The os Module All Versions

This module provides a portable way of using operating system dependent functionality.

Examples

makedirs - recursive directory creation

Given a local directory with the following contents:

```
└── dir1
    ├── subdir1
    └── subdir2
```

We want to create the same subdir1, subdir2 under a new directory dir2, which does not exist yet.

```
import os

os.makedirs("./dir2/subdir1")
os.makedirs("./dir2/subdir2")
```

Running this results in

```
├── dir1
│   ├── subdir1
│   └── subdir2
└── dir2
    ├── subdir1
    └── subdir2
```

dir2 is only created the first time it is needed, for subdir1's creation.

If we had used **os.mkdir** instead, we would have had an exception because dir2 would not have existed yet.

```
    os.mkdir("./dir2/subdir1")
OSError: [Errno 2] No such file or directory: './dir2/subdir1'
```

os.makedirs won't like it if the target directory exists already. If we re-run it again:

```
OSError: [Errno 17] File exists: './dir2/subdir1'
```

However, this could easily be fixed by catching the exception and checking that the directory has been created.

```
try:
    os.makedirs("./dir2/subdir1")
except OSError:
    if not os.path.isdir("./dir2/subdir1"):
        raise

try:
    os.makedirs("./dir2/subdir2")
except OSError:
    if not os.path.isdir("./dir2/subdir2"):
        raise
```

Change permissions on a file

```
os.chmod(path, mode)
```

where `mode` is the desired permission, in octal.

Create a directory

```
os.mkdir('newdir')
```

If you need to specify permissions, you can use the optional `mode` argument:

```
os.mkdir('newdir', mode=0700)
```

Determine the name of the operating system

The `os` module provides an interface to determine what type of operating system the code is currently running on.

```
os.name
```

This can return one of the following in Python 3:

- `posix`
- `nt`
- `ce`
- `java`

More detailed information can be retrieved from `sys.platform`

Follow a symlink (POSIX)

Sometimes you need to determine the target of a symlink. `os.readlink` will do this:

```
print(os.readlink(path_to_symlink))
```

Get current directory

Use the `os.getcwd()` function:

```
print(os.getcwd())
```

Remove a directory

Remove the directory at `path` :

```
os.rmdir(path)
```

You should not use `os.remove()` to remove a directory. That function is for *files* and using it on directories will result in an `OSError`

Syntax

```
import os
```

Parameters

Parameter	Details
Path	A path to a file. The path separator may be determined by `os.path.sep` .
Mode	The desired permission, in octal (e.g. `0700`)

Remarks