**Python os Library**

Python OS module allows us to use the operating system dependent functionalities and to interact with the underlying operating system in several different ways. For example, we can work with files, change the environment variables, and we can move files around, etc. This is as same as overriding all the os built-in functionalities in a module and using them in a file I/O and system handling.

## OS Module Common Functions

OS module provides some callable methods and some variables. Some of the common methods for different functional categories are:

1. **Manipulating directories:**

* chdir()
* getcwd()
* listdir()
* mkdir()
* makedirs()
* rmdir()
* removedirs()

1. **Removing a file:**

* remove()

1. **Renaming files/directories:**

* rename()

1. **Using more than one process**:

* system()
* popen()
* close()
* walk()

1. **User id and process id:**

* getgid(), os.getuid(), os.getpid()

1. **More about directories and files:**

* error
* stat()

1. **Cross-plateform os attributes:**

* name

1. **Accessing environment variables:**

* environ

funtions we used

**os.rename():**to rename a file or a folder.In arguments pass the original file name first and then the new name of the file.

1. **os.listdir():** It returns a list of files and the folders in the current directory.
2. **os.path** module is sub-module of OS module in Python used for common pathname manipulation.

***os.path.join()*** method in Python join one or more path components intelligently. This method concatenates various path components with exactly one directory separator (‘/’) following each non-empty part except the last path component. If the last path component to be joined is empty then a directory seperator (‘/’) is put at the end.  
If a path component represents an absolute path, then all previous components joined are discarded and joining continues from the absolute path component.

9 **os.path.exists()** method in Python is used to check whether the specified path exists or not. This method can be also used to check whether the given path refers to an open file descriptor or not.

**Note:** ***os.path.exists()*** function may return *False*, if permission is not granted to execute *os.stat()* on the requested file, even if the path exists.

**Reference:** <https://docs.python.org/3/library/os.path.html>

1. **os.remove():** It removes the path of a file. It takes path string as a variable.
2. os.**environb**
3. Bytes version of [environ](https://docs.python.org/3/library/os.html#os.environ): a [mapping](https://docs.python.org/3/glossary.html#term-mapping) object representing the environment as byte strings. [environ](https://docs.python.org/3/library/os.html#os.environ) and [environb](https://docs.python.org/3/library/os.html" \l "os.environb" \o "os.environb) are synchronized (modify [environb](https://docs.python.org/3/library/os.html" \l "os.environb" \o "os.environb) updates [environ](https://docs.python.org/3/library/os.html#os.environ), and vice versa).
4. [environb](https://docs.python.org/3/library/os.html#os.environb) is only available if [supports\_bytes\_environ](https://docs.python.org/3/library/os.html" \l "os.supports_bytes_environ" \o "os.supports_bytes_environ) is True.
5. *New in version 3.2.*
6. *Changed in version 3.9:*Updated to support [**PEP 584**](https://www.python.org/dev/peps/pep-0584)’s merge (|) and update (|=) operators.
7. os.**chdir**(*path*)
8. os.**fchdir**(*fd*)
9. os.**getcwd**()
10. These functions are described in [Files and Directories](https://docs.python.org/3/library/os.html#os-file-dir).
11. os.**fsencode**(*filename*)
12. Encode [path-like](https://docs.python.org/3/glossary.html#term-path-like-object) *filename* to the filesystem encoding with 'surrogateescape' error handler, or 'strict' on Windows; return [bytes](https://docs.python.org/3/library/stdtypes.html#bytes) unchanged.
13. [fsdecode()](https://docs.python.org/3/library/os.html#os.fsdecode) is the reverse function.
14. *New in version 3.2.*
15. *Changed in version 3.6:*Support added to accept objects implementing the [os.PathLike](https://docs.python.org/3/library/os.html" \l "os.PathLike" \o "os.PathLike) interface.
16. os.**fsdecode**(*filename*)
17. Decode the [path-like](https://docs.python.org/3/glossary.html#term-path-like-object) *filename* from the filesystem encoding with 'surrogateescape' error handler, or 'strict' on Windows; return [str](https://docs.python.org/3/library/stdtypes.html#str) unchanged.
18. [fsencode()](https://docs.python.org/3/library/os.html#os.fsencode) is the reverse function.
19. *New in version 3.2.*
20. *Changed in version 3.6:*Support added to accept objects implementing the [os.PathLike](https://docs.python.org/3/library/os.html" \l "os.PathLike" \o "os.PathLike) interface.
21. os.**fspath**(*path*)
22. Return the file system representation of the path.
23. If [str](https://docs.python.org/3/library/stdtypes.html#str) or [bytes](https://docs.python.org/3/library/stdtypes.html#bytes) is passed in, it is returned unchanged. Otherwise [\_\_fspath\_\_()](https://docs.python.org/3/library/os.html#os.PathLike.__fspath__) is called and its value is returned as long as it is a [str](https://docs.python.org/3/library/stdtypes.html#str) or [bytes](https://docs.python.org/3/library/stdtypes.html#bytes) object. In all other cases, [TypeError](https://docs.python.org/3/library/exceptions.html" \l "TypeError" \o "TypeError) is raised.
24. *New in version 3.6.*
25. *class*os.**PathLike**
26. An [abstract base class](https://docs.python.org/3/glossary.html#term-abstract-base-class) for objects representing a file system path, e.g. [pathlib.PurePath](https://docs.python.org/3/library/pathlib.html" \l "pathlib.PurePath" \o "pathlib.PurePath).

