Directory Commands	
`dirname()`	parent, os.path.dirname()
classmethod `getcwd()`	Return the current working directory as a path object. `os.get cwd()`
`dirs(args, *kwargs)`	*List of this directory's subdirectories.* The elements of the list are Path objects. This does not walk recursively into subdirectories (but see walkdirs()). Accepts parameters to listdir().
`exists()`	*os.path.exists() Return True if path refers to an existing path or an open file descriptor.*Returns False for broken symbolic links or missing permissions
property **drive**	The drive specifier, for example 'C:'.
`expand()`	Clean up a filename by calling expandvars(), expanduser(), and normpath() on it.
expanduser()	Return the argument with an initial component of ~ or ~user replaced by that user's home directory.
expandvars()	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 **ext**	The file extension, for example '.py'.
`files(args, *kwargs)`	List of the files in self. The elements of the list are Path objects. This does not walk into subdirectories (see walkfiles()). Accepts parameters to listdir().
`fnmatc h(p attern, normca se= N one)`	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.
`glob(p attern)`	Return a list of Path objects that match the pattern. **pattern** - a path relative to this directory, with wildcards. For example, Path('/users').glob('*/bin/*') returns a list of all the files users have in their bin directories.
get_owner()	Return the name of the owner of this file or directory. Follow symbolic links.
getatime()	Return the time of last access of path.
getctime()	Return the system's ctime which, on some systems (like Unix) is the time of the last metadata change
getsize()	Return the size, in bytes, of path.

Directory Commands (cont)	
getmtime()	Return the time of last modification of path.
iglob(pattern)	Return an iterator of Path objects that match the pattern. pattern - a path relative to this directory, with wildcards.
in_place(mode='r', buffering=-1, encoding=None, errors=None, newline=None, backup_extensio-n=None)	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.
isabs()	Return True if path is an absolute pathname. On Unix, that means it begins with a slash
`isdir()`	Return True if path is an existing directory
isfile()	Return True if path is an existing regular file.
islink()	Return True if path refers to an existing directory entry that is a symbolic link.
ismount()	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	
`walk(m atc h=None, errors ='s trict')`	**Iterator over files and subdirs, recursively.** The iterator yields Path objects naming each child item of this directory and its descendants.
symlink(newlink=None)	Create a symbolic link at newlink, pointing here. default = cwd
text(encoding=None, errors='strict')	Legacy function to read text. Converts all newline sequences to \n.
touch()	Set the access/modified times of this file to the current time. Create the file if it does not exist.
unlink()	Remove (delete) the file path. `os.rem ove()`
unlink_p()	Like remove(), but does not raise an exception if the file does not exist.
classmethod using_module(module)	
utime(*args,* *kwargs)	Set the access and modified times of this file.
walkdirs(*args, **kwargs)	Iterator over subdirs, recursively.
walkfiles(*args, **kwargs)	Iterator over files, recursively.
with_suffix(suffix)	Return a new path with the file suffix changed (or added, if none)
write_bytes(bytes, append=False)	Open this file and write the given bytes to it.
write_lines(lines, encoding=None, errors='strict', linese-p=_default_linesep, append=False)	Write the given lines of text to this file.
write_text(text, encoding=None, errors='strict', linese-p=os.linesep, append=False)	Write the given text to this file.

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 Commmands	
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 symlin-ks=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.

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

class path.Path - represents filesystem path	
abspath()	Return a normalized absolutized version of the pathname path.`normpa th( joi n(o s.g etc wd(), path) )`
access(*args, **kwargs)	Return does the real user have access to this path.`os.acc ess()`
basename()	Return the base name of pathname path. This is the second element of the pair returned by passing path to the function split().
bytes()	Open this file, read all bytes, return them as a string.
cd()	`os.chd ir( path)` Change the current working directory to path.
chdir()	`os.chd ir( path)`
chown(uid=-1, gid=-1)	Change the owner and group by names or numbers.
chroot()	Change the root directory of the current process to path.
chunks(size, *args,* *kwargs)	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 size will be passed to open().
copy(dst, *, follow_symlinks=True)	Copy data and mode bits ("cp src dst"). Return the file's destination. The destination may be a directory.
copy2(dst, *, follow_symlinks=True)	Copy data and metadata. Return the file's destination. Metadata is copied with copystat(). Please see the copystat function for more information.
copyfile(dst, *, follow_symlinks=True)	Copy data from src to dst in the most efficient way possible.

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

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

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()	`os.pat h.r eal path()`
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**

cheatography.com/mad100141/

Published 19th April, 2023.
Last updated 20th April, 2023.