Python Tutorials +
Python for Data Science +
Python Career Guides +
Python Projects New +
Python Interview Questi... +
Python Quiz +

List of 63 Python os Module with Syntax & Examples (Latest)

Free Python course with 35 real-time projects [Start Now!!]

In this tutorial on Python os Module, we will get closer to the os module and its methods. Moreover, we will study syntax and examples of os Module in Python Programming Language.

So, let's start Python Module Tutorial

List of 63 Python os Module with Syntax & Examples (Latest)

What is Python OS Module?

The Python OS module lets us work with files and directories. We have been using it a lot to get to the Desktop in our examples. But it is much more.

Let's discuss the important functions/methods it offers. In case of any doubt, please as us in comments.

Let's check the dir() on this module?

```
>>> dir(os)
```

Output

```
>['DirEntry', 'F_OK', 'MutableMapping', 'O_APPEND', 'O_BINARY', 'O_CREAT', 'O_EXCL', 'O_NOINHERIT', 'O_RANDOM', 'O_RDONLY',
'O_RDWR', 'O_SEQUENTIAL', 'O_SHORT_LIVED', 'O_TEMPORARY', 'O_TEXT', 'O_TRUNC', 'O_WRONLY', 'P_DETACH', 'P_NOWAIT',
'P_NOWAITO', 'P_OVERLAY', 'P_WAIT', 'PathLike', 'R_OK', 'SEEK_CUR', 'SEEK_END', 'SEEK_SET', 'TMP_MAX', 'W_OK', 'X_OK',
'_Environ', '__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__',
'__spec__', '_execvpe', '_exists', '_exit', '_fspath', '_get_exports_list', '_putenv', '_unsetenv', '_wrap_close', 'abc',
'abort', 'access', 'altsep', 'chdir', 'chmod', 'close', 'closerange', 'cpu_count', 'curdir', 'defpath', 'device_encoding',
'devnull', 'dup', 'dup2', 'environ', 'errno', 'error', 'execl', 'execle', 'execlp', 'execlpe', 'execv', 'execve', 'execvp',
'execvpe', 'extsep', 'fdopen', 'fsdecode', 'fsencode', 'fspath', 'fstat', 'fsync', 'ftruncate', 'get_exec_path',
'get_handle_inheritable', 'get_inheritable', 'get_terminal_size', 'getcwd', 'getcwdb', 'getenv', 'getlogin', 'getpid',
'getppid', 'isatty', 'kill', 'linesep', 'link', 'listdir', 'lseek', 'lstat', 'makedirs', 'mkdir', 'name', 'open', 'pardir',
'path', 'pathsep', 'pipe', 'popen', 'putenv', 'read', 'readlink', 'remove', 'removedirs', 'rename', 'renames', 'replace',
'rmdir', 'scandir', 'sep', 'set_handle_inheritable', 'set_inheritable', 'spawnl', 'spawnle', 'spawnv', 'spawnve', 'st',
'startfile', 'stat', 'stat_float_times', 'stat_result', 'statvfs_result', 'strerror', 'supports_bytes_environ',
'supports_dir_fd', 'supports_effective_ids', 'supports_fd', 'supports_follow_symlinks', 'symlink', 'sys', 'system',
'terminal_size', 'times', 'times_result', 'truncate', 'umask', 'uname_result', 'unlink', 'urandom', 'utime', 'waitpid',
'walk', 'write']
```

1. access(path,mode)

This method uses the real uid/gid to test for access to a path. If access is allowed, it returns True.

Else, it returns False. The first argument is the path; the second is the mode.

The mode can take one of four values:

1. os.F_OK — Found
2. os.R_OK — Readable
3. os.W_OK — Writable
4. os.X_OK — Executable

Now, let's take an example.

```
>>> os.chdir('C:\\Users\\lifei\\Desktop')
>>> os.access('Today.txt',os.R_OK)
```

Output

```
True
```

```
>>> os.access('Today.txt',os.F_OK)
```

Output

```
True
```

```
>>> os.access('Today.txt',os.W_OK)
```

Output

```
True
```

```
>>> os.access('Today.txt',os.X_OK)
```

Output

```
True
```

2. chdir(path)

This Python os module changes the current working directory to the path we specify.