*abstractmethod***\_\_fspath\_\_**()

1. Return the file system path representation of the object.
2. The method should only return a [str](https://docs.python.org/3/library/stdtypes.html#str) or [bytes](https://docs.python.org/3/library/stdtypes.html#bytes) object, with the preference being for [str](https://docs.python.org/3/library/stdtypes.html#str).
3. os.**getenv**(*key*, *default=None*)
4. Return the value of the environment variable *key* if it exists, or *default* if it doesn’t. *key*, *default* and the result are str.
5. On Unix, keys and values are decoded with [sys.getfilesystemencoding()](https://docs.python.org/3/library/sys.html" \l "sys.getfilesystemencoding" \o "sys.getfilesystemencoding) and 'surrogateescape' error handler. Use [os.getenvb()](https://docs.python.org/3/library/os.html" \l "os.getenvb" \o "os.getenvb) if you would like to use a different encoding.
6. [Availability](https://docs.python.org/3/library/intro.html#availability): most flavors of Unix, Windows.
7. os.**getenvb**(*key*, *default=None*)
8. Return the value of the environment variable *key* if it exists, or *default* if it doesn’t. *key*, *default* and the result are bytes.
9. [getenvb()](https://docs.python.org/3/library/os.html#os.getenvb) is only available if [supports\_bytes\_environ](https://docs.python.org/3/library/os.html" \l "os.supports_bytes_environ" \o "os.supports_bytes_environ) is True.
10. [Availability](https://docs.python.org/3/library/intro.html#availability): most flavors of Unix.
11. *New in version 3.2.*
12. os.**get\_exec\_path**(*env=None*)
13. Returns the list of directories that will be searched for a named executable, similar to a shell, when launching a process. *env*, when specified, should be an environment variable dictionary to lookup the PATH in. By default, when *env* is None, [environ](https://docs.python.org/3/library/os.html#os.environ) is used.
14. *New in version 3.2.*
15. os.**getegid**()
16. Return the effective group id of the current process. This corresponds to the “set id” bit on the file being executed in the current process.
17. [Availability](https://docs.python.org/3/library/intro.html#availability): Unix.
18. os.**geteuid**()
19. Return the current process’s effective user id.
20. [Availability](https://docs.python.org/3/library/intro.html#availability): Unix.

**EXTRA.**

## **File Descriptor Operations**

These functions operate on I/O streams referenced using file descriptors.

File descriptors are small integers corresponding to a file that has been opened by the current process. For example, standard input is usually file descriptor 0, standard output is 1, and standard error is 2. Further files opened by a process will then be assigned 3, 4, 5, and so forth. The name “file descriptor” is slightly deceptive; on Unix platforms, sockets and pipes are also referenced by file descriptors.

The [fileno()](https://docs.python.org/3/library/io.html" \l "io.IOBase.fileno" \o "io.IOBase.fileno) method can be used to obtain the file descriptor associated with a [file object](https://docs.python.org/3/glossary.html#term-file-object) when required. Note that using the file descriptor directly will bypass the file object methods, ignoring aspects such as internal buffering of data.

os.**close**(fd)

Close file descriptor fd.

