be used to transfer any data from a dependent bear to others.

## coala.results.LineDiff module

**exception** `coala.results.LineDiff.ConflictError`

Bases: `Exception`

**class** `coala.results.LineDiff.LineDiff` (*change=False, delete=False, add\_after=False*)

Bases: `object`

A LineDiff holds the difference between two strings.

**add\_after**

**change**

**delete**

## coala.results.RESULT\_SEVERITY module

### coala.results.Result module

**class** `coala.results.Result.Result` (*origin, message: str, affected\_code: (<class 'tuple'>, <class 'list'>) = (), severity: int = 1, additional\_info: str = '', debug\_msg='', diffs: (<class 'dict'>, None) = None, confidence: int = 100, aspect: (<class 'coala.bearlib.aspects.base.aspectbase'>, None) = None)*

Bases: `object`

A result is anything that has an origin and a message.

Optionally it might affect a file.

**apply** (*file\_dict: dict*)

Applies all contained diffs to the given file\_dict. This operation will be done in-place.

**Parameters** **file\_dict** – A dictionary containing all files with filename as key and all lines a value. Will be modified.

**classmethod** **from\_values** (*origin, message: str, file: str, line: (<class 'int'>, None) = None, column: (<class 'int'>, None) = None, end\_line: (<class 'int'>, None) = None, end\_column: (<class 'int'>, None) = None, severity: int = 1, additional\_info: str = '', debug\_msg='', diffs: (<class 'dict'>, None) = None, confidence: int = 100, aspect: (<class 'coala.bearlib.aspects.base.aspectbase'>, None) = None)*

Creates a result with only one SourceRange with the given start and end locations.

#### Parameters

- **origin** – Class name or creator object of this object.
- **message** – Message to show with this result.
- **file** – The related file.
- **line** – The first related line in the file. (First line is 1)
- **column** – The column indicating the first character. (First character is 1)
- **end\_line** – The last related line in the file.
- **end\_column** – The column indicating the last character.
- **severity** – Severity of this result.

- **additional\_info** – A long description holding additional information about the issue and/or how to fix it. You can use this like a manual entry for a category of issues.
- **debug\_msg** – A message which may help the user find out why this result was yielded.
- **diffs** – A dictionary with filenames as key and a sequence of `Diff` objects associated with them as values.
- **confidence** – A number between 0 and 100 describing the likelihood of this result being a real issue.
- **aspect** – An `Aspect` object which this result is associated to. Note that this should be a leaf of the aspect tree! (If you have a node, spend some time figuring out which of the leafs exactly your result belongs to.)

**location\_repr()**

Retrieves a string, that briefly represents the affected code of the result.

**Returns** A string containing all of the affected files separated by a comma.

**overlaps(ranges)**

Determines if the result overlaps with source ranges provided.

**Parameters** **ranges** – A list `SourceRange` objects to check for overlap.

**Returns** True if the ranges overlap with the result.

**to\_string\_dict()**

Makes a dictionary which has all keys and values as strings and contains all the data that the base `Result` has.

FIXME: diffs are not serialized ATM. FIXME: Only the first `SourceRange` of `affected_code` is serialized. If there are more, this data is currently missing.

**Returns** Dictionary with keys and values as string.

## coala.results.ResultFilter module

`coala.results.ResultFilter.basics_match(original_result, modified_result)`

Checks whether the following properties of two results match: \* origin \* message \* severity \* debug\_msg

**Parameters**

- **original\_result** – A result of the old files
- **modified\_result** – A result of the new files

**Returns** Boolean value whether or not the properties match

`coala.results.ResultFilter.ensure_files_present(original_file_dict, modified_file_dict)`

Ensures that all files are available as keys in both dicts.

**Parameters**

- **original\_file\_dict** – Dict of lists of file contents before changes
- **modified\_file\_dict** – Dict of lists of file contents after changes

**Returns** Return a dictionary of renamed files.

`coala.results.ResultFilter.filter_results(original_file_dict, modified_file_dict, original_results, modified_results)`

Filters results for such ones that are unique across file changes

#### Parameters

- **original\_file\_dict** – Dict of lists of file contents before changes
- **modified\_file\_dict** – Dict of lists of file contents after changes
- **original\_results** – List of results of the old files
- **modified\_results** – List of results of the new files

**Returns** List of results from new files that are unique from all those that existed in the old changes

`coolib.results.ResultFilter.remove_range(file_contents, source_range)`  
removes the chars covered by the sourceRange from the file

#### Parameters

- **file\_contents** – list of lines in the file
- **source\_range** – Source Range

**Returns** list of file contents without specified chars removed

`coolib.results.ResultFilter.remove_result_ranges_diffs(result_list, file_dict)`  
Calculates the diffs to all files in file\_dict that describe the removal of each respective result's affected code.

#### Parameters

- **result\_list** – list of results
- **file\_dict** – dict of file contents

**Returns** `returnvalue[result][file]` is a diff of the changes the removal of this result's affected code would cause for the file.

`coolib.results.ResultFilter.source_ranges_match(original_file_dict, diff_dict, original_result_diff_dict, modified_result_diff_dict, renamed_files)`

Checks whether the SourceRanges of two results match

#### Parameters

- **original\_file\_dict** – Dict of lists of file contents before changes
- **diff\_dict** – Dict of diffs describing the changes per file
- **original\_result\_diff\_dict** – diff for each file for this result
- **modified\_result\_diff\_dict** – guess
- **renamed\_files** – A dictionary containing file renamings across runs

**Returns** Boolean value whether the SourceRanges match

### coolib.results.SourcePosition module

**class** `coolib.results.SourcePosition.SourcePosition` (*file: str, line=None, column=None*)  
Bases: `coolib.results.TextPosition.TextPosition`  
**file**

## coala.results.SourceRange module

```
class coala.results.SourceRange.SourceRange (start: coala.results.SourcePosition.SourcePosition,
                                              end:                                     (<class
                                              'coala.results.SourcePosition.SourcePosition'>,
                                              None) = None)
```

Bases: *coala.results.TextRange.TextRange*

**expand** (*file\_contents*)

Passes a new SourceRange that covers the same area of a file as this one would. All values of None get replaced with absolute values.

values of None will be interpreted as follows: self.start.line is None: -> 1 self.start.column is None: -> 1 self.end.line is None: -> last line of file self.end.column is None: -> last column of self.end.line

**Parameters** *file\_contents* – File contents of the applicable file

**Returns** TextRange with absolute values

**file**

```
classmethod from_absolute_position (file:          str,          position_start:
                                   coala.results.AbsolutePosition.AbsolutePosition,
                                   position_end:      (<class
                                   'coala.results.AbsolutePosition.AbsolutePosition'>,
                                   None) = None)
```

Creates a SourceRange from a start and end positions.

**Parameters**

- **file** – Name of the file.
- **position\_start** – Start of range given by AbsolutePosition.
- **position\_end** – End of range given by AbsolutePosition or None.

```
classmethod from_clang_range (range)
```

Creates a SourceRange from a clang SourceRange object.

**Parameters** *range* – A cindex.SourceRange object.

```
classmethod from_values (file,    start_line=None,    start_column=None,    end_line=None,
                        end_column=None)
```

```
renamed_file (file_diff_dict: dict)
```

Retrieves the filename this source range refers to while taking the possible file renamings in the given file\_diff\_dict into account:

**Parameters** *file\_diff\_dict* – A dictionary with filenames as key and their associated Diff objects as values.

## coala.results.TextPosition module

```
class coala.results.TextPosition.TextPosition (line: (<class 'int'>, None) = None, column:
                                              (<class 'int'>, None) = None)
```

Bases: object

**column**

**line**

## coala.results.TextRange module

```
class coalib.results.TextRange.TextRange (start: coalib.results.TextPosition.TextPosition, end:
                                         (<class 'coala.results.TextPosition.TextPosition'>,
                                          None) = None)
```

Bases: object

**end**

**expand** (*text\_lines*)

Passes a new TextRange that covers the same area of a file as this one would. All values of None get replaced with absolute values.

values of None will be interpreted as follows: self.start.line is None: -> 1 self.start.column is None: -> 1 self.end.line is None: -> last line of file self.end.column is None: -> last column of self.end.line

**Parameters** *text\_lines* – File contents of the applicable file

**Returns** TextRange with absolute values

```
classmethod from_values (start_line=None, start_column=None, end_line=None,
                        end_column=None)
```

Creates a new TextRange.

**Parameters**

- **start\_line** – The line number of the start position. The first line is 1.
- **start\_column** – The column number of the start position. The first column is 1.
- **end\_line** – The line number of the end position. If this parameter is None, then the end position is set the same like start position and end\_column gets ignored.
- **end\_column** – The column number of the end position.

**Returns** A TextRange.

```
classmethod join (a, b)
```

Creates a new TextRange that covers the area of two overlapping ones

**Parameters**

- **a** – TextRange (needs to overlap b)
- **b** – TextRange (needs to overlap a)

**Returns** A new TextRange covering the union of the Area of a and b

**overlaps** (*other*)

**start**

## Module contents

### 1.1.9 coalib.settings package

#### Submodules

#### coala.settings.ConfigurationGathering module

```
coala.settings.ConfigurationGathering.find_user_config (file_path, max_trials=10)
```

Uses the filepath to find the most suitable user config file for the file by going down one directory at a time and finding config files there.



### Parameters

- **file\_path** – The path of the file whose user config needs to be found
- **max\_trials** – The maximum number of directories to go down to.

**Returns** The config file's path, empty string if none was found

```
coallib.settings.ConfigurationGathering.gather_configuration(acquire_settings,
                                                            log_printer,
                                                            arg_list=None,
                                                            arg_parser=None)
```

Loads all configuration files, retrieves bears and all needed settings, saves back if needed and warns about non-existent targets.

This function:

- Reads and merges all settings in sections from
  - Default config
  - User config
  - Configuration file
  - CLI
- Collects all the bears
- Fills up all needed settings
- Writes back the new sections to the configuration file if needed
- Gives all information back to caller

### Parameters

- **acquire\_settings** – The method to use for requesting settings. It will get a parameter which is a dictionary with the settings name as key and a list containing a description in [0] and the names of the bears who need this setting in all following indexes.
- **log\_printer** – The log printer to use for logging. The log level will be adjusted to the one given by the section.
- **arg\_list** – CLI args to use
- **arg\_parser** – Instance of ArgParser that is used to parse none-setting arguments.

### Returns

A tuple with the following contents:

- A dictionary with the sections
- Dictionary of list of local bears for each section
- Dictionary of list of global bears for each section
- The targets list

```
coallib.settings.ConfigurationGathering.get_config_directory(section)
```

Retrieves the configuration directory for the given section.

Given an empty section:

```
>>> section = Section("name")
```

The configuration directory is not defined and will therefore fallback to the current directory:

```
>>> get_config_directory(section) == os.path.abspath(".")
True
```

If the `files` setting is given with an originating coafile, the directory of the coafile will be assumed the configuration directory:

```
>>> section.append(Setting("files", "**", origin="/tmp/.coafile"))
>>> get_config_directory(section) == os.path.abspath('/tmp/')
True
```

However if its origin is already a directory this will be preserved:

```
>>> section['files'].origin = os.path.abspath('/tmp/dir/')
>>> os.makedirs(section['files'].origin, exist_ok=True)
>>> get_config_directory(section) == section['files'].origin
True
```

The user can manually set a project directory with the `project_dir` setting:

```
>>> section.append(Setting('project_dir', os.path.abspath('/tmp/'), '/'))
>>> get_config_directory(section) == os.path.abspath('/tmp')
True
```

If no section is given, the current directory is returned:

```
>>> get_config_directory(None) == os.path.abspath(".")
True
```

To summarize, the config directory will be chosen by the following priorities if possible in that order:

- the `project_dir` setting
- the origin of the `files` setting, if it's a directory
- the directory of the origin of the `files` setting
- the current directory

**Parameters** `section` – The section to inspect.

**Returns** The directory where the project is lying.

`coalaib.settings.ConfigurationGathering.get_filtered_bears` (*languages, log\_printer, arg\_parser=None*)

Fetch bears and filter them based on given list of languages.

**Parameters**

- **languages** – List of languages.
- **log\_printer** – The log\_printer to handle logging.
- **arg\_parser** – An ArgParser object.

**Returns** Tuple containing dictionaries of local bears and global bears.

`coalaib.settings.ConfigurationGathering.load_config_file` (*filename, log\_printer, silent=False*)

Loads sections from a config file. Prints an appropriate warning if it doesn't exist and returns a section dict containing an empty default section in that case.