Does this need another example?

```
>>> os.chdir('C:\\Users\\lifei\\Desktop')
```

Output

```
It returns None.
```

3. chflags(path,flags)

chflags() sets path flags to the numeric flags. These flags may take a combination(bitwise OR) of the following values:

- os.UF_NODUMP – Don't dump the file
- os.UF_IMMUTABLE – You may not change the file
- os.UF_APPEND – You may only append to the file
- os.UF_NOUNLINK – You may not rename or delete the file
- os.UF_OPAQUE – The directory is opaque when we view it through a union stack
- os.SF_ARCHIVED – You may archive the file
- os.SF_IMMUTABLE – You may not change the file
- os.SF_APPEND – You may only append to the file
- os.SF_NOUNLINK – You may not rename or delete the file
- os.SF_SNAPSHOT – It is a snapshot file

Most flags are such that only the super-user can change them. Also, some flags don't work on all systems.

Sample usage:

```
>>> os.chflags('Today.txt',os.SF_NOUNLINK)
```

4. chmod(path,mode)

This Python os Module alters the mode of the path to the passed numeric mode.

The mode may be on of the following values(or a bitwise OR combination of them):

- stat.S_ISUID – Set user ID on execution
- stat.S_ISGID – Set group ID on execution
- stat.S_ENFMT – Enforced record locking
- stat.S_ISVTX – After execution, save text image
- stat.S_IREAD – Read by owner
- stat.S_IWRITE – Write by owner
- stat.S_IEXEC – Execute by owner
- stat.S_IRWXU – Read, write, and execute by owner
- stat.S_IRUSR – Read by owner
- stat.S_IWUSR – Write by owner
- stat.S_IXUSR – Execute by owner
- stat.S_IRWXG – Read, write, and execute by group
- stat.S_IRGRP – Read by group
- stat.S_IWGRP – Write by group
- stat.S_IXGRP – Execute by group
- stat.S_IRWXO – Read, write, and execute by others
- stat.S_IROTH – Read by others
- stat.S_IWOTH – Write by others
- stat.S_IXOTH – Execute by others

Sample usage:

```
>>> import stat
>>> os.chmod('Today.txt',stat.S_ISVTX)
```

This method does not return any value.

5. chroot(path)

chroot Python os Module alters the current process' root directory to the given path.

To use this, we need super-user privileges.

Sample usage:

```
>>> os.chroot("/Photos")
```

This method returns no value.

6. close(fd)

This Python os module closes the associated file with descriptor fd.

```
>>> fd=os.open('Today.txt',os.O_RDWR)
>>> os.close(fd)
```

It does not return any value.

7. closerange(fd_low,fd_high)

closerange() closes all file descriptors from fd_low to fd_high. Here, fd_low is inclusive, and fd_high is exclusive.

Here, fd_low is the lowest file descriptor to be closed, while fd_high is the highest. This method ignores errors.

Sample usage:

```
>>> fd = os.open( "Today.txt", os.O_RDWR)
>>> os.write(fd, "Testing")
>>> os.closerange( fd, fd)
```

This method does not return any value.

8. dup(fd)

Python os Module dup(fd) returns a duplicate of the file descriptor fd.

Sample usage:

```
>>> fd = os.open( "Today.txt", os.O_RDWR)
>>> d_fd = os.dup( fd )
>>> os.write(d_fd, "Testing")
>>> os.closerange( fd, d_fd)
```

This method returns a duplicate of the file descriptor.

9. dup2(fd,fd2)

dup2() duplicates the descriptor fd to fd2. And if necessary, it closes fd2 first.

The interpreter assigns the new file description only when it is available.

Sample usage:

```
>>> fd = os.open( "Today.txt", os.O_RDWR)
>>> os.write(fd, "Testing")
>>> fd2 = 1000
>>> os.dup2(fd, fd2)
>>> os.lseek(fd2, 0, 0)
>>> str = os.read(fd2, 100)
>>> print(f"Read String is {str}")
>>> os.close( fd )
```

This method returns a duplicate of the file descriptor.

10. fchdir(fd)

fchdir() alters the current working directory to the directory that the file descriptor fd represents.

For this, it is mandatory that the descriptor must refer to an opened directory, and not to an open file.

Sample usage:

```
>>> os.chdir("/var/www/html" )
>>> print "Current working dir : %s" % os.getcwd()
>>> fd = os.open( "/tmp", os.O_RDONLY )
>>> os.fchdir(fd)
>>> print "Current working dir : %s" % os.getcwd()
>>> os.close( fd )
```

This method doesn't return any value.

11. fchmod(fd,mode)

This Python os Module alters the file mode of the file, specified by fd, to the numeric mode.

