repathlib Documentation

Release 0.1.5

Matthew Badger

Feb 11, 2019

Contents:

1 API 3

2 Indices and tables 7

repathlib combines the simplicity of pathlib with the power of regular expressions, making it easier to find files and directories.

repathlib provides a `Path` class which subclasses `pathlib.Path`, and adds the following methods to it:

- *search()*
- *match_()*
- *fullmatch()*
- *findall()*
- *finditer()*
- *reiterdir()*

CHAPTER 1

API

This documentation includes only those methods that have been added to the base `pathlib.Path` class; refer to that class's documentation for everything else. In addition, `pathlib.Path`'s definition overrides the `__new__` method to construct either a `PosixPath` or a `WindowsPath`. This doesn't have any impact on the end user, but means we have to include one of those classes here to use autodoc:.

class repathlib.repathlib.**PosixPath**

search(*pattern: Union[str, Pattern[AnyStr]], part: str = 'name', flags: int = 0*) → Optional[Match[AnyStr]]

Apply `re.search()` to the path

Parameters

- **pattern** (*Union[str, Pattern]*) – Regular expression string, or a compiled regular expression
- **part** (*str, optional {'name', 'full', 'suffix', 'suffixes', 'parent', 'stem', 'resolved'}, default 'name'*) – Part of the Path to search; `full` is the complete (but unresolved) path, `suffixes` is all suffixes joined with `.`, `resolved` is the complete resolved path, and the remaining options are exactly that property or attribute of the Path
- **flags** (*int, optional, default 0*) – Flags passed to search

Returns Match of the pattern to the given part of the path, or None

Return type Optional[Match]

match_(*pattern: Union[str, Pattern[AnyStr]], part: str = 'name', flags: int = 0*) → Optional[Match[AnyStr]]

Apply `re.match()` to the path

Parameters

- **pattern** (*Union[str, Pattern]*) – Regular expression string, or a compiled regular expression

- **part** (*str, optional {'name', 'full', 'suffix', 'suffixes', 'parent', 'stem', 'resolved'}, default 'name'*) – Part of the Path to search; `full` is the complete (but unresolved) path, `suffixes` is all suffixes joined with `.`, `resolved` is the complete resolved path, and the remaining options are exactly that property or attribute of the Path
- **flags** (*int, optional, default 0*) – Flags passed to match

Returns Match of the pattern to the given part of the path, or None

Return type Optional[Match]

> **Warning:** Note that this method is `match_`, not `match`, to maintain the existing method `pathlib.PurePath.match()`.

fullmatch(*pattern: Union[str, Pattern[AnyStr]], part: str = 'name', flags: int = 0*) → Optional[Match[AnyStr]]

Apply `re.fullmatch()` to the path

Parameters

- **pattern** (*Union[str, Pattern]*) – Regular expression string, or a compiled regular expression
- **part** (*str, optional {'name', 'full', 'suffix', 'suffixes', 'parent', 'stem', 'resolved'}, default 'name'*) – Part of the Path to search; `full` is the complete (but unresolved) path, `suffixes` is all suffixes joined with `.`, `resolved` is the complete resolved path, and the remaining options are exactly that property or attribute of the Path
- **flags** (*int, optional, default 0*) – Flags passed to fullmatch

Returns Match of the pattern to the given part of the path, or None

Return type Optional[Match]

findall(*pattern: Union[str, Pattern[AnyStr]], part: str = 'name', flags: int = 0*) → Optional[Match[AnyStr]]

Apply `re.findall()` to the path

Parameters

- **pattern** (*Union[str, Pattern]*) – Regular expression string, or a compiled regular expression
- **part** (*str, optional {'name', 'full', 'suffix', 'suffixes', 'parent', 'stem', 'resolved'}, default 'name'*) – Part of the Path to search; `full` is the complete (but unresolved) path, `suffixes` is all suffixes joined with `.`, `resolved` is the complete resolved path, and the remaining options are exactly that property or attribute of the Path
- **flags** (*int, optional, default 0*) – Flags passed to findall

Returns Match of the pattern to the given part of the path, or None

Return type Optional[Match]

finditer(*pattern: Union[str, Pattern[AnyStr]], part: str = 'name', flags: int = 0*) → Optional[Match[AnyStr]]

Apply `re.finditer()` to the path

Parameters

- **pattern** (*Union[str, Pattern]*) – Regular expression string, or a compiled regular expression
- **part** (*str, optional {'name', 'full', 'suffix', 'suffixes', 'parent', 'stem', 'resolved'}, default 'name'*) – Part of the Path to search; `full` is the complete (but unresolved) path, `suffixes` is all suffixes joined with `.`, `resolved` is the complete resolved path, and the remaining options are exactly that property or attribute of the Path
- **flags** (*int, optional, default 0*) – Flags passed to finditer

Returns Match of the pattern to the given part of the path, or None

Return type Optional[Match]

reiterdir(*pattern: Union[str, Pattern[AnyStr]] = None*, *method: str = 'search'*, *part: str = 'name'*, *yield_type: str = 'path'*, *flags=0*) → Generator[Union[repathlib.repathlib.RePath, Match[AnyStr], Tuple[repathlib.repathlib.RePath, Match[AnyStr]]], None, None]

Iterate over the path, yielding paths which match the given pattern. If no pattern is given, defaults to using Path.iterdir.

Parameters

- **pattern** (*str, default None*) – Regular expression string, or a compiled regular expression
- **method** (*str, optional {'search', 'match', 'fullmatch', 'findall', 'finditer'}, default 'search'*) – Method from `re` to use
- **part** (*str, optional {'name', 'full', 'suffix', 'suffixes', 'parent', 'stem', 'resolved'}, default 'name'*) – Part of the Path to search; `full` is the complete (but unresolved) path, `suffixes` is all suffixes joined with `.`, `resolved` is the complete resolved path, and the remaining options are exactly that property or attribute of the Path
- **yield_type** (*str, optional ['path', 'match', 'tuple'], default 'path'*) – Yield type of the generator. If `path`, returns the Path object; if `match`, returns the re.Match object; if `tuple`, returns a tuple of (Path, re.Match)
- **flags** (*int, optional, default 0*) – Flags passed to the re function

Yields *Generator[Union[Path, Match, Tuple[Path, Match]], None, None]* – If `yield_type` is 'path' (the default), yields Path instances; if 'match', yields matches; if 'tuple', yields a tuple of path and match

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Index

F

findall() (repathlib.repathlib.PosixPath method), 4
finditer() (repathlib.repathlib.PosixPath method), 4
fullmatch() (repathlib.repathlib.PosixPath method), 4

M

match_() (repathlib.repathlib.PosixPath method), 3

P

PosixPath (class in repathlib.repathlib), 3

R

reiterdir() (repathlib.repathlib.PosixPath method), 5

S

search() (repathlib.repathlib.PosixPath method), 3