**Note**

This function is intended for low-level I/O and must be applied to a file descriptor as returned by [os.open()](https://docs.python.org/3/library/os.html" \l "os.open" \o "os.open) or [pipe()](https://docs.python.org/3/library/os.html#os.pipe). To close a “file object” returned by the built-in function [open()](https://docs.python.org/3/library/functions.html#open) or by [popen()](https://docs.python.org/3/library/os.html" \l "os.popen" \o "os.popen) or [fdopen()](https://docs.python.org/3/library/os.html" \l "os.fdopen" \o "os.fdopen), use its [close()](https://docs.python.org/3/library/io.html#io.IOBase.close) method.

os.**closerange**(fd\_low, fd\_high)

Close all file descriptors from fd\_low (inclusive) to fd\_high (exclusive), ignoring errors. Equivalent to (but much faster than):

**for** fd **in** range(fd\_low, fd\_high):

**try**:

os.close(fd)

**except** OSError:

**pass**

os.**copy\_file\_range**(src, dst, count, offset\_src=None, offset\_dst=None)

Copy count bytes from file descriptor src, starting from offset offset\_src, to file descriptor dst, starting from offset offset\_dst. If offset\_src is None, then src is read from the current position; respectively for offset\_dst. The files pointed by src and dst must reside in the same filesystem, otherwise an [OSError](https://docs.python.org/3/library/exceptions.html" \l "OSError" \o "OSError) is raised with [errno](https://docs.python.org/3/library/exceptions.html" \l "OSError.errno" \o "OSError.errno) set to [errno.EXDEV](https://docs.python.org/3/library/errno.html" \l "errno.EXDEV" \o "errno.EXDEV).

This copy is done without the additional cost of transferring data from the kernel to user space and then back into the kernel. Additionally, some filesystems could implement extra optimizations. The copy is done as if both files are opened as binary.

The return value is the amount of bytes copied. This could be less than the amount requested.

[Availability](https://docs.python.org/3/library/intro.html#availability): Linux kernel >= 4.5 or glibc >= 2.27.

*New in version 3.8.*

os.**device\_encoding**(fd)

Return a string describing the encoding of the device associated with fd if it is connected to a terminal; else return [None](https://docs.python.org/3/library/constants.html#None).

os.**dup**(fd)

Return a duplicate of file descriptor fd. The new file descriptor is [non-inheritable](https://docs.python.org/3/library/os.html#fd-inheritance).

On Windows, when duplicating a standard stream (0: stdin, 1: stdout, 2: stderr), the new file descriptor is [inheritable](https://docs.python.org/3/library/os.html#fd-inheritance).

*Changed in version 3.4:*The new file descriptor is now non-inheritable.

os.**dup2**(fd, fd2, inheritable=True)

Duplicate file descriptor fd to fd2, closing the latter first if necessary. Return fd2. The new file descriptor is [inheritable](https://docs.python.org/3/library/os.html#fd-inheritance) by default or non-inheritable if inheritable is False.

*Changed in version 3.4:*Add the optional inheritable parameter.

*Changed in version 3.7:*Return fd2 on success. Previously, None was always returned.

os.**fchmod**(fd, mode)

Change the mode of the file given by fd to the numeric mode. See the docs for [chmod()](https://docs.python.org/3/library/os.html" \l "os.chmod" \o "os.chmod) for possible values of mode. As of Python 3.3, this is equivalent to os.chmod(fd, mode).

Raises an [auditing event](https://docs.python.org/3/library/sys.html#auditing) os.chmod with arguments path, mode, dir\_fd.

[Availability](https://docs.python.org/3/library/intro.html#availability): Unix.

os.**fchown**(fd, uid, gid)

Change the owner and group id of the file given by fd to the numeric uid and gid. To leave one of the ids unchanged, set it to -1. See [chown()](https://docs.python.org/3/library/os.html" \l "os.chown" \o "os.chown). As of Python 3.3, this is equivalent to os.chown(fd, uid, gid).

Raises an [auditing event](https://docs.python.org/3/library/sys.html#auditing) os.chown with arguments path, uid, gid, dir\_fd.

[Availability](https://docs.python.org/3/library/intro.html#availability): Unix.

os.**fdatasync**(fd)

Force write of file with filedescriptor fd to disk. Does not force update of metadata.