The mode may be one of the following (or an ORed combination of):

- stat.S_ISUID – Set user ID on execution
- stat.S_ISGID – Set group ID on execution
- stat.S_ENFMT – Record locking enforced
- stat.S_ISVTX – Save text image after execution
- stat.S_IREAD – Read by owner
- stat.S_IWRITE – Write by owner
- stat.S_IEXEC – Execute by owner
- stat.S_IRWXU – Read, write, and execute by owner
- stat.S_IRUSR – Read by owner
- stat.S_IWUSR – Write by owner
- stat.S_IXUSR – Execute by owner
- stat.S_IRWXG – Read, write, and execute by group
- stat.S_IRGRP – Read by group
- stat.S_IWGRP – Write by group
- stat.S_IXGRP – Execute by group
- stat.S_IRWXO – Read, write, and execute by others
- stat.S_IROTH – Read by others
- stat.S_IWOTH – Write by others
- stat.S_IXOTH – Execute by others

Sample usage:

```
>>> fd = os.open( "/tmp", os.O_RDONLY )
>>> os.fchmod( fd, stat.S_IXGRP)
>>> os.fchmod(fd, stat.S_IWOTH)
>>> print "Changed mode successfully!!"
>>> os.close( fd )
```

This method doesn't return any value.

12. fchown(fd,uid,gid)

fchown() alters the owner and the group id of the file specified by fd to the numeric uid and gid.

Setting an id to -1 leaves it unchanged.

Sample usage:

```
>>> fd = os.open( "/tmp", os.O_RDONLY )
>>> os.fchown( fd, 100, -1)
>>> os.fchown( fd, -1, 50)
>>> print "Changed ownership successfully!!"
>>> os.close( fd )
```

This method doesn't return any value.

13. fdatasync(fd)

fdatasync() forces writing the file with filedescriptor fd to disk. This, however, doesn’t force update on metadata.

You can do this to flush your buffer.

Sample usage:

```
>>> fd = os.open( "Today.txt", os.O_RDWR)
>>> os.write(fd, "Testing")
>>> os.fdatasync(fd)
>>> os.lseek(fd, 0, 0)
>>> str = os.read(fd, 100)
>>> print(f"Read String is {str}")
>>> os.close( fd )
```

This method doesn’t return any value.

14. fdopen(fd[, mode[, bufsize]])

fdopen(), Python os Module returns an open file object. This object is connected to the descriptor fd.

Once you do this, you can perform all defined functions on the file object.

Sample usage:

```
>>> fd = os.open( "Today.txt", os.O_RDWR)
>>> fo = os.fdopen(fd, "w+")
>>> print (f"Current I/O pointer position {fo.tell()}")
>>> fo.write( "Python is a great language.\nYeah its great!!\n");
>>> os.lseek(fd, 0, 0)
>>> str = os.read(fd, 100)
>>> print (f"Read String is {str}")
>>> print (f"Current I/O pointer position {fo.tell()}")
>>> fo.close()
```

fdopen() returns an open file object that is connected to the file descriptor.

15. fpathconf(fd, name)

fpathconf() returns system configuration information that is relevant to an open file.

This is quite similar to the unix system call fpathconf(). It also accepts similar arguments.

Sample usage:

```
>>> fd = os.open( "Today.txt", os.O_RDWR)
>>> print (f"{os.pathconf_names}")
>>> no = os.fpathconf(fd, 'PC_LINK_MAX')
>>> print (f"Maximum number of links to the file: {no}")
>>> no = os.fpathconf(fd, 'PC_NAME_MAX')
>>> print (f"Maximum length of a filename :{no}")
>>> os.close( fd)
```

fpathconf() returns system configuration that is relevant to an open file.

16. fstat(fd)

Python os Module fstat() returns information about the file pertaining to the fd.

Let’s take a look at the structure fstat() returns:

- st_dev – ID of device containing file
- st_ino – inode number
- st_mode – protection
- st_nlink – number of hard links
- st_uid – user ID of owner
- st_gid – group ID of owner
- st_rdev – device ID (if special file)
- st_size – total size, in bytes
- st_blksize – blocksize for filesystem I/O
- st_blocks – number of blocks allocated
- st_atime – time of last access

- st_mtime – time of last modification
- st_ctime – time of last status change

Sample usage:

```
>>> fd = os.open( "Today.txt", os.O_RDWR)
>>> info = os.fstat(fd)
>>> print (f"File Info: {info}")
>>> print (f"UID of the file: {info.st_uid}")
>>> print (f"GID of the file: {info.st_gid}")
>>> os.close( fd)
```

fstat() returns information about the file linked with the fd.