It assumes that the cli\_sections are available.

#### Parameters

- **filename** – The file to load settings from.
- **log\_printer** – The log printer to log the warning/error to (in case).
- **silent** – Whether or not to warn the user/exit if the file doesn't exist.

**Raises** **SystemExit** – Exits when the given filename is invalid and is not the default coafile. Only raised when **silent** is **False**.

`coala.lib.settings.ConfigurationGathering.load_configuration` (*arg\_list*, *log\_printer*,  
*arg\_parser=None*)

Parses the CLI args and loads the config file accordingly, taking default\_coafile and the users .coarc into account.

#### Parameters

- **arg\_list** – The list of command line arguments.
- **log\_printer** – The LogPrinter object for logging.

**Returns** A tuple holding (log\_printer: LogPrinter, sections: dict(str, Section), targets: list(str)).  
(Types indicated after colon.)

`coala.lib.settings.ConfigurationGathering.merge_section_dicts` (*lower*, *higher*)

Merges the section dictionaries. The values of higher will take precedence over the ones of lower. Lower will hold the modified dict in the end.

#### Parameters

- **lower** – A section.
- **higher** – A section which values will take precedence over the ones from the other.

**Returns** The merged dict.

`coala.lib.settings.ConfigurationGathering.save_sections` (*sections*)

Saves the given sections if they are to be saved.

**Parameters** **sections** – A section dict.

`coala.lib.settings.ConfigurationGathering.warn_config_absent` (*sections*, *argument*,  
*log\_printer*)

Checks if the given argument is present somewhere in the sections and emits a warning that code analysis can not be run without it.

#### Parameters

- **sections** – A dictionary of sections.
- **argument** – The argument to check for, e.g. "files".
- **log\_printer** – A log printer to emit the warning to.

`coala.lib.settings.ConfigurationGathering.warn_nonexistent_targets` (*targets*,  
*sections*,  
*log\_printer*)

Prints out a warning on the given log printer for all targets that are not existent within the given sections.

#### Parameters

- **targets** – The targets to check.
- **sections** – The sections to search. (Dict.)
- **log\_printer** – The log printer to warn to.

## coala.settings.DocstringMetadata module

**class** coala.settings.DocstringMetadata.**DocstringMetadata** (*desc*, *param\_dict*, *ret\_val\_desc*)

Bases: object

**classmethod** **from\_docstring** (*docstring*)

Parses a python docstring. Usable attributes are: :param @param :return @return

## coala.settings.FunctionMetadata module

**class** coala.settings.FunctionMetadata.**FunctionMetadata** (*name*: *str*, *desc*: *str* = '', *retval\_desc*: *str* = '', *non\_optional\_params*: (<class 'dict'>, None) = None, *optional\_params*: (<class 'dict'>, None) = None, *omit*: (<class 'set'>, <class 'tuple'>, <class 'list'>, <class 'frozenset'>) = frozenset(), *deprecated\_params*: (<class 'set'>, <class 'tuple'>, <class 'list'>, <class 'frozenset'>) = frozenset())

Bases: object

**add\_deprecated\_param** (*original*, *alias*)

Adds an alias for the original setting. The alias setting will have the same metadata as the original one. If the original setting is not optional, the alias will default to None.

### Parameters

- **original** – The name of the original setting.
- **alias** – The name of the alias for the original.

**Raises** **KeyError** – If the new setting doesn't exist in the metadata.

**create\_params\_from\_section** (*section*)

Create a params dictionary for this function that holds all values the function needs plus optional ones that are available.

**Parameters** **section** – The section to retrieve the values from.

**Returns** The params dictionary.

**desc**

Returns description of the function.

**filter\_parameters** (*dct*)

Filters the given dict for keys that are declared as parameters inside this metadata (either optional or non-optional).

You can use this function to safely pass parameters from a given dictionary:

```
>>> def multiply(a, b=2, c=0):
...     return a * b + c
>>> metadata = FunctionMetadata.from_function(multiply)
>>> args = metadata.filter_parameters({'a': 10, 'b': 20, 'd': 30})
```

You can safely pass the arguments to the function now:

```
>>> multiply(**args) # 10 * 20
200
```

**Parameters** `dct` – The dict to filter.

**Returns** A new dict containing the filtered items.

**classmethod** `from_function` (*func*, *omit=frozenset()*)

Creates a FunctionMetadata object from a function. Please note that any variable argument lists are not supported. If you do not want the first (usual named ‘self’) argument to appear please pass the method of an actual INSTANCE of a class; passing the method of the class isn’t enough. Alternatively you can add “self” to the omit set.

**Parameters**

- **func** – The function. If `__metadata__` of the unbound function is present it will be copied and used, otherwise it will be generated.
- **omit** – A set of parameter names that are to be ignored.

**Returns** The FunctionMetadata object corresponding to the given function.

**classmethod** `merge` (\**metadatas*)

Merges signatures of FunctionMetadata objects.

Parameter (either optional or non-optional) and non-parameter descriptions are merged from left to right, meaning the right hand metadata overrides the left hand one.

```
>>> def a(x, y):
...     '''
...     desc of *a*
...     :param x: x of a
...     :param y: y of a
...     :return: 5*x*y
...     '''
...     return 5 * x * y
>>> def b(x):
...     '''
...     desc of *b*
...     :param x: x of b
...     :return: 100*x
...     '''
...     return 100 * x
>>> metadata1 = FunctionMetadata.from_function(a)
>>> metadata2 = FunctionMetadata.from_function(b)
>>> merged = FunctionMetadata.merge(metadata1, metadata2)
>>> merged.name
"<Merged signature of 'a', 'b'>"
>>> merged.desc
'desc of *b*'
>>> merged.retval_desc
'100*x'
>>> merged.non_optional_params['x'][0]
'x of b'
>>> merged.non_optional_params['y'][0]
'y of a'
```

**Parameters** `metadatas` – The sequence of metadatas to merge.

**Returns** A `FunctionMetadata` object containing the merged signature of all given metadatas.

**`non_optional_params`**

Retrieves a dict containing the name of non optional parameters as the key and a tuple of a description and the python annotation. Values that are present in `self.omit` will be omitted.

**`optional_params`**

Retrieves a dict containing the name of optional parameters as the key and a tuple of a description, the python annotation and the default value. Values that are present in `self.omit` will be omitted.

**`str_nodesc`** = ‘No description given.’

**`str_optional`** = “Optional, defaults to ‘{}’.”

## coala.settings.Section module

**class** `coala.settings.Section.Section` (*name*, *defaults=None*)

Bases: `object`

This class holds a set of settings.

**`add_or_create_setting`** (*setting*, *custom\_key=None*, *allow\_appending=True*)

Adds the value of the setting to an existing setting if there is already a setting with the key. Otherwise creates a new setting.

**`append`** (*setting*, *custom\_key=None*)

**`bear_dirs`** ()

**`copy`** ()

**Returns** a deep copy of this object

**`delete_setting`** (*key*)

Delete a setting :param key: The key of the setting to be deleted

**`get`** (*key*, *default=''*, *ignore\_defaults=False*)

Retrieves the item without raising an exception. If the item is not available an appropriate Setting will be generated from your provided default value.

**Parameters**

- **`key`** – The key of the setting to return.
- **`default`** – The default value
- **`ignore_defaults`** – Whether or not to ignore the default section.

**Returns** The setting.

**`is_enabled`** (*targets*)

Checks if this section is enabled or, if targets is not empty, if it is included in the targets list.

**Parameters** `targets` – List of target section names, all lower case.

**Returns** True or False

**`update`** (*other\_section*, *ignore\_defaults=False*)

Incorporates all keys and values from the other section into this one. Values from the other section override the ones from this one.

Default values from the other section override the default values from this only.

#### Parameters

- **other\_section** – Another Section
- **ignore\_defaults** – If set to true, do not take default values from other

**Returns** self

**update\_setting** (*key*, *new\_key=None*, *new\_value=None*)

Updates a setting with new values. :param key: The old key string. :param new\_key: The new key string.  
:param new\_value: The new value for the setting

`coalaib.settings.Section.append_to_sections` (*sections*, *key*, *value*, *origin*, *section\_name=None*, *from\_cli=False*)

Appends the given data as a Setting to a Section with the given name. If the Section does not exist before it will be created empty.

#### Parameters

- **sections** – The sections dictionary to add to.
- **key** – The key of the setting to add.
- **value** – The value of the setting to add.
- **origin** – The origin value of the setting to add.
- **section\_name** – The name of the section to add to.
- **from\_cli** – Whether or not this data comes from the CLI.

### coalaib.settings.SectionFilling module

`coalaib.settings.SectionFilling.fill_section` (*section*, *acquire\_settings*, *log\_printer*, *bears*)

Retrieves needed settings from given bears and asks the user for missing values.

If a setting is requested by several bears, the help text from the latest bear will be taken.

#### Parameters

- **section** – A section containing available settings. Settings will be added if some are missing.
- **acquire\_settings** – The method to use for requesting settings. It will get a parameter which is a dictionary with the settings name as key and a list containing a description in [0] and the names of the bears who need this setting in all following indexes.
- **log\_printer** – The log printer for logging.
- **bears** – All bear classes or instances.

**Returns** The new section.

`coalaib.settings.SectionFilling.fill_settings` (*sections*, *acquire\_settings*, *log\_printer*)

Retrieves all bears and requests missing settings via the given `acquire_settings` method.

This will retrieve all bears and their dependencies.

#### Parameters

- **sections** – The sections to fill up, modified in place.

- **acquire\_settings** – The method to use for requesting settings. It will get a parameter which is a dictionary with the settings name as key and a list containing a description in [0] and the names of the bears who need this setting in all following indexes.
- **log\_printer** – The log printer to use for logging.

**Returns** A tuple containing (local\_bears, global\_bears), each of them being a dictionary with the section name as key and as value the bears as a list.

## coolib.settings.Setting module

```
class coalib.settings.Setting.Setting(key, value, origin='', strip_whitespace=True,
                                       list_delimiters=(' ', ';'), from_cli=False, re-
                                       move_empty_iter_elements=True)
```

Bases: coalib\_utils.string\_processing.StringConverter.StringConverter

A Setting consists mainly of a key and a value. It mainly offers many conversions into common data types.

### key

```
coolib.settings.Setting.glob(obj, *args, **kwargs)
```

Creates a path in which all special glob characters in all the parent directories in the given setting are properly escaped.

**Parameters** **obj** – The Setting object from which the key is obtained.

**Returns** Returns a path in which special glob characters are escaped.

```
coolib.settings.Setting.glob_list(obj, *args, **kwargs)
```

Creates a list of paths in which all special glob characters in all the parent directories of all paths in the given setting are properly escaped.

**Parameters** **obj** – The Setting object from which the key is obtained.

**Returns** Returns a list of paths in which special glob characters are escaped.

```
coolib.settings.Setting.path(obj, *args, **kwargs)
```

```
coolib.settings.Setting.path_list(obj, *args, **kwargs)
```

```
coolib.settings.Setting.typed_dict(key_type, value_type, default)
```

Creates a function that converts a setting into a dict with the given types.

### Parameters

- **key\_type** – The type conversion function for the keys.
- **value\_type** – The type conversion function for the values.
- **default** – The default value to use if no one is given by the user.

**Returns** A conversion function.

```
coolib.settings.Setting.typed_list(conversion_func)
```

Creates a function that converts a setting into a list of elements each converted with the given conversion function.

**Parameters** **conversion\_func** – The conversion function that converts a string into your desired list item object.

**Returns** A conversion function.

```
coolib.settings.Setting.typed_ordered_dict(key_type, value_type, default)
```

Creates a function that converts a setting into an ordered dict with the given types.



### Parameters

- **key\_type** – The type conversion function for the keys.
- **value\_type** – The type conversion function for the values.
- **default** – The default value to use if no one is given by the user.

**Returns** A conversion function.

`coolib.settings.Setting.url(obj, *args, **kwargs)`

## Module contents

### 1.1.10 coalib.testing package

#### Submodules

#### coolib.testing.BearTestHelper module

`coolib.testing.BearTestHelper.generate_skip_decorator(bear)`

Creates a skip decorator for a *unittest* module test from a bear.

*check\_prerequisites* is used to determine a test skip.

**Parameters** **bear** – The bear whose prerequisites determine the test skip.

**Returns** A decorator that skips the test if appropriate.

#### coolib.testing.LocalBearTestHelper module

**class** `coolib.testing.LocalBearTestHelper.LocalBearTestHelper(methodName='runTest')`  
**Bases:** `unittest.case.TestCase`

This is a helper class for simplification of testing of local bears.

