OS modules in python

on March 11, 2020

os.path.splitext()

This method in Python is used to split the path name into a pair root and ext. Here, ext stands for extension and has the extension portion of the specified path while root is everything except ext part.
ext is empty if specified path does not have any extension. If the specified path has leading period (‘.’), it will be ignored.

path name	root	e
/home/User/Desktop/file.txt	/home/User/Desktop/file	
/home/User/Desktop	/home/User/Desktop	
file.py	file	
.txt	.txt	

os.**name**

The name of the operating system dependent module imported. The following names have currently been registered: `'posix'`, `'nt'`, `'java'`.

os.**chdir**(*path*)

os.**mkdir**(*path*, *mode=0o777*, *, *dir_fd=None*)

Create a directory named *path* with numeric mode *mode*.
If the directory already exists, `FileExistsError` is raised.

os.**makedirs**(*name*, *mode=0o777*, *exist_ok=False*)

Recursive directory creation function. Like `mkdir()`, but makes all intermediate-level directories needed to contain the leaf directory.
If *exist_ok* is `False` (the default), an `FileExistsError` is raised if the target directory already exists.

os.**remove**(*path*, *, *dir_fd=None*)

Remove (delete) the file *path*. If *path* is a directory, an `IsADirectoryError` is raised. Use `rmdir()` to remove directories.
This function can support paths relative to directory descriptors.
On Windows, attempting to remove a file that is in use causes an exception to be raised; on Unix, the directory entry is removed but the storage allocated to the file is not made available until the original file is no longer in use.

os.**removedirs**(*name*)

Remove directories recursively. Works like `rmdir()` except that, if the leaf directory is successfully removed, `removedirs()` tries to successively remove every parent directory mentioned in *path* until an error is raised (which is ignored, because it generally means that a parent directory is not empty). For example, `os.removedirs('foo/bar/baz')` will first remove the directory `'foo/bar/baz'`, and then remove `'foo/bar'` and `'foo'` if they are empty. Raises `OSError` if the leaf directory could not be successfully removed.

os.**rename**(*src*, *dst*, *, *src_dir_fd=None*, *dst_dir_fd=None*)

Rename the file or directory *src* to *dst*. If *dst* exists, the operation will fail with an `OSError` subclass in a number of cases:
On Windows, if *dst* exists a `FileExistsError` is always raised.

`os.`**`rmdir`**`(`*`path`*`, *, `*`dir_fd=None`*`)`

Remove (delete) the directory *path*. If the directory does not exist or is not empty, an `FileNotFoundError` or an `OSError` is raised respectively. In order to remove whole directory trees, `shutil.rmtree()` can be used.
This function can support paths relative to directory descriptors.

`os.path.`**`abspath`**`(`*`path`*`)`

Return a normalized absolutized version of the pathname *path*. On most platforms, this is equivalent to calling the function `normpath()` as follows: `normpath(join(os.getcwd(), path))`.

`os.path.`**`dirname`**`(`*`path`*`)`

Return the directory name of pathname *path*. This is the first element of the pair returned by passing *path* to the function `split()`.

`os.path.`**`isfile`**`(`*`path`*`)`

Return `True` if *path* is an `existing` regular file. This follows symbolic links, so both `islink()` and `isfile()` can be true for the same path.
Changed in version 3.6: Accepts a path-like object.

`os.path.`**`isdir`**`(`*`path`*`)`

Return `True` if *path* is an `existing` directory. This follows symbolic links, so both `islink()` and `isdir()` can be true for the same path.
Changed in version 3.6: Accepts a path-like object.

`os.path.`**`join`**`(`*`path`*`, *`*`paths`*`)`

Join one or more path components intelligently. The return value is the concatenation of *path* and any members of **paths* with exactly one directory separator (`os.sep`) following each non-empty part except the last, meaning that the result will only end in a separator if the last part is empty. If a component is an absolute path, all previous components are thrown away and joining continues from the absolute path component.
On Windows, the drive letter is not reset when an absolute path component (e.g., `r'\foo'`) is encountered. If a component contains a drive letter, all previous components are thrown away and the drive letter is reset. Note that since there is a current directory for each drive, `os.path.join("c:", "foo")` represents a path relative to the current directory on drive `C:` (`c:foo`), not `c:\foo`.

Changed in version 3.6: Accepts a path-like object for *path* and *paths*.

`os.path.`**`split`**(*`path`*)

Split the pathname *path* into a pair, `(head, tail)` where *tail* is the last pathname component and *head* is everything leading up to that. The *tail* part will never contain a slash; if *path* ends in a slash, *tail* will be empty. If there is no slash in *path*, *head* will be empty. If *path* is empty, both *head* and *tail* are empty. Trailing slashes are stripped from *head* unless it is the root (one or more slashes only). In all cases, `join(head, tail)` returns a path to the same location as *path* (but the strings may differ). Also see the functions `dirname()` and `basename()`.
Changed in version 3.6: Accepts a path-like object.

Source :

https://www.geeksforgeeks.org/python-os-path-splitext-method/

https://docs.python.org/3/library/os.html

MODULES PYTHON

Enter your comment...

Translate

Select Language ▼

Powered by Blogger