17. fstatvfs(fd)

This Python os module returns information pertaining to the file system containing the file linked with file descriptor fd.

This is the structure it returns:

- f_bsize – file system block size
- f_frsize – fragment size
- f_blocks – size of fs in f_frsize units
- f_bfree – free blocks
- f_bavail – free blocks for non-root
- f_files – inodes
- f_ffree – free inodes
- f_favail – free inodes for non-root
- f_fsid – file system ID
- f_flag – mount flags
- f_namemax – maximum filename length

Sample usage:

```
>>> fd = os.open( "Today.txt", os.O_RDWR)
>>> info = os.fstatvfs(fd)
>>> print(f"File Info: {info}")
>>> print(f"Maximum filename length: {info.f_namemax}")
>>> print (f"Free blocks: {info.f_bfree}")
>>> os.close( fd)
```

fstatvfs() returns information about the file system containing the file linked.

18. fsync(fd)

This Python os Module forces write on the file liknked to the descriptor fd to disk.

Beginning with a Python file object f, first execute f.flush(), then perform os.fsync(f.fileno()).

Do this to ensure all internal buffers linked to f are written to the disk.

Sample usage:

```
>>> fd = os.open( "Today.txt", os.O_RDWR)
>>> os.write(fd, "Testing")
>>> os.fsync(fd)
>>> os.lseek(fd, 0, 0)
>>> str = os.read(fd, 100)
>>> print("Read String is: {str} ")
>>> os.close( fd )
```

fsync() doesn’t return any value.

19. ftruncate(fd,length)

ftruncate() truncates the file linked to the descriptor fd, so it holds at most length bytes in size.

Sample usage:

```
>>> fd = os.open( "Today.txt", os.O_RDWR)
>>> os.write(fd, "Testing")
>>> os.ftruncate(fd, 10)
>>> os.lseek(fd, 0, 0)
>>> str = os.read(fd, 100)
>>> print("Read String is: {str}”)
>>> os.close( fd )
```

ftruncate() doesn’t return any value.

20. getcwd()

getcwd() Python os Module returns the current working directory of a process.

Sample usage:

```
>>> os.getcwd()
```

Output

```
‘C:\\Users\\lifei\\Desktop’
```

21. getcwdu()

getcwdu() returns a unicode object that represents the current working directory.

Sample usage:

```
>>> os.chdir("/var/www/html" )
>>> print(f"Current working dir: {os.getcwdu()}")
>>> fd = os.open( "/tmp", os.O_RDONLY )
>>> os.fchdir(fd)
>>> print(f"Current working dir: {os.getcwdu()}”)
>>> os.close( fd )
```

22. isatty(fd)

isatty()returns True if the descriptor fd is open, and is connected to a tty(-like) device. Otherwise, it returns False.

Sample usage:

```
>>> fd = os.open( "Today.txt", os.O_RDWR)
>>> os.write(fd, "Testing")
>>> ret = os.isatty(fd)
>>> print(f"Returned value is: {ret}")
>>> os.close( fd )
```

23. lchflags(path,flags)

This Python os Module sets path flags to the numeric flags. Unlike chflags(), ut doesn’t follow symbolic links.

The flags may be one of the following values, or a bitwise OR combination of:

- UF_NODUMP – Do not dump the file
- UF_IMMUTABLE – The file may not be changed
- UF_APPEND – The file may only be appended to
- UF_NOUNLINK – The file may not be renamed or deleted
- UF_OPAQUE – The directory is opaque when viewed through a union stack
- SF_ARCHIVED – The file may be archived
- SF_IMMUTABLE – The file may not be changed
- SF_APPEND – The file may only be appended to
- SF_NOUNLINK – The file may not be renamed or deleted
- SF_SNAPSHOT – The file is a snapshot file.

Sample usage:

```
>>> path = "/var/www/html/Today.txt"
>>> fd = os.open( path, os.O_RDWR)
>>> os.close( fd )
>>> ret = os.lchflags(path, os.UF_IMMUTABLE )
```

lchflags() doesn’t return a value.

24. lchmod(path,mode)

lchmod() Python os Module ters the path mode to the numeric mode. If the path is a symlink, it affects the symlink, not the target.

