

Directory Commands

<code>dirname()</code>	parent, <code>os.path.dirname()</code>
classmethod <code>getcwd()</code>	Return the current working directory as a path object. <code>os.getcwd()</code>
<code>dirs(args, *kwargs)</code>	<i>List of this directory's subdirectories.</i> The elements of the list are Path objects. This does not walk recursively into subdirectories (but see <code>walkdirs()</code>). Accepts parameters to <code>listdir()</code> .
<code>exists()</code>	<i>os.path.exists()</i> Return True if path refers to an existing path or an open file descriptor. Returns False for broken symbolic links or missing permissions
property <code>drive</code>	The drive specifier, for example 'C:'.
<code>expand()</code>	Clean up a filename by calling <code>expandvars()</code> , <code>expanduser()</code> , and <code>normpath()</code> on it.
<code>expanduser()</code>	Return the argument with an initial component of ~ or ~user replaced by that user's home directory.
<code>expandvars()</code>	Return the argument with environment variables expanded. Substrings of the form \$name or \${name} are replaced by the value of environment variable name. Bad variables are left unchanged.
property <code>ext</code>	The file extension, for example '.py'.
<code>files(args, *kwargs)</code>	List of the files in self. The elements of the list are Path objects. This does not walk into subdirectories (see <code>walkfiles()</code>). Accepts parameters to <code>listdir()</code> .
<code>fnmatch.fnmatch(self.name, pattern, normcase=Normcase)</code>	Return True if self.name matches the given pattern. pattern - A filename pattern with wildcards, for example '*.py'. normcase - (optional) A function used to normalize the pattern and filename before matching.
<code>glob(pattern)</code>	Return a list of Path objects that match the pattern. pattern - a path relative to this directory, with wildcards. For example, <code>Path('/users').glob('/bin/')</code> returns a list of all the files users have in their bin directories.
<code>get_owner()</code>	Return the name of the owner of this file or directory. Follow symbolic links.
<code>getatime()</code>	Return the time of last access of path.
<code>getctime()</code>	Return the system's ctime which, on some systems (like Unix) is the time of the last metadata change
<code>getsize()</code>	Return the size, in bytes, of path.



By mad100141

Published 19th April, 2023.

Last updated 20th April, 2023.

Page 1 of 6.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

Directory Commands (cont)

<code>getmtime()</code>	Return the time of last modification of path.
<code>iglob(pattern)</code>	Return an iterator of Path objects that match the pattern. pattern - a path relative to this directory, with wildcards.
<code>in_place(mode='r', buffering=-1, encoding=None, errors=None, newline=None, backup_extension=None)</code>	A context in which a file may be re-written in-place with new content. Yields a tuple of (readable, writable) file objects, where writable replaces readable. A ValueError is raised on invalid modes.
<code>isabs()</code>	Return True if path is an absolute pathname. On Unix, that means it begins with a slash
<code>isdir()</code>	Return True if path is an existing directory
<code>isfile()</code>	Return True if path is an existing regular file.
<code>islink()</code>	Return True if path refers to an existing directory entry that is a symbolic link.
<code>ismount()</code>	Return True if pathname path is a mount point: a point in a file system where a different file system has been mounted

Creation Commands

<code>walk(m atc h=None, errors='strict')</code>	Iterator over files and subdirs, recursively. The iterator yields Path objects naming each child item of this directory and its descendants.
<code>symlink(newlink=None)</code>	Create a symbolic link at newlink, pointing here. default = cwd
<code>text(encoding=None, errors='strict')</code>	Legacy function to read text. Converts all newline sequences to \n.
<code>touch()</code>	Set the access/modified times of this file to the current time. Create the file if it does not exist.
<code>unlink()</code>	Remove (delete) the file path. <code>os.remove()</code>
<code>unlink_p()</code>	Like remove(), but does not raise an exception if the file does not exist.
<code>classmethod using_module(module)</code>	
<code>utime(args, *kwargs)</code>	Set the access and modified times of this file.
<code>walkdirs(*args, **kwargs)</code>	Iterator over subdirs, recursively.
<code>walkfiles(*args, **kwargs)</code>	Iterator over files, recursively.
<code>with_suffix(suffix)</code>	Return a new path with the file suffix changed (or added, if none)
<code>write_bytes(bytes, append=False)</code>	Open this file and write the given bytes to it.
<code>write_lines(lines, encoding=None, errors='strict', linesep=_default_linesep, append=False)</code>	Write the given lines of text to this file.
<code>write_text(text, encoding=None, errors='strict', linesep=os.linesep, append=False)</code>	Write the given text to this file.