[Availability](https://docs.python.org/3/library/intro.html#availability): Unix.

**Note**

This function is not available on MacOS.

os.**fpathconf**(fd, name)

Return system configuration information relevant to an open file. name specifies the configuration value to retrieve; it may be a string which is the name of a defined system value; these names are specified in a number of standards (POSIX.1, Unix 95, Unix 98, and others). Some platforms define additional names as well. The names known to the host operating system are given in the pathconf\_names dictionary. For configuration variables not included in that mapping, passing an integer for name is also accepted.

If name is a string and is not known, [ValueError](https://docs.python.org/3/library/exceptions.html" \l "ValueError" \o "ValueError) is raised. If a specific value for name is not supported by the host system, even if it is included in pathconf\_names, an [OSError](https://docs.python.org/3/library/exceptions.html" \l "OSError" \o "OSError) is raised with [errno.EINVAL](https://docs.python.org/3/library/errno.html" \l "errno.EINVAL" \o "errno.EINVAL) for the error number.

As of Python 3.3, this is equivalent to os.pathconf(fd, name).

[Availability](https://docs.python.org/3/library/intro.html#availability): Unix.

os.**fstat**(fd)

Get the status of the file descriptor fd. Return a [stat\_result](https://docs.python.org/3/library/os.html" \l "os.stat_result" \o "os.stat_result) object.

As of Python 3.3, this is equivalent to os.stat(fd).

**See also**

The [stat()](https://docs.python.org/3/library/os.html#os.stat) function.

os.**fstatvfs**(fd)

Return information about the filesystem containing the file associated with file descriptor fd, like [statvfs()](https://docs.python.org/3/library/os.html" \l "os.statvfs" \o "os.statvfs). As of Python 3.3, this is equivalent to os.statvfs(fd).

[Availability](https://docs.python.org/3/library/intro.html#availability): Unix.

os.**fsync**(fd)

Force write of file with filedescriptor fd to disk. On Unix, this calls the native fsync() function; on Windows, the MS \_commit() function.

If you’re starting with a buffered Python [file object](https://docs.python.org/3/glossary.html#term-file-object) f, first do f.flush(), and then do os.fsync(f.fileno()), to ensure that all internal buffers associated with f are written to disk.

[Availability](https://docs.python.org/3/library/intro.html#availability): Unix, Windows.

os.**ftruncate**(fd, length)

Truncate the file corresponding to file descriptor fd, so that it is at most length bytes in size. As of Python 3.3, this is equivalent to os.truncate(fd, length).

Raises an [auditing event](https://docs.python.org/3/library/sys.html#auditing) os.truncate with arguments fd, length.

[Availability](https://docs.python.org/3/library/intro.html#availability): Unix, Windows.

*Changed in version 3.5:*Added support for Windows

os.**get\_blocking**(fd)

Get the blocking mode of the file descriptor: False if the [O\_NONBLOCK](https://docs.python.org/3/library/os.html#os.O_NONBLOCK) flag is set, True if the flag is cleared.

See also [set\_blocking()](https://docs.python.org/3/library/os.html" \l "os.set_blocking" \o "os.set_blocking) and [socket.socket.setblocking()](https://docs.python.org/3/library/socket.html" \l "socket.socket.setblocking" \o "socket.socket.setblocking).

[Availability](https://docs.python.org/3/library/intro.html#availability): Unix.

*New in version 3.5.*

os.**isatty**(fd)

Return True if the file descriptor fd is open and connected to a tty(-like) device, else False.

os.**lockf**(fd, cmd, len)

Apply, test or remove a POSIX lock on an open file descriptor. fd is an open file descriptor. cmd specifies the command to use - one of [F\_LOCK](https://docs.python.org/3/library/os.html#os.F_LOCK), [F\_TLOCK](https://docs.python.org/3/library/os.html#os.F_TLOCK), [F\_ULOCK](https://docs.python.org/3/library/os.html#os.F_ULOCK) or [F\_TEST](https://docs.python.org/3/library/os.html#os.F_TEST). len specifies the section of the file to lock.

Raises an [auditing event](https://docs.python.org/3/library/sys.html#auditing) os.lockf with arguments fd, cmd, len.

[Availability](https://docs.python.org/3/library/intro.html#availability): Unix.

*New in version 3.3.*