The mode may be one of the following values, or a bitwise OR combination of:

- stat.S_ISUID – Set user ID on execution
- stat.S_ISGID – Set group ID on execution
- stat.S_ENFMT – Record locking enforced
- stat.S_ISVTX – Save text image after execution
- stat.S_IREAD – Read by owner
- stat.S_IWRITE – Write by owner
- stat.S_IEXEC – Execute by owner
- stat.S_IRWXU – Read, write, and execute by owner
- stat.S_IRUSR – Read by owner
- stat.S_IWUSR – Write by owner
- stat.S_IXUSR – Execute by owner
- stat.S_IRWXG – Read, write, and execute by group
- stat.S_IRGRP – Read by group
- stat.S_IWGRP – Write by group
- stat.S_IXGRP – Execute by group
- stat.S_IRWXO – Read, write, and execute by others
- stat.S_IROTH – Read by others
- stat.S_IWOTH – Write by others
- stat.S_IXOTH – Execute by others

Sample usage:

```
>>> path = "/var/www/html/Today.txt"
>>> fd = os.open( path, os.O_RDWR )
>>> os.close( fd )
>>> os.lchmod( path, stat.S_IXGRP)
>>> os.lchmod("/tmp/Today.txt", stat.S_IWOTH)
```

lchmod() doesn’t return any value.

25. lchown(path,uid,gid)

Python os Module lchown() alters the owner and group id of path to the numeric uid and gid.

It doensn’t follow symbolic links. Setting an id to -1 leaves it unchanged.

Sample usage:

```
>>> path = "/var/www/html/Today.txt"
>>> fd = os.open( path, os.O_RDWR)
>>> os.close( fd )
>>> os.lchown( path, 500, -1)
>>> os.lchown( path, -1, 500)
```

lchown() doesn’t return any value.

26. link(src,dst)

link() will create a hard link that points to an src named dst.

You can do this when you want to create a copy of an existing file.

Sample usage:

```
>>> path = "/var/www/html/Today.txt"
>>> fd = os.open( path, os.O_RDWR )
>>> os.close( fd )
>>> dst = "/tmp/Today.txt"
>>> os.link( path, dst)
```

lilnk() doesn't return any value.

27. listdir(path)

listdir() will return a list holding the names of the entries in the directory at the path.

This list is in an arbitrary order, and it exclude special entries '.' and '..', even if they exist in the directory.

Sample usage:

```
>>> path = "/var/www/html/"
>>> dirs = os.listdir( path )
>>> for file in dirs:
print(file)
```

28. lseek(fd,pos,how)

lseek() will set the current position of the descriptor fd to the specified position pos. 'how' modifies it.

Sample usage:

```
>>> fd = os.open( "Today.txt", os.O_RDWR)
>>> os.write(fd, "This is test")
>>> os.fsync(fd)
>>> os.lseek(fd, 0, 0)
>>> str = os.read(fd, 100)
>>> print(f"Read String is: {str}")
>>> os.close( fd )
```

lseek() doesn't return any value.

29. lstat(path)

Like fstat(), lstat() returns information about a file, but does not follow symbolic links.

lstat is an alias for fstat() on those platforms that do not support symbolic links, for instance, Windows.

It returns the following structure:

- st_dev – ID of device containing file
- st_ino – inode number
- st_mode – protection
- st_nlink – number of hard links
- st_uid – user ID of owner
- st_gid – group ID of owner
- st_rdev – device ID (if special file)
- st_size – total size, in bytes
- st_blksize – blocksize for filesystem I/O
- st_blocks – number of blocks allocated
- st_atime – time of last access
- st_mtime – time of last modification
- st_ctime – time of last status change

Sample usage:

```
>>> path = "/var/www/html/Today.txt"
>>> fd = os.open( path, os.O_RDWR)
>>> os.close( fd )
>>> info = os.lstat(path)
>>> print(f"File Info: {info}")
>>> print(f"UID of the file: {info.st_uid}")
>>> print(f"GID of the file: {info.st_gid}")
```

30. major(device)

major() takes a raw device number, and extracts the device major number (usually the st_dev or st_rdev field from stat).

Sample usage:

```
>>> path = "/var/www/html/Today.txt"
>>> info = os.lstat(path)
>>> major_dnum = os.major(info.st_dev)
>>> minor_dnum = os.minor(info.st_dev)
>>> print(f"Major Device Number: {major_dnum}")
>>> print(f"Minor Device Number: {minor_dnum}")
```

major() returns the device major number.