Please note that all abstraction will prepare the lines so you don't need to do that if you use them.

If you miss some methods, get in contact with us, we'll be happy to help!

**check\_results**(*local\_bear*, *lines*, *results*, *filename=None*, *check\_order=False*,  
*force\_linebreaks=True*, *create\_tempfile=True*, *tempfile\_kwargs={}*, *settings={}*)

Asserts that a check of the given lines with the given local bear does yield exactly the given results.

### Parameters

- **local\_bear** – The local bear to check with.
- **lines** – The lines to check. (List of strings)
- **results** – The expected list of results.
- **filename** – The filename, if it matters.
- **force\_linebreaks** – Whether to append newlines at each line if needed. (Bears expect a `\n` for every line)
- **create\_tempfile** – Whether to save lines in tempfile if needed.
- **tempfile\_kwargs** – Kwargs passed to `tempfile.mkstemp()`.
- **settings** – A dictionary of keys and values (both strings) from which settings will be created that will be made available for the tested bear.

**check\_validity**(*local\_bear*, *lines*, *filename=None*, *valid=True*, *force\_linebreaks=True*, *create\_tempfile=True*, *tempfile\_kwargs={}*)

Asserts that a check of the given lines with the given local bear either yields or does not yield any results.

#### Parameters

- **local\_bear** – The local bear to check with.
- **lines** – The lines to check. (List of strings)
- **filename** – The filename, if it matters.
- **valid** – Whether the lines are valid or not.
- **force\_linebreaks** – Whether to append newlines at each line if needed. (Bears expect a `\n` for every line)
- **create\_tempfile** – Whether to save lines in tempfile if needed.
- **tempfile\_kwargs** – Kwargs passed to `tempfile.mkstemp()`.

`coalaib.testing.LocalBearTestHelper.execute_bear(bear, *args, **kwargs)`

`coalaib.testing.LocalBearTestHelper.verify_local_bear(bear, valid_files, invalid_files, filename=None, settings={}, force_linebreaks=True, create_tempfile=True, timeout=None, tempfile_kwargs={})`

Generates a test for a local bear by checking the given valid and invalid file contents. Simply use it on your module level like:

**YourTestName** = `verify_local_bear(YourBear, ([‘valid line’]), ([‘invalid line’]),)`

#### Parameters

- **bear** – The Bear class to test.
- **valid\_files** – An iterable of files as a string list that won’t yield results.
- **invalid\_files** – An iterable of files as a string list that must yield results.
- **filename** – The filename to use for valid and invalid files.
- **settings** – A dictionary of keys and values (both string) from which settings will be created that will be made available for the tested bear.
- **force\_linebreaks** – Whether to append newlines at each line if needed. (Bears expect a `\n` for every line)
- **create\_tempfile** – Whether to save lines in tempfile if needed.
- **timeout** – The total time to run the test for.
- **tempfile\_kwargs** – Kwargs passed to `tempfile.mkstemp()` if tempfile needs to be created.

**Returns** A `unittest.TestCase` object.

## Module contents

## 1.2 Submodules

### 1.3 coalib.coala module

```
coolib.coala.main()
```

### 1.4 coalib.coala\_ci module

```
coolib.coala_ci.main()
```

### 1.5 coalib.coala\_delete\_orig module

```
coolib.coala_delete_orig.main(log_printer=None, section: coalib.settings.Section.Section =  
                             None)
```

### 1.6 coalib.coala\_format module

```
coolib.coala_format.main()
```

### 1.7 coalib.coala\_json module

```
coolib.coala_json.main()
```

### 1.8 coalib.coala\_main module

```
coolib.coala_main.do_nothing(*args)
```

```
coolib.coala_main.run_coala(console_printer=None, log_printer=None, print_results=<function  
                             <lambda>>, acquire_settings=<function fail_acquire_settings>,  
                             print_section_beginning=<function <lambda>>, nothing_  
                             ing_done=<function <lambda>>, autoapply=True,  
                             force_show_patch=False, arg_parser=None, arg_list=None)
```

This is a main method that should be usable for almost all purposes and reduces executing coala to one function call.

#### Parameters

- **console\_printer** – Object to print messages on the console.
- **log\_printer** – A LogPrinter object to use for logging.
- **print\_results** – A callback that takes a LogPrinter, a section, a list of results to be printed, the file dict and the mutable file diff dict.

- **acquire\_settings** – The method to use for requesting settings. It will get a parameter which is a dictionary with the settings name as key and a list containing a description in [0] and the names of the bears who need this setting in all following indexes.
- **print\_section\_beginning** – A callback that will be called with a section name string whenever analysis of a new section is started.
- **nothing\_done** – A callback that will be called with only a log printer that shall indicate that nothing was done.
- **autoapply** – Set this to false to not autoapply any actions. If you set this to *False*, *force\_show\_patch* will be ignored.
- **force\_show\_patch** – If set to True, a patch will be always shown. (Using Apply-PatchAction.)
- **arg\_list** – The CLI argument list.

**Returns** A dictionary containing a list of results for all analyzed sections as key.

## 1.9 coalib.coala\_modes module

```
coolib.coala_modes.mode_format()
coolib.coala_modes.mode_json(args)
coolib.coala_modes.mode_non_interactive(console_printer, args)
coolib.coala_modes.mode_normal(console_printer, log_printer)
```

## 1.10 Module contents

The coalib package is a collection of various subpackages regarding writing, executing and editing bears. Various other packages such as formatting and settings are also included in coalib.

```
coolib.assert_supported_version()
coolib.get_version()
```

## C

- coailib, 102
- coailib.bearlib, 38
- coailib.bearlib.abstractions, 7
- coailib.bearlib.abstractions.ExternalBearWrap, 3
- coailib.bearlib.abstractions.Linter, 3
- coailib.bearlib.abstractions.SectionCreatable, 6
- coailib.bearlib.aspects, 19
- coailib.bearlib.aspects.base, 18
- coailib.bearlib.aspects.docs, 18
- coailib.bearlib.aspects.meta, 19
- coailib.bearlib.aspects.Metadata, 7
- coailib.bearlib.aspects.Redundancy, 14
- coailib.bearlib.aspects.taste, 19
- coailib.bearlib.languages, 36
- coailib.bearlib.languages.definitions, 26
- coailib.bearlib.languages.definitions.C, 26
- coailib.bearlib.languages.definitions.CPP, 26
- coailib.bearlib.languages.definitions.CSharp, 26
- coailib.bearlib.languages.definitions.CSS, 26
- coailib.bearlib.languages.definitions.Java, 26
- coailib.bearlib.languages.definitions.JavaScript, 26
- coailib.bearlib.languages.definitions.Python, 26
- coailib.bearlib.languages.definitions.Unknown, 26
- coailib.bearlib.languages.definitions.Vala, 26
- coailib.bearlib.languages.documentation, 31
- coailib.bearlib.languages.documentation.DocstyleDefn, 27
- coailib.bearlib.languages.documentation.Documentation, 29
- coailib.bearlib.languages.documentation.DocumentationPage, 30
- coailib.bearlib.languages.Language, 31
- coailib.bearlib.languages.LanguageDefinition, 35
- coailib.bearlib.naming\_conventions, 36
- coailib.bearlib.spacing, 38
- coailib.bearlib.spacing.SpacingHelper, 38
- coailib.bears, 47
- coailib.bears.Bear, 42
- coailib.bears.BEAR\_KIND, 42
- coailib.bears.GlobalBear, 46
- coailib.bears.LocalBear, 46
- coailib.bears.requirements, 42
- coailib.bears.requirements.DistributionRequirement, 40
- coailib.bears.requirements.GemRequirement, 40
- coailib.bears.requirements.GoRequirement, 40
- coailib.bears.requirements.JuliaRequirement, 41
- coailib.bears.requirements.NpmRequirement, 41
- coailib.bears.requirements.PackageRequirement, 41
- coailib.bears.requirements.PipRequirement, 42
- coailib.bears.requirements.RscriptRequirement, 42
- coailib.coala, 101
- coailib.coala\_ci, 101
- coailib.coala\_delete\_orig, 101
- coailib.coala\_format, 101
- coailib.coala\_json, 101
- coailib.coala\_main, 101

- coala.lib.coala\_modes, 102
- coala.lib.collecting, 50
- coala.lib.collecting.Collectors, 47
- coala.lib.collecting.Dependencies, 49
- coala.lib.collecting.Importers, 49
- coala.lib.misc, 58
- coala.lib.misc.Annotations, 50
- coala.lib.misc.BuildManPage, 51
- coala.lib.misc.Caching, 51
- coala.lib.misc.CachingUtilities, 53
- coala.lib.misc.Compatibility, 55
- coala.lib.misc.Constants, 55
- coala.lib.misc.ContextManagers, 55
- coala.lib.misc.DictUtilities, 56
- coala.lib.misc.Enum, 56
- coala.lib.misc.Exceptions, 56
- coala.lib.misc.MutableValue, 56
- coala.lib.misc.Shell, 57
- coala.lib.output, 66
- coala.lib.output.ConfWriter, 59
- coala.lib.output.ConsoleInteraction, 60
- coala.lib.output.Interactions, 65
- coala.lib.output.JSONEncoder, 65
- coala.lib.output.printers, 59
- coala.lib.output.printers.ListLogPrinter, 58
- coala.lib.output.printers.LOG\_LEVEL, 58
- coala.lib.output.printers.LogPrinter, 58
- coala.lib.parsing, 69
- coala.lib.parsing.CliParsing, 66
- coala.lib.parsing.ConfParser, 67
- coala.lib.parsing.DefaultArgParser, 67
- coala.lib.parsing.Globbing, 67
- coala.lib.parsing.LineParser, 69
- coala.lib.processes, 78
- coala.lib.processes.BearRunning, 70
- coala.lib.processes.communication, 70
- coala.lib.processes.communication.LogMessage, 69
- coala.lib.processes.CONTROL\_ELEMENT, 74
- coala.lib.processes.LogPrinterThread, 74
- coala.lib.processes.Processing, 74
- coala.lib.results, 90
- coala.lib.results.AbsolutePosition, 81
- coala.lib.results.Diff, 82
- coala.lib.results.HiddenResult, 85
- coala.lib.results.LineDiff, 86
- coala.lib.results.Result, 86
- coala.lib.results.result\_actions, 81
- coala.lib.results.result\_actions.ApplyPatchAction, 78
- coala.lib.results.result\_actions.IgnoreResultAction, 79
- coala.lib.results.result\_actions.OpenEditorAction, 79
- coala.lib.results.result\_actions.PrintDebugMessageAction, 79
- coala.lib.results.result\_actions.PrintMoreInfoAction, 79
- coala.lib.results.result\_actions.ResultAction, 80
- coala.lib.results.result\_actions.ShowPatchAction, 80
- coala.lib.results.RESULT\_SEVERITY, 86
- coala.lib.results.ResultFilter, 87
- coala.lib.results.SourcePosition, 88
- coala.lib.results.SourceRange, 89
- coala.lib.results.TextPosition, 89
- coala.lib.results.TextRange, 90
- coala.lib.settings, 99
- coala.lib.settings.ConfigurationGathering, 90
- coala.lib.settings.DocstringMetadata, 94
- coala.lib.settings.FunctionMetadata, 94
- coala.lib.settings.Section, 96
- coala.lib.settings.SectionFilling, 97
- coala.lib.settings.Setting, 98
- coala.lib.testing, 101
- coala.lib.testing.BearTestHelper, 99
- coala.lib.testing.LocalBearTestHelper, 99