By mad100141

Published 19th April, 2023.

Last updated 20th April, 2023.

Page 2 of 6.

Sponsored by CrosswordCheats.com

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

Creation Commands (cont)

`class path.TempDir(args, *kwargs)` A temporary directory via `tempfile.mkdtemp()`, and constructed with the same parameters that you can use as a context manager.

Directory Manipulation Commands

`joinpath(*others)` Join one or more path segments intelligently. The return value is the concatenation of path and all members of `*paths`, with exactly one directory separator following each non-empty part, except the last. That is, the result will only end in a separator if the last part is either empty or ends in a separator.

`lines(encoding=None, errors=None, retain=True)` Open this file, read all lines, return them in a list.

`link(newpath)` Create a hard link at `newpath`, pointing to this file.

`listdir(match=None)` List of items in this directory.

`lstat()` Like `stat()`, but do not follow symbolic links.

`makedirs(mode=0o777)` Recursive directory creation function. Like `mkdir()`, but makes all intermediate-level directories needed to contain the leaf directory.

`makedirs_p(mode=0o777)` Like `makedirs()`, but does not raise an exception if the directory already exists.

`merge_tree(dst, symlinks=False, *, copy_function=shutil.copy2, ignore=lambda dir, contents: ...)` Copy entire contents of self to dst, overwriting existing contents in dst with those in self. Pass `symlinks=True` to copy symbolic links as links. Accepts a `copy_function`, similar to `copytree`.

`mkdir(mode=0o777)` Create a directory named path with numeric mode mode.

`mkdir_p(mode=0o777)` Like `mkdir()`, but does not raise an exception if the directory already exists.

`module = <module 'posixpath' (frozen)>` The path module to use for path operations.

`move(dst, copy_function=copy2)` Recursively move a file or directory to another location. This is similar to the Unix "mv" command. Return the file or directory's destination.

property **mtime** Last modified time of the file.

property **name** The name of this file or directory without the full path. `basename()`

`normcase()` Normalize the case of a pathname.



By mad100141

Published 19th April, 2023.

Last updated 20th April, 2023.

Page 3 of 6.

Sponsored by CrosswordCheats.com

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

Directory Manipulation Commands (cont)

<code>normpath()</code>	Normalize a pathname by collapsing redundant separators and up-level references so that <code>A//B</code> , <code>A/B/</code> , <code>A/.B</code> and <code>A/foo/../B</code> all become <code>A/B</code> . This string manipulation may change the meaning of a path that contains
<code>open(args, *kwargs)</code>	Open this file and return a corresponding file object.
<code>property owner</code>	Name of the owner of this file or directory.
<code>property parent</code>	This path's parent directory, as a new Path object.
<code>parts()</code>	<code>Path('/foo/bar/baz').parts() -> (Path('/'), 'foo', 'bar', 'baz')</code>
<code>pathconf(-name)</code>	Return system configuration information relevant to a named file

class path.Path - represents filesystem path

<code>abspath()</code>	Return a normalized absolutized version of the pathname <code>path</code> . <code>normpath(join(os.getcwd(), path))</code>
<code>access(*args, **kwargs)</code>	Return does the real user have access to this path. <code>os.access()</code>
<code>basename()</code>	Return the base name of pathname <code>path</code> . This is the second element of the pair returned by passing <code>path</code> to the function <code>split()</code> .
<code>bytes()</code>	Open this file, read all bytes, return them as a string.
<code>cd()</code>	<code>os.chdir(path)</code> Change the current working directory to <code>path</code> .
<code>chdir()</code>	<code>os.chdir(path)</code>
<code>chown(uid=-1, gid=-1)</code>	Change the owner and group by names or numbers.
<code>chroot()</code>	Change the root directory of the current process to <code>path</code> .
<code>chunks(size, args, *kwargs)</code>	Returns a generator yielding chunks of the file, so it can be read piece by piece with a simple for loop. Any argument you pass after <code>size</code> will be passed to <code>open()</code> .
<code>copy(dst, *, follow_symlinks=True)</code>	Copy data and mode bits ("cp src dst"). Return the file's destination. The destination may be a directory.
<code>copy2(dst, *, follow_symlinks=True)</code>	Copy data and metadata. Return the file's destination. Metadata is copied with <code>copystat()</code> . Please see the <code>copystat</code> function for more information.
<code>copyfile(dst, *, follow_symlinks=True)</code>	Copy data from <code>src</code> to <code>dst</code> in the most efficient way possible.