31. makedev(major,minor)

This Python os Module takes the minor and major device numbers, and creates a raw device number.

Sample usage:

```
>>> path = "/var/www/html/Today.txt"
>>> info = os.lstat(path)
>>> major_dnum = os.major(info.st_dev)
>>> minor_dnum = os.minor(info.st_dev)
>>> print(f"Major Device Number: {major_dnum}")
>>> print(f"Minor Device Number: {minor_dnum}")
>>> dev_num = os.makedev(major_dnum, minor_dnum)
>>> print(f"Device Number: {dev_num}")
```

makedev() returns the device number.

32. makedirs(path[, mode])

makedirs() creates a directory recursively. This way, it is like mkdir().

However, it mandates that all intermediate-level directories contain the leaf directory.

Sample usage:

```
>>> path = "/tmp/home/monthly/daily"
>>> os.makedirs( path, 0755 )
```

33. minor(device)

Python os Module minor() will take a raw device number, and extract the device's minor (usually the st_dev or st_rdev field from stat).

Sample usage:

```
>>> path = "/var/www/html/Today.txt"
>>> info = os.lstat(path)
>>> major_dnum = os.major(info.st_dev)
>>> minor_dnum = os.minor(info.st_dev)
>>> print(f"Major Device Number: {major_dnum}")
>>> print(f"Minor Device Number: {minor_dnum}")
```

minor() returns the device's minor number.

34. mkdir(path[, mode])

mkdir() Python os Module creates a directory 'path' with the numeric mode 'mode'. Some systems ignore mode.

But where used, it masks out the current umask value first.
Default mode=0777 (octal).

Sample usage:

```
>>> path = "/tmp/home/monthly/daily/hourly"
>>> os.mkdir( path, 0755 )
```

mkdir() doesn't return any value.

35. mkfifo(path[, mode])

mkfifo() creates a FIFO named 'path' with the specified numeric mode. It masks out the current umask value first. Default mode=0666 (octal).

Sample usage:

```
>>> path = "/tmp/hourly"
>>> os.mkfifo( path, 0644 )
```

mkfifo() doesn't return any value.

36. mknod(filename[, mode=0600, device])

This Python os Module will create a filesystem node named 'filename'. This can be a file, a device-special file, or a named pipe.

Sample usage:

```
>>> filename = '/tmp/tmpfile'
>>> mode = 0600|stat.S_IRUSR
>>> os.mknod(filename, mode)
```

mknod() doesn't return any value.

37. open(file, flags[, mode])

open() will open the file 'file', and will set flags based on the specified flags.

It possibly sets its mode according to the specified mode. It also masks out the current umask value first.

Default mode=0777 (octal).

The flags may take one of these values, or a bitwise-OR combination of these:

- os.O_RDONLY – open for reading only
- os.O_WRONLY – open for writing only
- os.O_RDWR – open for reading and writing
- os.O_NONBLOCK – do not block on open
- os.O_APPEND – append on each write
- os.O_CREAT – create file if it does not exist
- os.O_TRUNC – truncate size to 0
- os.O_EXCL – error if create and file exists
- os.O_SHLOCK – atomically obtain a shared lock
- os.O_EXLOCK – atomically obtain an exclusive lock
- os.O_DIRECT – eliminate or reduce cache effects
- os.O_FSYNC – synchronous writes
- os.O_NOFOLLOW – do not follow symlinks

Sample usage:

```
>>> fd = os.open( "Today.txt", os.O_RDWR)
>>> os.write(fd, "This is test")
>>> os.close( fd )
```

open() returns the descriptor for the file we opened.

38. openpty()

Python os Module openpty() opens a pseudo-terminal pair.

Then, it returns a pair of descriptors- master & slave- for the pty & the tty, respectively.

Sample usage:

```
>>> m,s = os.openpty()
>>> print(m)
>>> print(s)
>>> s = os.ttyname(s)
>>> print(m)
>>> print(s)
```

39. pathconf(path,name)

Python os Module pathconf() returns system configuration information pertaining to a named file.

Sample usage:

```
>>> print(f"{os.pathconf_names}" )
>>> no = os.pathconf('a2.py', 'PC_NAME_MAX')
>>> print(f"Maximum length of a filename: {no}")
>>> no = os.pathconf('a2.py', 'PC_FILESIZEBITS')
>>> print(f"file size in bits: {no}")
```

40. pipe()

pipe() creates a pipe. Then, it returns a pair of descriptors- r & w- for reading and writing.