## A

AbsolutePosition (class in coalib.results.AbsolutePosition), 81  
 acquire\_actions\_and\_apply() (in module coalib.output.ConsoleInteraction), 60  
 acquire\_settings() (in module coalib.output.ConsoleInteraction), 60  
 add\_after (coalib.results.LineDiff.LineDiff attribute), 86  
 add\_deprecated\_param() (coalib.settings.FunctionMetadata.FunctionMetadata method), 94  
 add\_lines() (coalib.results.Diff.Diff method), 82  
 add\_or\_create\_setting() (coalib.settings.Section.Section method), 96  
 add\_pair\_to\_dict() (in module coalib.misc.DictUtilities), 56  
 affected\_code() (coalib.results.Diff.Diff method), 82  
 all (coalib.bearlib.languages.Language.LanguageUberMeta attribute), 34  
 append() (coalib.settings.Section.Section method), 96  
 append\_to\_sections() (in module coalib.settings.Section), 97  
 apply() (coalib.results.Result.Result method), 86  
 apply() (coalib.results.result\_actions.ApplyPatchAction.ApplyPatchAction method), 78  
 apply() (coalib.results.result\_actions.IgnoreResultAction.IgnoreResultAction method), 79  
 apply() (coalib.results.result\_actions.OpenEditorAction.OpenEditorAction method), 79  
 apply() (coalib.results.result\_actions.PrintDebugMessageAction.PrintDebugMessageAction method), 79  
 apply() (coalib.results.result\_actions.PrintMoreInfoAction.PrintMoreInfoAction method), 79  
 apply() (coalib.results.result\_actions.ResultAction.ResultAction method), 80  
 apply() (coalib.results.result\_actions.ShowPatchAction.ShowPatchAction method), 80  
 apply\_from\_section() (coalib.results.result\_actions.ResultAction.ResultAction method), 80  
 ApplyPatchAction (class in coalib.results.result\_actions.ApplyPatchAction), 78  
 ASCIINEMA\_URL (coalib.bears.Bear.Bear attribute), 44  
 ask\_for\_action\_and\_apply() (in module coalib.output.ConsoleInteraction), 60  
 aspectbase (class in coalib.bearlib.aspects.base), 18  
 aspectclass (class in coalib.bearlib.aspects), 25  
 aspectclass (class in coalib.bearlib.aspects.meta), 19  
 assemble() (coalib.bearlib.languages.documentation.DocumentationComments.DocumentationComments method), 29  
 assert\_supported\_version() (in module coalib), 102  
 attributes (coalib.bearlib.languages.Language.Language attribute), 33  
 AUTHORS (coalib.bears.Bear.Bear attribute), 44  
 AUTHORS\_EMAILS (coalib.bears.Bear.Bear attribute), 44  
 autoapply\_actions() (in module coalib.processes.Processing), 74

## B

BackgroundMessageStyle (class in coalib.output.ConsoleInteraction), 60  
 BackgroundSourceRangeStyle (class in coalib.output.ConsoleInteraction), 60  
 basics\_match() (in module coalib.results.ResultFilter), 87  
 Bear (class in coalib.bears.Bear), 42  
 BEAR\_DEPS (coalib.bears.Bear.Bear attribute), 44  
 be\_editor() (coalib.settings.Section.Section method), 96  
 Body (class in coalib.bearlib.aspects.Metadata), 7  
 BodyPrintDebugMessageAction (class in coalib.bearlib.aspects.Metadata), 7  
 BodyPrintMoreInfoAction (class in coalib.bearlib.aspects.Metadata), 7  
 BuildManPage (class in coalib.misc.BuildManPage), 51

## C

can\_without\_output (in module coalib.misc.Shell), 57  
 CAN\_DETECT (coalib.bears.Bear.Bear attribute), 44  
 can\_detect (coalib.bears.Bear.Bear attribute), 44  
 CAN\_FIX (coalib.bears.Bear.Bear attribute), 44

- change (coala.results.LineDiff.LineDiff attribute), 86
- change\_directory() (in module coalib.misc.ContextManagers), 55
- change\_line() (coala.results.Diff.Diff method), 82
- check\_conflicts() (in module coalib.parsing.CliParsing), 66
- check\_consistency() (coala.bearlib.aspects.docs.DocumentationChecker method), 18
- check\_prerequisites() (coala.bears.Bear.Bear class method), 44
- check\_result\_ignore() (in module coalib.processes.Processing), 75
- check\_results() (coala.testing.LocalBearTestHelper.LocalBearTestHelper method), 99
- check\_validity() (coala.testing.LocalBearTestHelper.LocalBearTestHelper method), 99
- choose\_action() (in module coalib.output.ConsoleInteraction), 61
- CircularDependencyError, 49
- Clone (class in coalib.bearlib.aspects.Redundancy), 14
- coala (module), 102
- coala.bearlib (module), 38
- coala.bearlib.abstractions (module), 7
- coala.bearlib.abstractions.ExternalBearWrap (module), 3
- coala.bearlib.abstractions.Linter (module), 3
- coala.bearlib.abstractions.SectionCreatable (module), 6
- coala.bearlib.aspects (module), 19
- coala.bearlib.aspects.base (module), 18
- coala.bearlib.aspects.docs (module), 18
- coala.bearlib.aspects.meta (module), 19
- coala.bearlib.aspects.Metadata (module), 7
- coala.bearlib.aspects.Redundancy (module), 14
- coala.bearlib.aspects.taste (module), 19
- coala.bearlib.languages (module), 36
- coala.bearlib.languages.definitions (module), 26
- coala.bearlib.languages.definitions.C (module), 26
- coala.bearlib.languages.definitions.CPP (module), 26
- coala.bearlib.languages.definitions.CSharp (module), 26
- coala.bearlib.languages.definitions.CSS (module), 26
- coala.bearlib.languages.definitions.Java (module), 26
- coala.bearlib.languages.definitions.JavaScript (module), 26
- coala.bearlib.languages.definitions.Python (module), 26
- coala.bearlib.languages.definitions.Unknown (module), 26
- coala.bearlib.languages.definitions.Vala (module), 26
- coala.bearlib.languages.documentation (module), 31
- coala.bearlib.languages.documentation.DocstyleDefinition (module), 27
- coala.bearlib.languages.documentation.DocumentationCommand (module), 29
- coala.bearlib.languages.documentation.DocumentationExtractor (module), 30
- coala.bearlib.languages.Language (module), 31
- coala.bearlib.languages.LanguageDefinition (module), 35
- coala.bearlib.naming\_conventions (module), 36
- coala.bearlib.spacing (module), 38
- coala.bearlib.spacing.SpacingHelper (module), 38
- coala.bears (module), 47
- coala.bears.Bear (module), 42
- coala.bears.BEAR\_KIND (module), 42
- coala.bears.GlobalBear (module), 46
- coala.bears.LocalBear (module), 46
- coala.bears.requirements (module), 42
- coala.bears.requirements.DistributionRequirement (module), 40
- coala.bears.requirements.GemRequirement (module), 40
- coala.bears.requirements.GoRequirement (module), 40
- coala.bears.requirements.JuliaRequirement (module), 41
- coala.bears.requirements.NpmRequirement (module), 41
- coala.bears.requirements.PackageRequirement (module), 41
- coala.bears.requirements.PipRequirement (module), 42
- coala.bears.requirements.RscriptRequirement (module), 42
- coala.coala (module), 101
- coala.coala\_ci (module), 101
- coala.coala\_delete\_orig (module), 101
- coala.coala\_format (module), 101
- coala.coala\_json (module), 101
- coala.coala\_main (module), 101
- coala.coala\_modes (module), 102
- coala.collecting (module), 50
- coala.collecting.Collectors (module), 47
- coala.collecting.Dependencies (module), 49
- coala.collecting.Importers (module), 49
- coala.misc (module), 58
- coala.misc.Annotations (module), 50
- coala.misc.BuildManPage (module), 51
- coala.misc.Caching (module), 51
- coala.misc.CachingUtilities (module), 53
- coala.misc.Compatibility (module), 55
- coala.misc.Constants (module), 55
- coala.misc.ContextManagers (module), 55
- coala.misc.DictUtilities (module), 56
- coala.misc.Enum (module), 56
- coala.misc.Exceptions (module), 56
- coala.misc.MutableValue (module), 56
- coala.misc.Shell (module), 57
- coala.output (module), 66
- coala.output.ConfWriter (module), 59
- coala.output.ConsoleInteraction (module), 60
- coala.output.Interactions (module), 65
- coala.output.JSONEncoder (module), 65
- coala.output.printers (module), 59
- coala.output.printers.ListLogPrinter (module), 58
- coala.output.printers.LOG\_LEVEL (module), 58



- coala.output.printers.LogPrinter (module), 58
  - coala.parsing (module), 69
  - coala.parsing.CliParsing (module), 66
  - coala.parsing.ConfParser (module), 67
  - coala.parsing.DefaultArgParser (module), 67
  - coala.parsing.Globber (module), 67
  - coala.parsing.LineParser (module), 69
  - coala.processes (module), 78
  - coala.processes.BearRunning (module), 70
  - coala.processes.communication (module), 70
  - coala.processes.communication.LogMessage (module), 69
  - coala.processes.CONTROL\_ELEMENT (module), 74
  - coala.processes.LogPrinterThread (module), 74
  - coala.processes.Processing (module), 74
  - coala.results (module), 90
  - coala.results.AbsolutePosition (module), 81
  - coala.results.Diff (module), 82
  - coala.results.HiddenResult (module), 85
  - coala.results.LineDiff (module), 86
  - coala.results.Result (module), 86
  - coala.results.result\_actions (module), 81
  - coala.results.result\_actions.ApplyPatchAction (module), 78
  - coala.results.result\_actions.IgnoreResultAction (module), 79
  - coala.results.result\_actions.OpenEditorAction (module), 79
  - coala.results.result\_actions.PrintDebugMessageAction (module), 79
  - coala.results.result\_actions.PrintMoreInfoAction (module), 79
  - coala.results.result\_actions.ResultAction (module), 80
  - coala.results.result\_actions.ShowPatchAction (module), 80
  - coala.results.RESULT\_SEVERITY (module), 86
  - coala.results.ResultFilter (module), 87
  - coala.results.SourcePosition (module), 88
  - coala.results.SourceRange (module), 89
  - coala.results.TextPosition (module), 89
  - coala.results.TextRange (module), 90
  - coala.settings (module), 99
  - coala.settings.ConfigurationGathering (module), 90
  - coala.settings.DocstringMetadata (module), 94
  - coala.settings.FunctionMetadata (module), 94
  - coala.settings.Section (module), 96
  - coala.settings.SectionFilling (module), 97
  - coala.settings.Setting (module), 98
  - coala.testing (module), 101
  - coala.testing.BearTestHelper (module), 99
  - coala.testing.LocalBearTestHelper (module), 99
  - collect\_all\_bears\_from\_sections() (in module coala.collecting.Collectors), 47
  - collect\_bears() (in module coala.collecting.Collectors), 47
  - collect\_dirs() (in module coala.collecting.Collectors), 47
  - collect\_files() (in module coala.collecting.Collectors), 47
  - collect\_registered\_bears\_dirs() (in module coala.collecting.Collectors), 48
  - ColonExistence (class in coala.bearlib.aspects.Metadata), 8
  - column (coala.results.TextPosition.TextPosition attribute), 89
  - CommitMessage (class in coala.bearlib.aspects.Metadata), 8
  - CommitMessage.Body (class in coala.bearlib.aspects.Metadata), 8
  - CommitMessage.Body.Existence (class in coala.bearlib.aspects.Metadata), 8
  - CommitMessage.Body.Length (class in coala.bearlib.aspects.Metadata), 8
  - CommitMessage.Emptyiness (class in coala.bearlib.aspects.Metadata), 8
  - CommitMessage.Shortlog (class in coala.bearlib.aspects.Metadata), 9
  - CommitMessage.Shortlog.ColonExistence (class in coala.bearlib.aspects.Metadata), 9
  - CommitMessage.Shortlog.FirstCharacter (class in coala.bearlib.aspects.Metadata), 9
  - CommitMessage.Shortlog.Length (class in coala.bearlib.aspects.Metadata), 9
  - CommitMessage.Shortlog.Tense (class in coala.bearlib.aspects.Metadata), 9
  - CommitMessage.Shortlog.TrailingPeriod (class in coala.bearlib.aspects.Metadata), 9
  - configure\_logging() (in module coala.misc.Constants), 55
  - ConflictError, 86
  - ConfParser (class in coala.parsing.ConfParser), 67
  - ConfWriter (class in coala.output.ConfWriter), 59
  - copy() (coala.settings.Section.Section method), 96
  - create\_json\_encoder() (in module coala.output.JSONEncoder), 65
  - create\_params\_from\_section() (coala.settings.FunctionMetadata.FunctionMetadata method), 94
  - create\_process\_group() (in module coala.processes.Processing), 75
  - CustomFormatter (class in coala.parsing.DefaultArgParser), 67
- ## D
- data\_dir (coala.bears.Bear.Bear attribute), 44
  - debug() (coala.output.printers.LogPrinter.LogPrinterMixin method), 59
  - default\_arg\_parser() (in module coala.parsing.DefaultArgParser), 67