By mad100141

Published 19th April, 2023.

Last updated 20th April, 2023.

Page 4 of 6.

Sponsored by CrosswordCheats.com

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

class path.Path - represents filesystem path (cont)

<code>copymode(dst, *, follow_symlinks=True)</code>	Copy mode bits from src to dst.
<code>copystat(dst, *, follow_symlinks=True)</code>	Copy file metadata. Copy the permission bits, last access time, last modification time, and flags from src to dst.
<code>copytree(dst, symlinks=False, ignore=None, copy_function=copy2, ignore_dangling_symlinks=False, dirs_exist_ok=False)</code>	Recursively copy a directory tree and return the destination directory. If exception(s) occur, an Error is raised with a list of reasons.
<code>property ctime</code>	Creation time of the file.

Remove Change Commands

<code>remove()</code>	Remove (delete) the file path. Raises Error if Directory or FileNotFoundError.
<code>remove_p()</code>	Like <code>remove()</code> , but does not raise an exception if the file does not exist.
<code>removedirs()</code>	Remove directories recursively.
<code>removedirs_p()</code>	Like <code>removedirs()</code> , but does not raise an exception if the directory is not empty or does not exist.
<code>rename(new)</code>	Rename the file or directory src to dst.
<code>renames(new)</code>	Recursive directory or file renaming function. Works like <code>rename()</code> , except creation of any intermediate directories needed to make the new pathname good is attempted first.
<code>rmdir()</code>	<code>os.rmdir()</code>
<code>rmdir_p()</code>	Like <code>rmdir()</code> , but does not raise an exception if the directory is not empty or does not exist.
<code>rmtree(ignore_errors=False, onerror=None, *, dir_fd=None)</code>	Recursively delete a directory tree.
<code>rmtree_p()</code>	Like <code>rmtree()</code> , but does not raise an exception if the directory does not exist.
<code>samefile(other)</code>	Return True if both pathname arguments refer to the same file or directory.
<code>property size</code>	Size of the file, in bytes.
<code>special = functools.partial(<class 'path.SpecialResolver'>, <class 'path.Path'>)</code>	
<code>splitall()</code>	Return a list of the path components in this path.
<code>splitdrive()</code>	Return two-tuple of <code>.drive</code> and rest without drive.
<code>splittext()</code>	Return two-tuple of <code>.stripext()</code> and <code>.ext</code> .
<code>splitpath()</code>	Return two-tuple of <code>.parent</code> , <code>.name</code> .
<code>stat()</code>	Perform a <code>stat()</code> system call on this path.



By mad100141

Published 19th April, 2023.

Last updated 20th April, 2023.

Page 5 of 6.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

Remove Change Commmands (cont)

statvfs()	Perform a statvfs() system call on this path. f_bsize, f_frsize, f_blocks, f_bfree, f_bavail, f_files, f_ffree, f_favail, f_flag, f_namemax, f_fsid.
property stem	The same as name(), but with one file extension stripped off.
stripext()	Remove one file extension from the path.

Read Commands

read_bytes()	Return the contents of this file as bytes.
read_hash(hash_name)	Calculate given hash for this file.
read_hexhash(hash_name)	Calculate given hash for this file, returning hexdigest.
read_md5()	Calculate the md5 hash for this file.
read_text(encoding=None, errors=None)	Open this file, read it in, return the content as a string. Optional parameters are passed to open().
readlink()	Return the path to which this symbolic link points. The result may be an absolute or a relative path.
readlinkabs()	Return the path to which this symbolic link points.
realpath()	<code>os.path.realpath()</code>
relpath(start='.')	Return this path as a relative path, based from start, which defaults to the current working directory.
relpathto(dest)	Return a relative path from self to dest.



By mad100141

Published 19th April, 2023.

Last updated 20th April, 2023.

Page 6 of 6.

Sponsored by CrosswordCheats.com

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>