Sample usage:

```
>>> os.pipe()
```

Output

```
(3, 4)
```

41. popen(command[, mode[, bufsize]])

This Python os Module popen() will open a pipe to, or from, the command specified .It returns an open file object that is connected to the pipe.

We can read or write to this object depending on whether the mode is 'r' (default) or 'w'. The bufsize argument means the same as in the open() function.

Sample usage:

```
>>> a = 'mkdir nwdir'
>>> b = os.popen(a,'r',1)
```

42. read(fd,n)

read() Python os Module will let us read at most n bytes from the desciptor fd. It returns a string holding the bytes we just read.

And if it reaches the end of file, it returns an empty string.

Sample usage:

```
>>> fd = os.open("f1.txt",os.O_RDWR)
>>> ret = os.read(fd,12)
>>> print(ret)
>>> os.close(fd)
```

43. readlink(path)

Python os Module readlink() will return a string denoting the path to which the symbolic link points. It may return a relative or an absolute pathname.

Sample usage:

```
>>> src = '/usr/bin/python'
>>> dst = '/tmp/python'
>>> os.symlink(src, dst)
>>> path = os.readlink( dst )
>>> print(path)
```

44. remove(path)

remove() removes the specified file path. If that path is a directory, it raises an OSError.

Sample usage:

```
>>> print(f"The dir is: {os.listdir(os.getcwd())}")
>>> os.remove("aa.txt")
>>> print(f"The dir after removal of path: {os.listdir(os.getcwd())}")
```

remove() doesn’t return any value.

45. removedirs(path)

This Python os Module will remove directories recursively.

And if we successfully remove the leaf directory, it attempts to successively remove every parent directory displayed in that path.

Sample usage:

```
>>> print(f"The dir is: {os.listdir(os.getcwd())}")
>>> os.removedirs("/tutorialsdir")
>>> print(f"The dir after removal is: {os.listdir(os.getcwd())}")
```

removedirs() doesn’t return any value.

46. rename(src,dst)

rename() renames a file or directory. If the destination is a file or a directory that already exists, it raises an OSError.

Sample usage:

```
>>> print(f"The dir is: {os.listdir(os.getcwd())}”)
>>> os.rename("tutorialsdir","tutorialsdirectory")
>>> print(“Successfully renamed”)
>>> print(f"The dir is: {os.listdir(os.getcwd())}")
```

rename() doesn’t return any value.

47. renames(old,new)

renames() Python os Module renames directories and files recursively.

It is like os.rename(), but it also moves a file to a directory, or a whole tree of directories, that do not already exist.

Sample usage:

```
>>> print("Current directory is: { os.getcwd()}")
>>> print("The dir is: { os.listdir(os.getcwd())}")
>>> os.renames("aa1.txt","newdir/aanew.txt")
>>> print("Successfully renamed”)
>>> print(f"The dir is: {os.listdir(os.getcwd())}")
```

renames() does not return any value.

48. rmdir(path)

Python os Module rmdir() removes the directory path specified. If the directory isn’t empty, however, it raises an OSError.

Sample usage:

```
>>> print(f"the dir is: { os.listdir(os.getcwd())}")
>>> os.rmdir("mydir")
>>> print(f"the dir is: { os.listdir(os.getcwd())}"
```

rmdir() doesn't return any value.

49. stat(path)

This Python os Module performs a stat system call on the specified path.
These are the members of the stat structure:

- st_mode – protection bits
- st_ino – inode number
- st_dev – device
- st_nlink – number of hard links
- st_uid – user id of owner
- st_gid – group id of owner
- st_size – size of file, in bytes
- st_atime – time of most recent access
- st_mtime – time of most recent content modification
- st_ctime – time of most recent metadata change.

Sample usage:

```
>>> statinfo = os.stat('a2.py')
>>> print(statinfo)
```

50. stat_float_times([newvalue])

stat_float_times() Python os Module decides whether stat_result denotes time stamps as float objects.

Sample usage:

```
>>> import os, sys
>>> statinfo = os.stat('a2.py')
>>> print(statinfo)
>>> statinfo = os.stat_float_times()
>>> print(statinfo)
```

51. statvfs(path)

Python os Module statvfs() executes a statvfs system call on the specified path.