[illegible]

docs (coalib.bearlib.aspects.Redundancy.Redundancy.Clone docs (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Emptiness attribute), 14 attribute), 22

docs (coalib.bearlib.aspects.Redundancy.Redundancy.UnreachableCode docs (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Shortlog attribute), 15 attribute), 23

docs (coalib.bearlib.aspects.Redundancy.Redundancy.UnreachableCode docs (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Shortlog.Color attribute), 14 attribute), 22

docs (coalib.bearlib.aspects.Redundancy.Redundancy.UnreachableCode docs (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Shortlog.FirstC attribute), 15 attribute), 22

docs (coalib.bearlib.aspects.Redundancy.Redundancy.UnusedDocstring docs (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Shortlog.Length attribute), 15 attribute), 22

docs (coalib.bearlib.aspects.Redundancy.Redundancy.UnusedDocstring docs (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Shortlog.Tense attribute), 16 attribute), 23

docs (coalib.bearlib.aspects.Redundancy.Redundancy.UnusedDocstring docs (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Shortlog.Trailing attribute), 15 attribute), 23

docs (coalib.bearlib.aspects.Redundancy.Redundancy.UnusedDocstring docs (coalib.bearlib.aspects.Root.Redundancy attribute), attribute), 15 25

docs (coalib.bearlib.aspects.Redundancy.Redundancy.UnusedDocstring docs (coalib.bearlib.aspects.Root.Redundancy.Clone attribute), attribute), 15 attribute), 23

docs (coalib.bearlib.aspects.Redundancy.UnreachableCode docs (coalib.bearlib.aspects.Root.Redundancy.UnreachableCode attribute), 16 attribute), 24

docs (coalib.bearlib.aspects.Redundancy.UnreachableCode docs (coalib.bearlib.aspects.Root.Redundancy.UnreachableCode.Unreachable attribute), 16 attribute), 24

docs (coalib.bearlib.aspects.Redundancy.UnreachableCode docs (coalib.bearlib.aspects.Root.Redundancy.UnreachableCode.UnusedFunction attribute), 16 attribute), 24

docs (coalib.bearlib.aspects.Redundancy.UnreachableCode docs (coalib.bearlib.aspects.Root.Redundancy.UnusedImport attribute), 16 attribute), 24

docs (coalib.bearlib.aspects.Redundancy.UnusedFunction docs (coalib.bearlib.aspects.Root.Redundancy.UnusedVariable attribute), 16 attribute), 25

docs (coalib.bearlib.aspects.Redundancy.UnusedGlobalVariable docs (coalib.bearlib.aspects.Root.Redundancy.UnusedVariable.UnusedGlobal attribute), 17 attribute), 24

docs (coalib.bearlib.aspects.Redundancy.UnusedImport docs (coalib.bearlib.aspects.Root.Redundancy.UnusedVariable.UnusedLocal attribute), 17 attribute), 24

docs (coalib.bearlib.aspects.Redundancy.UnusedLocalVariable docs (coalib.bearlib.aspects.Root.Redundancy.UnusedVariable.UnusedParameter attribute), 17 attribute), 25

docs (coalib.bearlib.aspects.Redundancy.UnusedParameter DocstringMetadata (class in coalib.settings.DocstringMetadata), 94 attribute), 17

docs (coalib.bearlib.aspects.Redundancy.UnusedVariable docstyle (coalib.bearlib.languages.documentation.DocstyleDefinition.Docstyle attribute), 18 attribute), 27

docs (coalib.bearlib.aspects.Redundancy.UnusedVariable.UnusedGlobalVariable docstyle (coalib.bearlib.languages.documentation.DocumentationComment.Docstyle attribute), 17 attribute), 29

docs (coalib.bearlib.aspects.Redundancy.UnusedVariable.UnusedGlobalVariable DocstyleDefinition (class in coalib.bearlib.languages.documentation.DocstyleDefinition), attribute), 18 attribute), 27

docs (coalib.bearlib.aspects.Redundancy.UnusedVariable.UnusedParameter DocstyleDefinition.Metadata (class in coalib.bearlib.languages.documentation.DocstyleDefinition), attribute), 18 attribute), 27

docs (coalib.bearlib.aspects.Root.Metadata attribute), 23

docs (coalib.bearlib.aspects.Root.Metadata.CommitMessage Documentation (class in coalib.bearlib.aspects.docs), 18 attribute), 23

docs (coalib.bearlib.aspects.Root.Metadata.CommitMessage DocumentationComment (class in coalib.bearlib.languages.documentation.DocumentationComment attribute), 22

docs (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Body.Existence DocumentationComment.Description (class in coalib.bearlib.languages.documentation.DocumentationComment attribute), 21

docs (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Body.Length DocumentationComment.Length (class in coalib.bearlib.languages.documentation.DocumentationComment attribute), 22

DocumentationComment.Parameter (class in filter\_raising\_callables() (in module coalib.bearlib.languages.documentation.DocumentationComment), 76  
 29  
 DocumentationComment.ReturnValue (class in filter\_section\_bears\_by\_languages() (in module coalib.bearlib.languages.documentation.DocumentationComment), 48  
 29  
 download\_cached\_file() (coalib.bears.Bear.Bear method), 51  
 44

## E

Emptiness (class in coalib.bearlib.aspects.Metadata), 10  
 end (coalib.results.TextRange.TextRange attribute), 90  
 ensure\_files\_present() (in module coalib.results.ResultFilter), 87  
 enum() (in module coalib.misc.Enum), 56  
 err() (coalib.output.printers.LogPrinter.LogPrinterMixin method), 59  
 execute() (coalib.bears.Bear.Bear method), 45  
 execute\_bear() (in module coalib.testing.LocalBearTestHelper), 100  
 execute\_section() (in module coalib.processes.Processing), 75  
 Existence (class in coalib.bearlib.aspects.Metadata), 10  
 expand() (coalib.results.SourceRange.SourceRange method), 89  
 expand() (coalib.results.TextRange.TextRange method), 90  
 external\_bear\_wrap() (in module coalib.bearlib.abstractions.ExternalBearWrap), 3  
 extract\_documentation() (in module coalib.bearlib.languages.documentation.DocumentationComment), 30  
 extract\_documentation\_with\_markers() (in module coalib.bearlib.languages.documentation.DocumentationExtractor), 31

## F

fail\_acquire\_settings() (in module coalib.output.Interactions), 65  
 file (coalib.results.SourcePosition.SourcePosition attribute), 88  
 file (coalib.results.SourceRange.SourceRange attribute), 89  
 FileCache (class in coalib.misc.Caching), 51  
 fill\_queue() (in module coalib.processes.Processing), 75  
 fill\_section() (in module coalib.settings.SectionFilling), 97  
 fill\_settings() (in module coalib.settings.SectionFilling), 97  
 filter\_capabilities\_by\_languages() (in module coalib.collecting.Collectors), 48  
 filter\_parameters() (coalib.settings.FunctionMetadata.FunctionMetadata method), 94

FirstCharacter (class in coalib.bearlib.aspects.Metadata), 10  
 flush\_cache() (coalib.misc.Caching.FileCache method), 52  
 fnmatch() (in module coalib.parsing.Globbing), 67  
 for\_bears() (coalib.collecting.Dependencies.CircularDependencyError class method), 49  
 format\_line() (in module coalib.results.result\_actions.ShowPatchAction), 81  
 format\_lines() (in module coalib.output.ConsoleInteraction), 61  
 format\_man\_page() (coalib.misc.BuildManPage.ManPageFormatter method), 51  
 from\_absolute\_position() (coalib.results.SourceRange.SourceRange class method), 89  
 from\_clang\_fixit() (coalib.results.Diff.Diff class method), 83  
 from\_clang\_range() (coalib.results.SourceRange.SourceRange class method), 89  
 from\_docstring() (coalib.settings.DocstringMetadata.DocstringMetadata class method), 94  
 from\_function() (coalib.settings.FunctionMetadata.FunctionMetadata class method), 95  
 from\_metadata() (coalib.bearlib.languages.documentation.DocumentationComment class method), 29  
 from\_section() (coalib.bearlib.abstractions.SectionCreatable.SectionCreatable class method), 7  
 from\_string\_arrays() (coalib.results.Diff.Diff class method), 83  
 from\_values() (coalib.results.Result.Result class method), 86  
 from\_values() (coalib.results.SourceRange.SourceRange class method), 89  
 from\_values() (coalib.results.TextRange.TextRange class method), 90  
 FunctionMetadata (class in coalib.settings.FunctionMetadata), 94

## G

gather\_configuration() (in module coalib.settings.ConfigurationGathering), 91

GemRequirement (class in coalib.bears.requirements.GemRequirement), 40

generate\_skip\_decorator() (in module coalib.testing.BearTestHelper), 99

get() (coalib.settings.Section.Section method), 96

get\_action\_info() (in module coalib.output.ConsoleInteraction), 61

get\_all\_bears\_names() (in module coalib.collecting.Collectors), 48

get\_available\_definitions() (coalib.bearlib.languages.documentation.DocstyleDefinition (class in coalib.bearlib.languages.documentation.DocstyleDefinition), 102 static method), 27

get\_config\_dir() (coalib.bears.Bear.Bear method), 45

get\_config\_directory() (in module coalib.settings.ConfigurationGathering), 91

get\_cpu\_count() (in module coalib.processes.Processing), 76

get\_data\_path() (in module coalib.misc.CachingUtilities), 53

get\_default\_actions() (in module coalib.processes.Processing), 76

get\_default\_version() (coalib.bearlib.languages.Language.Language method), 33

get\_exitcode() (in module coalib.misc.Exceptions), 56

get\_file\_dict() (in module coalib.processes.Processing), 76

get\_file\_list() (in module coalib.processes.Processing), 76

get\_filtered\_bears() (in module coalib.settings.ConfigurationGathering), 92

get\_global\_dependency\_results() (in module coalib.processes.BearRunning), 70

get\_ignore\_scope() (in module coalib.processes.Processing), 76

get\_indentation() (coalib.bearlib.spacing.SpacingHelper.SpacingHelper method), 38

get\_local\_dependency\_results() (in module coalib.processes.BearRunning), 70

get\_metadata() (coalib.bearlib.abstractions.SectionCreatable.SectionCreatable class method), 7

get\_metadata() (coalib.bears.Bear.Bear class method), 45

get\_metadata() (coalib.bears.LocalBear.LocalBear class method), 46

get\_metadata() (coalib.results.result\_actions.ResultAction.ResultAction class method), 80

get\_next\_global\_bear() (in module coalib.processes.BearRunning), 70

get\_non\_optional\_settings() (coalib.bearlib.abstractions.SectionCreatable.SectionCreatable class method), 7

get\_non\_optional\_settings() (coalib.bears.Bear.Bear class method), 45

get\_optional\_settings() (coalib.bearlib.abstractions.SectionCreatable.SectionCreatable class method), 7

get\_running\_processes() (in module coalib.processes.Processing), 76

get\_section() (coalib.parsing.ConfParser.ConfParser method), 67

get\_settings\_hash() (in module coalib.misc.CachingUtilities), 53

get\_shell\_type() (in module coalib.misc.Shell), 57

get\_uncached\_files() (coalib.misc.Caching.FileCache method), 52

get\_version() (coalib.bearlib.languages.Language.Language method), 102

glob() (in module coalib.parsing.Globbing), 68

glob() (in module coalib.settings.Setting), 98

glob\_escape() (in module coalib.parsing.Globbing), 68

glob\_list() (in module coalib.settings.Setting), 98

GlobalBear (class in coalib.bears.GlobalBear), 46

GoRequirement (class in coalib.bears.requirements.GoRequirement), 40

## H

has\_wildcard() (in module coalib.parsing.Globbing), 68

has\_cache() (in module coalib.misc.CachingUtilities), 53

HiddenResult (class in coalib.results.HiddenResult), 85

highlight\_text() (in module coalib.output.ConsoleInteraction), 61

## I

icollect() (in module coalib.collecting.Collectors), 48

icollect\_bears() (in module coalib.collecting.Collectors), 48

iglob() (in module coalib.parsing.Globbing), 68

IgnoreResultAction (class in coalib.results.result\_actions.IgnoreResultAction), 79

import\_objects() (in module coalib.collecting.Importers), 49

import\_objects() (in module coalib.collecting.Importers), 49

INCLUDE\_LOCAL\_FILES (coalib.bears.Bear.Bear attribute), 44

info() (coalib.output.printers.LogPrinter.LogPrinterMixin method), 59

initialize\_options() (coalib.misc.BuildManPage.BuildManPage method), 51

insert() (coalib.results.Diff.Diff method), 83

instantiate\_bears() (in module coalib.processes.Processing), 76

instantiate\_processes() (in module coalib.processes.Processing), 77

inverse\_dicts() (in module coalib.misc.DictUtilities), 56

is\_applicable() (coalib.results.result\_actions.ApplyPatchAction.ApplyPatchAction static method), 79



