

Sr.No.	Methods with Description
1	os.access(path, mode) ↗ Use the real uid/gid to test for access to path.
2	os.chdir(path) ↗ Change the current working directory to path
3	os.chflags(path, flags) ↗ Set the flags of path to the numeric flags.
4	os.chmod(path, mode) ↗ Change the mode of path to the numeric mode.
5	os.chown(path, uid, gid) ↗ Change the owner and group id of path to the numeric uid and gid.
6	os.chroot(path) ↗ Change the root directory of the current process to path.
7	os.close(fd) ↗ Close file descriptor fd.
8	os.closerange(fd_low, fd_high) ↗ Close all file descriptors from fd_low (inclusive) to fd_high (exclusive), ignoring errors.
9	os.dup(fd) ↗ Return a duplicate of file descriptor fd.
10	os.dup2(fd, fd2) ↗ Duplicate file descriptor fd to fd2, closing the latter first if necessary.
11	os.fchdir(fd) ↗ Change the current working directory to the directory represented by the file descriptor fd.
12	os.fchmod(fd, mode) ↗ Change the mode of the file given by fd to the numeric mode.
13	os.fchown(fd, uid, gid) ↗ Change the owner and group id of the file given by fd to the numeric uid and gid.
14	os.fdatasync(fd) ↗ Force write of file with filedescriptor fd to disk.
15	os.fdopen(fd[, mode[, bufsize]]) ↗ Return an open file object connected to the file descriptor fd.
16	os.fpathconf(fd, name) ↗ Return system configuration information relevant to an open file. name specifies the configuration value to retrieve.
17	os.fstat(fd) ↗ Return status for file descriptor fd, like stat().
18	os.fstatvfs(fd) ↗ Return information about the filesystem containing the file associated with file descriptor fd, like statvfs().
19	os.fsync(fd) ↗ Force write of file with filedescriptor fd to disk.
20	os.ftruncate(fd, length) ↗ Truncate the file corresponding to file descriptor fd, so that it is at most length bytes in size.
21	os.getcwd() ↗ Return a string representing the current working directory.
22	os.getcwdu() ↗ Return a Unicode object representing the current working directory.
23	os.isatty(fd) ↗ Return True if the file descriptor fd is open and connected to a tty(-like) device, else False.
24	os.lchflags(path, flags) ↗ Set the flags of path to the numeric flags, like chflags(), but do not follow symbolic links.
25	os.lchmod(path, mode) ↗ Change the mode of path to the numeric mode.
26	os.lchown(path, uid, gid) ↗ Change the owner and group id of path to the numeric uid and gid. This function will not follow symbolic links.
27	os.link(src, dst) ↗ Create a hard link pointing to src named dst.
28	os.listdir(path) ↗ Return a list containing the names of the entries in the directory given by path.
29	os.lseek(fd, pos, how) ↗ Set the current position of file descriptor fd to position pos, modified by how.
30	os.lstat(path) ↗ Like stat(), but do not follow symbolic links.
31	os.major(device) ↗ Extract the device major number from a raw device number.
32	os.makedev(major, minor) ↗ Compose a raw device number from the major and minor device numbers.
33	os.makedirs(path[, mode]) ↗ Recursive directory creation function.
34	os.minor(device) ↗ Extract the device minor number from a raw device number.
35	os.mkdir(path[, mode]) ↗ Create a directory named path with numeric mode mode.
36	os.mkfifo(path[, mode]) ↗ Create a FIFO (a named pipe) named path with numeric mode mode. The default mode is 0666 (octal).
37	os.mknod(filename[, mode=0600, device]) ↗ Create a filesystem node (file, device special file or named pipe) named filename.
38	os.open(file, flags[, mode]) ↗ Open the file file and set various flags according to flags and possibly its mode according to mode.
39	os.openpty() ↗ Open a new pseudo-terminal pair. Return a pair of file descriptors (master, slave) for the pty and the tty, respectively.
40	os.pathconf(path, name) ↗ Return system configuration information relevant to a named file.
41	os.pipe() ↗ Create a pipe. Return a pair of file descriptors (r, w) usable for reading and writing, respectively.
42	os.popen(command[, mode[, bufsize]]) ↗ Open a pipe to or from command.
43	os.read(fd, n) ↗ Read at most n bytes from file descriptor fd. Return a string containing the bytes read. If the end of the file referred to by fd has been reached, an empty string is returned.
44	os.readlink(path) ↗ Return a string representing the path to which the symbolic link points.
45	os.remove(path) ↗ Remove the file path.
46	os.removedirs(path) ↗ Remove directories recursively.
47	os.rename(src, dst) ↗ Rename the file or directory src to dst.
48	os.renames(old, new) ↗ Recursive directory or file renaming function.
49	os.rmdir(path) ↗ Remove the directory path
50	os.stat(path) ↗ Perform a stat system call on the given path.
51	os.stat_float_times([newvalue]) ↗ Determine whether stat_result represents time stamps as float objects.
52	os.statvfs(path) ↗ Perform a statvfs system call on the given path.
53	os.symlink(src, dst) ↗ Create a symbolic link pointing to src named dst.
54	os.tcgetpgrp(fd) ↗ Return the process group associated with the terminal given by fd (an open file descriptor as returned by open()).
55	os.tcsetpgrp(fd, pg) ↗ Set the process group associated with the terminal given by fd (an open file descriptor as returned by open()) to pg.
56	os.tmpnam([dir[, prefix]]) ↗ Return a unique path name that is reasonable for creating a temporary file.
57	os.tmpfile() ↗ Return a new file object opened in update mode (w+b).
58	os.tmpnam() ↗ Return a unique path name that is reasonable for creating a temporary file.
59	os.ttyname(fd) ↗ Return a string which specifies the terminal device associated with file descriptor fd. If fd is not associated with a terminal device, an exception is raised.
60	os.unlink(path) ↗ Remove the file path.
61	os.utime(path, times) ↗ Set the access and modified times of the file specified by path.
62	os.walk(top[, topdown=True[, onerror=None[, followlinks=False]]]) ↗ Generate the file names in a directory tree by walking the tree either top-down or bottom-up.
63	os.write(fd, str) ↗ Write the string str to file descriptor fd. Return the number of bytes actually written.