[is\\_applicable\(\) \(coala.results.result\\_actions.IgnoreResultAction.IgnoreResultAction static method\), 79](#)  
[is\\_applicable\(\) \(coala.results.result\\_actions.OpenEditorAction.OpenEditorAction static method\), 79](#)  
[is\\_applicable\(\) \(coala.results.result\\_actions.PrintDebugMessageAction.PrintDebugMessageAction static method\), 79](#)  
[is\\_applicable\(\) \(coala.results.result\\_actions.PrintMoreInfoAction.PrintMoreInfoAction static method\), 79](#)  
[is\\_applicable\(\) \(coala.results.result\\_actions.ResultAction.ResultAction static method\), 80](#)  
[is\\_applicable\(\) \(coala.results.result\\_actions.ShowPatchAction.ShowPatchAction static method\), 81](#)  
[is\\_comment\(\) \(coala.output.ConfWriter.ConfWriter static method\), 59](#)  
[is\\_enabled\(\) \(coala.settings.Section.Section method\), 96](#)  
[is\\_installed\(\) \(coala.bears.requirements.GemRequirement.GemRequirement method\), 40](#)  
[is\\_installed\(\) \(coala.bears.requirements.GoRequirement.GoRequirement method\), 40](#)  
[is\\_installed\(\) \(coala.bears.requirements.JuliaRequirement.JuliaRequirement method\), 41](#)  
[is\\_installed\(\) \(coala.bears.requirements.NpmRequirement.NpmRequirement method\), 41](#)  
[is\\_installed\(\) \(coala.bears.requirements.PackageRequirement.PackageRequirement method\), 41](#)  
[is\\_installed\(\) \(coala.bears.requirements.PipRequirement.PipRequirement method\), 42](#)  
[is\\_installed\(\) \(coala.bears.requirements.RscriptRequirement.RscriptRequirement method\), 42](#)

**J**

[join\(\) \(coala.results.TextRange.TextRange class method\), 90](#)  
[join\\_names\(\) \(in module coala.output.ConsoleInteraction\), 61](#)  
[JuliaRequirement \(class in coala.bears.requirements.JuliaRequirement\), 41](#)

**K**

[key \(coala.settings.Setting.Setting attribute\), 98](#)  
[kind\(\) \(coala.bears.Bear.Bear static method\), 45](#)  
[kind\(\) \(coala.bears.GlobalBear.GlobalBear static method\), 46](#)  
[kind\(\) \(coala.bears.LocalBear.LocalBear static method\), 46](#)

**L**

[Language \(class in coala.bearlib.languages.Language\), 31](#)  
[language \(coala.bearlib.languages.documentation.DocstyleDefinition.DocstyleDefinition attribute\), 27](#)  
[language \(coala.bearlib.languages.documentation.DocumentationComment.DocumentationComment attribute\), 30](#)

[LanguageDefinition \(class in coala.bearlib.languages.LanguageDefinition\), 34](#)  
[LanguageMeta \(class in coala.bearlib.languages.Language\), 34](#)  
[Languages \(class in coala.bearlib.languages.Language\), 34](#)  
[LANGUAGES \(coala.bears.Bear.Bear attribute\), 44](#)  
[LanguageNumberMeta \(class in coala.bearlib.languages.Language\), 34](#)  
[LicenseAction \(coala.bearlib.aspects.Metadata\), 10](#)  
[LICENSE \(coala.bears.Bear.Bear attribute\), 44](#)  
[limit\\_versions\(\) \(in module coala.bearlib.languages.Language\), 34](#)  
[line \(coala.results.TextPosition.TextPosition attribute\), 86](#)  
[LineDiff \(class in coala.parsing.LineParser\), 69](#)  
[linter\(\) \(in module coala.bearlib.abstractions.Linter\), 3](#)  
[ListLogPrinter \(class in coala.output.printers.ListLogPrinter\), 58](#)  
[LocalBear \(class in coala.bears.LocalBear\), 46](#)  
[LocalBearTestHelper \(class in coala.testing.LocalBearTestHelper\), 99](#)  
[location\\_repr\(\) \(coala.results.Result.Result method\), 87](#)  
[log\(\) \(coala.output.printers.LogPrinter.LogPrinterMixin method\), 59](#)  
[log\\_exception\(\) \(coala.output.printers.LogPrinter.LogPrinterMixin method\), 59](#)  
[log\\_level \(coala.output.printers.LogPrinter.LogPrinter attribute\), 58](#)  
[log\\_message\(\) \(coala.bears.Bear.Bear method\), 45](#)  
[log\\_message\(\) \(coala.output.printers.ListLogPrinter.ListLogPrinter method\), 58](#)  
[log\\_message\(\) \(coala.output.printers.LogPrinter.LogPrinter method\), 58](#)  
[log\\_message\(\) \(coala.output.printers.LogPrinter.LogPrinterMixin method\), 59](#)  
[LogMessage \(class in coala.processes.communication.LogMessage\), 69](#)  
[LogPrinter \(class in coala.output.printers.LogPrinter\), 58](#)  
[LogPrinterMixin \(class in coala.output.printers.LogPrinter\), 59](#)  
[LogPrinterThread \(class in coala.processes.LogPrinterThread\), 74](#)

## M

main() (in module coalib.coala), 101  
 main() (in module coalib.coala\_ci), 101  
 main() (in module coalib.coala\_delete\_orig), 101  
 main() (in module coalib.coala\_format), 101  
 main() (in module coalib.coala\_json), 101  
 MAINTAINERS (coalib.bears.Bear.Bear attribute), 44  
 maintainers (coalib.bears.Bear.Bear attribute), 45  
 MAINTAINERS\_EMAILS (coalib.bears.Bear.Bear attribute), 44  
 maintainers\_emails (coalib.bears.Bear.Bear attribute), 45  
 make\_temp() (in module coalib.misc.ContextManagers), 55  
 ManPageFormatter (class in coalib.misc.BuildManPage), 51  
 markers (coalib.bearlib.languages.documentation.DocstyleDefinition.DocstyleDefinition attribute), 28  
 merge() (coalib.settings.FunctionMetadata.FunctionMetadata class method), 95  
 merge\_section\_dicts() (in module coalib.settings.ConfigurationGathering), 93  
 Metadata (class in coalib.bearlib.aspects.Metadata), 10  
 metadata (coalib.bearlib.languages.documentation.DocstyleDefinition.DocstyleDefinition attribute), 28  
 metadata (coalib.bearlib.languages.documentation.DocumentationComment.DocumentationComment attribute), 30  
 Metadata.CommitMessage (class in coalib.bearlib.aspects.Metadata), 10  
 Metadata.CommitMessage.Body (class in coalib.bearlib.aspects.Metadata), 10  
 Metadata.CommitMessage.Body.Existence (class in coalib.bearlib.aspects.Metadata), 11  
 Metadata.CommitMessage.Body.Length (class in coalib.bearlib.aspects.Metadata), 11  
 Metadata.CommitMessage.Emptiness (class in coalib.bearlib.aspects.Metadata), 11  
 Metadata.CommitMessage.Shortlog (class in coalib.bearlib.aspects.Metadata), 11  
 Metadata.CommitMessage.Shortlog.ColonExistence (class in coalib.bearlib.aspects.Metadata), 11  
 Metadata.CommitMessage.Shortlog.FirstCharacter (class in coalib.bearlib.aspects.Metadata), 11  
 Metadata.CommitMessage.Shortlog.Length (class in coalib.bearlib.aspects.Metadata), 12  
 Metadata.CommitMessage.Shortlog.Tense (class in coalib.bearlib.aspects.Metadata), 12  
 Metadata.CommitMessage.Shortlog.TrailingPeriod (class in coalib.bearlib.aspects.Metadata), 12  
 missing\_dependencies() (coalib.bears.Bear.Bear class method), 45  
 mode\_format() (in module coalib.coala\_modes), 102  
 mode\_json() (in module coalib.coala\_modes), 102

mode\_non\_interactive() (in module coalib.coala\_modes), 102  
 mode\_normal() (in module coalib.coala\_modes), 102  
 modified (coalib.results.Diff.Diff attribute), 83  
 MutableValue (class in coalib.misc.MutableValue), 56

## N

name (coalib.bearlib.languages.documentation.DocumentationComment.DocumentationComment attribute), 29  
 name (coalib.bears.Bear.Bear attribute), 45  
 new\_result (coalib.bears.Bear.Bear attribute), 46  
 NoColorStyle (class in coalib.output.ConsoleInteraction), 60  
 non\_optional\_params (coalib.settings.FunctionMetadata.FunctionMetadata attribute), 96  
 nothing\_does\_it (class in module coalib.output.ConsoleInteraction), 62  
 NpmRequirement (class in coalib.bears.requirements.NpmRequirement), 41

## O

object\_defined\_in() (in module coalib.bearlib.languages.documentation.DocstyleDefinition.DocstyleDefinition), 50  
 OpenEditorAction (class in coalib.results.result\_actions.OpenEditorAction), 79  
 in optional\_params (coalib.settings.FunctionMetadata.FunctionMetadata attribute), 96  
 in original (coalib.results.Diff.Diff attribute), 83  
 overlaps() (coalib.results.Result.Result method), 87  
 overlaps() (coalib.results.TextRange.TextRange method), 90

## P

PackageRequirement (class in coalib.bears.requirements.PackageRequirement), 41  
 param\_end (coalib.bearlib.languages.documentation.DocstyleDefinition.DocstyleDefinition attribute), 27  
 param\_start (coalib.bearlib.languages.documentation.DocstyleDefinition.DocstyleDefinition attribute), 27  
 parent (coalib.bearlib.aspects.Metadata.Body attribute), 8  
 parent (coalib.bearlib.aspects.Metadata.Body.Existence attribute), 7  
 parent (coalib.bearlib.aspects.Metadata.Body.Length attribute), 7  
 parent (coalib.bearlib.aspects.Metadata.ColonExistence attribute), 8  
 parent (coalib.bearlib.aspects.Metadata.CommitMessage attribute), 10  
 parent (coalib.bearlib.aspects.Metadata.CommitMessage.Body attribute), 8

parent (coailib.bearlib.aspects.Metadata.CommitMessage.Body.Exists attribute), 8	parent (coailib.bearlib.aspects.Metadata.Shortlog.FirstCharacter attribute), 13
parent (coailib.bearlib.aspects.Metadata.CommitMessage.Body.Length attribute), 8	parent (coailib.bearlib.aspects.Metadata.Shortlog.Length attribute), 13
parent (coailib.bearlib.aspects.Metadata.CommitMessage.Emptyiness attribute), 8	parent (coailib.bearlib.aspects.Metadata.Shortlog.Tense attribute), 13
parent (coailib.bearlib.aspects.Metadata.CommitMessage.Shortlog attribute), 9	parent (coailib.bearlib.aspects.Metadata.Shortlog.TrailingPeriod attribute), 13
parent (coailib.bearlib.aspects.Metadata.CommitMessage.Shortlog.Clone attribute), 9	parent (coailib.bearlib.aspects.Metadata.Tense attribute), 14
parent (coailib.bearlib.aspects.Metadata.CommitMessage.Shortlog.FirstCharacter attribute), 9	parent (coailib.bearlib.aspects.Metadata.TrailingPeriod attribute), 14
parent (coailib.bearlib.aspects.Metadata.CommitMessage.Shortlog.Length attribute), 9	parent (coailib.bearlib.aspects.Redundancy.Clone attribute), 14
parent (coailib.bearlib.aspects.Metadata.CommitMessage.Shortlog.Tense attribute), 9	parent (coailib.bearlib.aspects.Redundancy.Redundancy attribute), 16
parent (coailib.bearlib.aspects.Metadata.CommitMessage.Shortlog.TrailingPeriod attribute), 9	parent (coailib.bearlib.aspects.Redundancy.Redundancy.Clone attribute), 14
parent (coailib.bearlib.aspects.Metadata.Emptyiness attribute), 10	parent (coailib.bearlib.aspects.Redundancy.Redundancy.UnreachableCode attribute), 15
parent (coailib.bearlib.aspects.Metadata.Existence attribute), 10	parent (coailib.bearlib.aspects.Redundancy.Redundancy.UnreachableCode.U attribute), 14
parent (coailib.bearlib.aspects.Metadata.FirstCharacter attribute), 10	parent (coailib.bearlib.aspects.Redundancy.Redundancy.UnreachableCode.U attribute), 15
parent (coailib.bearlib.aspects.Metadata.Length attribute), 10	parent (coailib.bearlib.aspects.Redundancy.Redundancy.UnusedImport attribute), 15
parent (coailib.bearlib.aspects.Metadata.Metadata attribute), 12	parent (coailib.bearlib.aspects.Redundancy.Redundancy.UnusedVariable attribute), 16
parent (coailib.bearlib.aspects.Metadata.Metadata.CommitMessage attribute), 12	parent (coailib.bearlib.aspects.Redundancy.Redundancy.UnusedVariable.Un attribute), 15
parent (coailib.bearlib.aspects.Metadata.Metadata.CommitMessage.Body attribute), 11	parent (coailib.bearlib.aspects.Redundancy.Redundancy.UnusedVariable.Un attribute), 15
parent (coailib.bearlib.aspects.Metadata.Metadata.CommitMessage.Body.Exists attribute), 11	parent (coailib.bearlib.aspects.Redundancy.Redundancy.UnusedVariable.Un attribute), 15
parent (coailib.bearlib.aspects.Metadata.Metadata.CommitMessage.Body.Length attribute), 11	parent (coailib.bearlib.aspects.Redundancy.UnreachableCode attribute), 16
parent (coailib.bearlib.aspects.Metadata.Metadata.CommitMessage.Body.UnreachableCode attribute), 11	parent (coailib.bearlib.aspects.Redundancy.UnreachableCode.UnreachableSt attribute), 16
parent (coailib.bearlib.aspects.Metadata.Metadata.CommitMessage.Shortlog attribute), 12	parent (coailib.bearlib.aspects.Redundancy.UnreachableCode.UnusedFunction attribute), 16
parent (coailib.bearlib.aspects.Metadata.Metadata.CommitMessage.Shortlog.Clone attribute), 11	parent (coailib.bearlib.aspects.Redundancy.UnreachableStatement attribute), 16
parent (coailib.bearlib.aspects.Metadata.Metadata.CommitMessage.Shortlog.FirstCharacter attribute), 11	parent (coailib.bearlib.aspects.Redundancy.UnusedFunction attribute), 16
parent (coailib.bearlib.aspects.Metadata.Metadata.CommitMessage.Shortlog.Length attribute), 12	parent (coailib.bearlib.aspects.Redundancy.UnusedGlobalVariable attribute), 17
parent (coailib.bearlib.aspects.Metadata.Metadata.CommitMessage.Shortlog.Tense attribute), 12	parent (coailib.bearlib.aspects.Redundancy.UnusedImport attribute), 17
parent (coailib.bearlib.aspects.Metadata.Metadata.CommitMessage.Shortlog.TrailingPeriod attribute), 12	parent (coailib.bearlib.aspects.Redundancy.UnusedLocalVariable attribute), 17
parent (coailib.bearlib.aspects.Metadata.Shortlog attribute), 13	parent (coailib.bearlib.aspects.Redundancy.UnusedParameter attribute), 17
parent (coailib.bearlib.aspects.Metadata.Shortlog.ColonExistence attribute), 13	parent (coailib.bearlib.aspects.Redundancy.UnusedVariable attribute), 18



[illegible]

PrintDebugMessageAction (class in coalib.results.result\_actions.PrintDebugMessageAction), 79  
 printer (coalib.output.printers.LogPrinter.LogPrinter attribute), 58  
 PrintMoreInfoAction (class in coalib.results.result\_actions.PrintMoreInfoAction), 79  
 process\_queues() (in coalib.processes.Processing), 77  
**R**  
 range() (coalib.results.Diff.Diff method), 83  
 Redundancy (class in coalib.bearlib.aspects.Redundancy), 14  
 Redundancy.Clone (class in coalib.bearlib.aspects.Redundancy), 14  
 Redundancy.UnreachableCode (class in coalib.bearlib.aspects.Redundancy), 14  
 Redundancy.UnreachableCode.UnreachableStatement (class in coalib.bearlib.aspects.Redundancy), 14  
 Redundancy.UnreachableCode.UnusedFunction (class in coalib.bearlib.aspects.Redundancy), 14  
 Redundancy.UnusedImport (class in coalib.bearlib.aspects.Redundancy), 15  
 Redundancy.UnusedVariable (class in coalib.bearlib.aspects.Redundancy), 15  
 Redundancy.UnusedVariable.UnusedGlobalVariable (class in coalib.bearlib.aspects.Redundancy), 15  
 Redundancy.UnusedVariable.UnusedLocalVariable (class in coalib.bearlib.aspects.Redundancy), 15  
 Redundancy.UnusedVariable.UnusedParameter (class in coalib.bearlib.aspects.Redundancy), 15  
 relative\_flat\_glob() (in module coalib.parsing.Globbing), 68  
 relative\_recursive\_glob() (in coalib.parsing.Globbing), 68  
 relative\_wildcard\_glob() (in coalib.parsing.Globbing), 69  
 remove() (coalib.results.Diff.Diff method), 84  
 remove\_range() (in module coalib.results.ResultFilter), 88  
 remove\_result\_ranges\_diffs() (in coalib.results.ResultFilter), 88  
 rename (coalib.results.Diff.Diff attribute), 84  
 renamed\_file() (coalib.results.SourceRange.SourceRange method), 89  
 replace() (coalib.results.Diff.Diff method), 84  
 replace\_spaces\_with\_tabs() (coalib.bearlib.spacing.SpacingHelper.SpacingHelper method), 38  
 replace\_stderr() (in coalib.misc.ContextManagers), 55  
 replace\_stdout() (in coalib.misc.ContextManagers), 55  
 replace\_tabs\_with\_spaces() (in coalib.bearlib.spacing.SpacingHelper.SpacingHelper method), 38  
 require\_setting() (in coalib.output.ConsoleInteraction), 64  
 REQUIREMENTS (coalib.bears.Bear.Bear attribute), 44  
 resolve() (in module coalib.collecting.Dependencies), 49  
 Result (class in coalib.results.Result), 86  
 ResultAction (class in coalib.results.result\_actions.ResultAction), 80  
 retrieve\_stderr() (in coalib.misc.ContextManagers), 55  
 retrieve\_stdout() (in coalib.misc.ContextManagers), 55  
 return\_sep (coalib.bearlib.languages.documentation.DocstyleDefinition.DocstyleDefinition attribute), 27  
 Root (class in coalib.bearlib.aspects), 19  
 Root.Metadata (class in coalib.bearlib.aspects), 21  
 Root.Metadata.CommitMessage (class in coalib.bearlib.aspects), 21  
 Root.Metadata.CommitMessage.Body (class in coalib.bearlib.aspects), 21  
 Root.Metadata.CommitMessage.Body.Existence (class in coalib.bearlib.aspects), 21  
 Root.Metadata.CommitMessage.Body.Length (class in coalib.bearlib.aspects), 21  
 Root.Metadata.CommitMessage.Emptiness (class in coalib.bearlib.aspects), 22  
 Root.Metadata.CommitMessage.Shortlog (class in coalib.bearlib.aspects), 22  
 Root.Metadata.CommitMessage.Shortlog.ColonExistence (class in coalib.bearlib.aspects), 22  
 Root.Metadata.CommitMessage.Shortlog.FirstCharacter (class in coalib.bearlib.aspects), 22  
 Root.Metadata.CommitMessage.Shortlog.Length (class in coalib.bearlib.aspects), 22  
 Root.Metadata.CommitMessage.Shortlog.Tense (class in coalib.bearlib.aspects), 23  
 Root.Metadata.CommitMessage.Shortlog.TrailingPeriod (class in coalib.bearlib.aspects), 23  
 Root.Redundancy (class in coalib.bearlib.aspects), 23  
 Root.Redundancy.Clone (class in coalib.bearlib.aspects), 23  
 Root.Redundancy.UnreachableCode (class in coalib.bearlib.aspects), 23  
 Root.Redundancy.UnreachableCode.UnreachableStatement (class in coalib.bearlib.aspects), 23  
 Root.Redundancy.UnreachableCode.UnusedFunction (class in coalib.bearlib.aspects), 24

- Root.Redundancy.UnusedImport (class in coalib.bearlib.aspects), 24
- Root.Redundancy.UnusedVariable (class in coalib.bearlib.aspects), 24
- Root.Redundancy.UnusedVariable.UnusedGlobalVariable (class in coalib.bearlib.aspects), 24
- Root.Redundancy.UnusedVariable.UnusedLocalVariable (class in coalib.bearlib.aspects), 24
- Root.Redundancy.UnusedVariable.UnusedParameter (class in coalib.bearlib.aspects), 25
- RscriptRequirement (class in coalib.bears.requirements.RscriptRequirement), 42
- run() (coalib.bears.Bear.Bear method), 46
- run() (coalib.bears.GlobalBear.GlobalBear method), 46
- run() (coalib.bears.LocalBear.LocalBear method), 46
- run() (coalib.misc.BuildManPage.BuildManPage method), 51
- run() (coalib.processes.LogPrinterThread.LogPrinterThread method), 74
- run() (in module coalib.processes.BearRunning), 70
- run\_bear() (in module coalib.processes.BearRunning), 71
- run\_bear\_from\_section() (coalib.bears.Bear.Bear method), 46
- run\_coala() (in module coalib.coala\_main), 101
- run\_global\_bear() (in coalib.processes.BearRunning), 72
- run\_global\_bears() (in coalib.processes.BearRunning), 72
- run\_interactive\_shell\_command() (in coalib.misc.Shell), 57
- run\_local\_bear() (in coalib.processes.BearRunning), 72
- run\_local\_bears() (in coalib.processes.BearRunning), 73
- run\_local\_bears\_on\_file() (in coalib.processes.BearRunning), 73
- run\_shell\_command() (in module coalib.misc.Shell), 57
- S**
- save\_sections() (in coalib.settings.ConfigurationGathering), 93
- Section (class in coalib.settings.Section), 96
- SectionCreatable (class in coalib.bearlib.abstractions.SectionCreatable), 6
- send\_msg() (in module coalib.processes.BearRunning), 73
- Setting (class in coalib.settings.Setting), 98
- settings\_changed() (in coalib.misc.CachingUtilities), 54
- setup\_dependencies() (coalib.bears.Bear.Bear static method), 46
- Shortlog (class in coalib.bearlib.aspects.Metadata), 12
- Shortlog.ColonExistence (class in coalib.bearlib.aspects.Metadata), 12
- Shortlog.FirstCharacter (class in coalib.bearlib.aspects.Metadata), 13
- Shortlog.Length (class in coalib.bearlib.aspects.Metadata), 13
- Shortlog.Tense (class in coalib.bearlib.aspects.Metadata), 13
- Shortlog.TrailingPeriod (class in coalib.bearlib.aspects.Metadata), 13
- show\_bear() (in coalib.output.ConsoleInteraction), 64
- show\_bears() (in coalib.output.ConsoleInteraction), 64
- show\_enumeration() (in coalib.output.ConsoleInteraction), 64
- show\_language\_bears\_capabilities() (in coalib.output.ConsoleInteraction), 65
- ShowPatchAction (class in coalib.results.result\_actions.ShowPatchAction), 80
- simplify\_section\_result() (in coalib.processes.Processing), 78
- simulate\_console\_inputs() (in coalib.misc.ContextManagers), 55
- source\_ranges\_match() (in coalib.results.ResultFilter), 88
- SourcePosition (class in coalib.results.SourcePosition), 88
- SourceRange (class in coalib.results.SourceRange), 89
- SpacingHelper (class in coalib.bearlib.spacing.SpacingHelper), 38
- split\_diff() (coalib.results.Diff.Diff method), 85
- start (coalib.results.TextRange.TextRange attribute), 90
- stats() (coalib.results.Diff.Diff method), 85
- str\_nodesc (coalib.settings.FunctionMetadata.FunctionMetadata attribute), 96
- str\_optional (coalib.settings.FunctionMetadata.FunctionMetadata attribute), 96
- styles (coalib.output.ConsoleInteraction.BackgroundMessageStyle attribute), 60
- styles (coalib.output.ConsoleInteraction.BackgroundSourceRangeStyle attribute), 60
- styles (coalib.output.ConsoleInteraction.NoColorStyle attribute), 60
- subaspect() (coalib.bearlib.aspects.aspectclass method), 25
- subaspect() (coalib.bearlib.aspects.meta.aspectclass method), 19
- subaspects (coalib.bearlib.aspects.Metadata.Body attribute), 8
- subaspects (coalib.bearlib.aspects.Metadata.Body.Existence attribute), 7

subaspects (coallib.bearlib.aspects.Metadata.Body.Length attribute), 8	subaspects (coallib.bearlib.aspects.Metadata.Metadata.CommitMessage.Shortlog attribute), 12
subaspects (coallib.bearlib.aspects.Metadata.ColonExistence attribute), 8	subaspects (coallib.bearlib.aspects.Metadata.Metadata.CommitMessage.Shortlog attribute), 12
subaspects (coallib.bearlib.aspects.Metadata.CommitMessage attribute), 10	subaspects (coallib.bearlib.aspects.Metadata.Shortlog attribute), 13
subaspects (coallib.bearlib.aspects.Metadata.CommitMessage.Body attribute), 8	subaspects (coallib.bearlib.aspects.Metadata.Shortlog.ColonExistence attribute), 13
subaspects (coallib.bearlib.aspects.Metadata.CommitMessage.Body.Length attribute), 8	subaspects (coallib.bearlib.aspects.Metadata.Shortlog.FirstCharacter attribute), 13
subaspects (coallib.bearlib.aspects.Metadata.CommitMessage.Body.Length attribute), 8	subaspects (coallib.bearlib.aspects.Metadata.Shortlog.Length attribute), 13
subaspects (coallib.bearlib.aspects.Metadata.CommitMessage.Body.Length attribute), 9	subaspects (coallib.bearlib.aspects.Metadata.Shortlog.Tense attribute), 13
subaspects (coallib.bearlib.aspects.Metadata.CommitMessage.Body.Length attribute), 10	subaspects (coallib.bearlib.aspects.Metadata.Shortlog.TrailingPeriod attribute), 13
subaspects (coallib.bearlib.aspects.Metadata.CommitMessage.Body.Length attribute), 9	subaspects (coallib.bearlib.aspects.Metadata.Tense attribute), 14
subaspects (coallib.bearlib.aspects.Metadata.CommitMessage.Body.Length attribute), 9	subaspects (coallib.bearlib.aspects.Metadata.TrailingPeriod attribute), 14
subaspects (coallib.bearlib.aspects.Metadata.CommitMessage.Body.Length attribute), 9	subaspects (coallib.bearlib.aspects.Redundancy.Clone attribute), 14
subaspects (coallib.bearlib.aspects.Metadata.CommitMessage.Body.Length attribute), 9	subaspects (coallib.bearlib.aspects.Redundancy.Redundancy attribute), 16
subaspects (coallib.bearlib.aspects.Metadata.CommitMessage.Body.Length attribute), 9	subaspects (coallib.bearlib.aspects.Redundancy.Redundancy.Clone attribute), 14
subaspects (coallib.bearlib.aspects.Metadata.Emptyiness attribute), 10	subaspects (coallib.bearlib.aspects.Redundancy.Redundancy.UnreachableCode attribute), 15
subaspects (coallib.bearlib.aspects.Metadata.Existence attribute), 10	subaspects (coallib.bearlib.aspects.Redundancy.Redundancy.UnreachableCode attribute), 14
subaspects (coallib.bearlib.aspects.Metadata.FirstCharacter attribute), 10	subaspects (coallib.bearlib.aspects.Redundancy.Redundancy.UnreachableCode attribute), 15
subaspects (coallib.bearlib.aspects.Metadata.Length attribute), 10	subaspects (coallib.bearlib.aspects.Redundancy.Redundancy.UnusedImport attribute), 15
subaspects (coallib.bearlib.aspects.Metadata.Metadata attribute), 12	subaspects (coallib.bearlib.aspects.Redundancy.Redundancy.UnusedVariable attribute), 16
subaspects (coallib.bearlib.aspects.Metadata.Metadata.CommitMessage attribute), 12	subaspects (coallib.bearlib.aspects.Redundancy.Redundancy.UnusedVariable attribute), 15
subaspects (coallib.bearlib.aspects.Metadata.Metadata.CommitMessage attribute), 11	subaspects (coallib.bearlib.aspects.Redundancy.Redundancy.UnusedVariable attribute), 15
subaspects (coallib.bearlib.aspects.Metadata.Metadata.CommitMessage attribute), 11	subaspects (coallib.bearlib.aspects.Redundancy.Redundancy.UnusedVariable attribute), 15
subaspects (coallib.bearlib.aspects.Metadata.Metadata.CommitMessage attribute), 11	subaspects (coallib.bearlib.aspects.Redundancy.UnreachableCode attribute), 16
subaspects (coallib.bearlib.aspects.Metadata.Metadata.CommitMessage attribute), 11	subaspects (coallib.bearlib.aspects.Redundancy.UnreachableCode.UnreachableCode attribute), 16
subaspects (coallib.bearlib.aspects.Metadata.Metadata.CommitMessage attribute), 12	subaspects (coallib.bearlib.aspects.Redundancy.UnreachableCode.UnusedFunction attribute), 16
subaspects (coallib.bearlib.aspects.Metadata.Metadata.CommitMessage attribute), 11	subaspects (coallib.bearlib.aspects.Redundancy.UnreachableStatement attribute), 16
subaspects (coallib.bearlib.aspects.Metadata.Metadata.CommitMessage attribute), 12	subaspects (coallib.bearlib.aspects.Redundancy.UnusedFunction attribute), 17
subaspects (coallib.bearlib.aspects.Metadata.Metadata.CommitMessage attribute), 12	subaspects (coallib.bearlib.aspects.Redundancy.UnusedGlobalVariable attribute), 17

subaspects (coalib.bearlib.aspects.Redundancy.UnusedImport attribute), 17

subaspects (coalib.bearlib.aspects.Redundancy.UnusedLocalSubaspects (coalib.bearlib.aspects.Root.Redundancy.UnusedVariable.Unused attribute), 17

subaspects (coalib.bearlib.aspects.Redundancy.UnusedParameter attribute), 17

subaspects (coalib.bearlib.aspects.Redundancy.UnusedVariable attribute), 18

subaspects (coalib.bearlib.aspects.Redundancy.UnusedVariable attribute), 17

subaspects (coalib.bearlib.aspects.Redundancy.UnusedVariable attribute), 18

subaspects (coalib.bearlib.aspects.Redundancy.UnusedVariable attribute), 18

subaspects (coalib.bearlib.aspects.Root attribute), 25

subaspects (coalib.bearlib.aspects.Root.Metadata attribute), 23

subaspects (coalib.bearlib.aspects.Root.Metadata.CommitMessage attribute), 23

subaspects (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Body attribute), 22

subaspects (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Body.Existence attribute), 21

subaspects (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Body.Length attribute), 22

subaspects (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Body.Taste attribute), 22

subaspects (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Body.TasteError, 19, 25

subaspects (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Body.TasteMetaClass in coalib.bearlib.aspects.taste), 19

subaspects (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Body.Tastes (coalib.bearlib.aspects.aspectclass attribute), 25

subaspects (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Body.Tastes.Shortlog.ColumnExistence attribute), 18

subaspects (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Body.Tastes.Shortlog.FirstCharacter attribute), 19

subaspects (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Body.Tastes.Shortlog.Length attribute), 22

subaspects (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Body.Tastes.Shortlog.Tense attribute), 23

subaspects (coalib.bearlib.aspects.Root.Metadata.CommitMessage.Body.Tastes.Shortlog.TrailingPeriod attribute), 23

subaspects (coalib.bearlib.aspects.Root.Redundancy attribute), 25

subaspects (coalib.bearlib.aspects.Root.Redundancy.Clone attribute), 23

subaspects (coalib.bearlib.aspects.Root.Redundancy.UnreachableCode attribute), 24

subaspects (coalib.bearlib.aspects.Root.Redundancy.UnreachableCode.UnreachableStatement attribute), 24

subaspects (coalib.bearlib.aspects.Root.Redundancy.UnreachableCode.UnreachableFunction attribute), 24

subaspects (coalib.bearlib.aspects.Root.Redundancy.UnusedImport attribute), 24

subaspects (coalib.bearlib.aspects.Root.Redundancy.UnusedVariable attribute), 25

subprocess\_timeout() (in module coalib.misc.ContextManagers), 56

SUCCESS\_MESSAGE (coalib.results.result\_actions.ApplyPatchAction.ApplyPatchAction attribute), 78

SUCCESS\_MESSAGE (coalib.results.result\_actions.IgnoreResultAction.IgnoreResultAction attribute), 79

SUCCESS\_MESSAGE (coalib.results.result\_actions.OpenEditorAction.OpenEditorAction attribute), 79

SUCCESS\_MESSAGE (coalib.results.result\_actions.ResultAction.ResultAction attribute), 80

SUCCESS\_MESSAGE (coalib.results.result\_actions.ShowPatchAction.ShowPatchAction attribute), 80

suppress\_stdout() (in module coalib.misc.ContextManagers), 56

task\_done() (in module coalib.processes.BearRunning), 14

Taste (in module coalib.bearlib.aspects), 25

TasteError, 19, 25

TasteMetaClass in coalib.bearlib.aspects.taste), 19

Tastes (coalib.bearlib.aspects.aspectclass attribute), 25

Tastes.Shortlog.ColumnExistence (coalib.bearlib.aspects.aspectbase attribute), 18

Tastes.Shortlog.FirstCharacter (coalib.bearlib.aspects.meta.aspectclass attribute), 19

Tastes.Shortlog.Length (class in coalib.bearlib.aspects.Metadata), 13

TextPosition (class in coalib.results.TextPosition), 89

TextRange (class in coalib.results.TextRange), 90

to\_camelcase() (in module coalib.bearlib.naming\_conventions), 36

to\_pascalcase() (in module coalib.bearlib.naming\_conventions), 36

to\_snakecase() (in module coalib.bearlib.naming\_conventions), 37

to\_spacecase() (in module coalib.bearlib.naming\_conventions), 37

to\_string\_dict() (coalib.processes.communication.LogMessage.LogMessage method), 40

to\_string\_dict() (coalib.results.Result.Result method), 87

typed\_dict() (in module coalib.settings.Setting), 98

typed\_list() (in module coalib.settings.Setting), 98



typed\_ordered\_dict() (in module coalib.settings.Setting),  
98

## U

unified\_diff (coalib.results.Diff.Diff attribute), 85  
 UnreachableCode (class coalib.bearlib.aspects.Redundancy), 16  
 UnreachableCode.UnreachableStatement (class coalib.bearlib.aspects.Redundancy), 16  
 UnreachableCode.UnusedFunction (class coalib.bearlib.aspects.Redundancy), 16  
 UnreachableStatement (class coalib.bearlib.aspects.Redundancy), 16  
 untrack\_files() (coalib.misc.Caching.FileCache method), 53  
 UnusedFunction (class coalib.bearlib.aspects.Redundancy), 16  
 UnusedGlobalVariable (class coalib.bearlib.aspects.Redundancy), 17  
 UnusedImport (class coalib.bearlib.aspects.Redundancy), 17  
 UnusedLocalVariable (class coalib.bearlib.aspects.Redundancy), 17  
 UnusedParameter (class coalib.bearlib.aspects.Redundancy), 17  
 UnusedVariable (class coalib.bearlib.aspects.Redundancy), 17  
 UnusedVariable.UnusedGlobalVariable (class coalib.bearlib.aspects.Redundancy), 17  
 UnusedVariable.UnusedLocalVariable (class coalib.bearlib.aspects.Redundancy), 17  
 UnusedVariable.UnusedParameter (class coalib.bearlib.aspects.Redundancy), 18  
 update() (coalib.settings.Section.Section method), 96  
 update\_ordered\_dict\_key() (in module coalib.misc.DictUtilities), 56  
 update\_setting() (coalib.settings.Section.Section method), 97  
 update\_settings\_db() (in module coalib.misc.CachingUtilities), 54  
 url() (in module coalib.settings.Setting), 99  
 user\_options (coalib.misc.BuildManPage.BuildManPage attribute), 51

## V

validate\_results() (in module coalib.processes.BearRunning), 74  
 verify\_local\_bear() (in module coalib.testing.LocalBearTestHelper), 100

## W

warn() (coalib.output.printers.LogPrinter.LogPrinterMixin method), 59

warn\_config\_absent() (in module coalib.settings.ConfigurationGathering),  
93

warn\_nonexistent\_targets() (in module coalib.settings.ConfigurationGathering),  
93

write() (coalib.misc.Caching.FileCache method), 53

write\_section() (coalib.output.ConfWriter.ConfWriter method), 59

write\_sections() (coalib.output.ConfWriter.ConfWriter method), 59

## Y

yield\_ignore\_ranges() (in module coalib.processes.Processing), 78

yield\_tab\_lengths() (coalib.bearlib.spacing.SpacingHelper.SpacingHelper method), 38