The structure has the following members:

- f_bsize – preferred file system block size
- f_frsize – fundamental file system block size
- f_blocks – total number of blocks in the filesystem
- f_bfree – total number of free blocks
- f_bavail – free blocks available to non-super user
- f_files – total number of file nodes
- f_ffree – total number of free file nodes
- f_favail – free nodes available to non-super user
- f_flag – system dependent
- f_namemax – maximum file name length

Sample usage:

```
>>> stinfo = os.statvfs('a1.py')
>>> print(stinfo)
```

52. symlink(src,dst)

symlink() composes a symbolic link dst that points to the source.

Sample usage:

```
>>>src = '/usr/bin/python'
>>> dst = '/tmp/python'
>>> os.symlink(src, dst)
```

symlink() returns no value.

53. tcgetpgrp(fd)

This Python os Module returns the process group linked to the terminal specified by fd, which is an open file descriptor, and is returned by os.open().

Sample usage:

```
>>> print(f"Current working dir : { os.getcwd()}")
>>> fd = os.open("/dev/tty",os.O_RDONLY)
>>> f = os.tcgetpgrp(fd)
>>> print(f"the process group associated is: {f}")
>>> os.close(fd)
```

tcgetpgrp() returns the process group.

54. tcsetpgrp(fd, pg)

Python os Module tcsetpgrp() sets the process group linked to the terminal specified by fd, which is an open file descriptor, and is returned by os.open(), to pg.

Sample usage:

```
>>> print(f"Current working dir : { os.getcwd()}")
>>> fd = os.open("/dev/tty",os.O_RDONLY)
>>> f = os.tcgetpgrp(fd)
>>> print(f"the process group associated is: {f}")
>>> os.tcsetpgrp(fd,2672)
>>> print("done")
>>> os.close(fd)
```

tcsetpgrp() returns no value.

55. tempnam([dir[, prefix]])

tempnam() Python os Module returns a unique path name reasonable enough to create a temporary file.

Sample usage:

```
>>> tmpfn = os.tempnam('/tmp/tutorialsdir,'tuts1')
```

tempnam() returns a unique path.

56. tmpfile()

tmpfile() will return a new temporary file object, opening it in update mode (w+b).

This file has zero directory entries linked to it, and will automatically delete when no descriptors are available.

Sample usage:

```
>>> tmpfile = os.tmpfile()
>>> tmpfile.write('Temporary newfile is here.....')
>>> tmpfile.seek(0)
>>> print(tmpfile.read())
>>> tmpfile.close()
```

57. tmpnam()

tmpnam() will return a unique path name reasonable enough to create a temporary file.

Sample usage:

```
>>> tmpfn = os.tmpnam()
>>> print(f"This is the unique path: {tmpfn}")
```

58. ttyname(fd)

ttyname() Python os Module will return a string that denotes the terminal device linked to the descriptor fd.

If it isn't linked to a terminal device, it raises an exception.

Sample usage:

```
>>> print(f"Current working dir : { os.getcwd()}")
>>> fd = os.open("/dev/tty",os.O_RDONLY)
>>> p = os.ttyname(fd)
>>> print(f"the terminal device associated is: {p}")
>>> os.close(fd)
```

ttyname() returns a string that denotes the terminal device.

59. unlink(path)

This Python os Module will remove specified file path. If it is a directory, it raises an OSError.

Sample usage:

```
>>> print(f"The dir is: { os.listdir(os.getcwd())}")
>>> os.unlink("aa.txt")
>>> print(f"The dir after removal of path : { os.listdir(os.getcwd())}")
```

unlink() doesn't return any value.

60. utime(path,times)

Python os Module utime() sets the access and modified times of the file at the specified path.

Sample usage:

```
>>> stinfo = os.stat('a2.py')
>>> print(stinfo)
>>> print(f"access time of a2.py: { stinfo.st_atime }")
>>> print(f"modified time of a2.py: { stinfo.st_mtime }")
>>> os.utime("a2.py",(1330712280, 1330712292))
```

utime() returns no value.

61. walk(top[, topdown=True[, onerror=None[, followlinks=False]]])

walk() creates file names in a directory tree. It does so by walking the tree either bottom-up or top-down.

It has the following parameters:

1. top – Each directory rooted at directory
2. topdown – If topdown is True, or not specified, it scans directories top-down.
3. onerror – This may show an error to continue with the walk, or may raise an exception to abort the walk.
4. followlinks – This will visit directories that symlinks points to, that is, if set to true.

Sample usage:

```
>>> for root, dirs, files in os.walk(".", topdown=False):
```

Output

```
for name in files:
print(os.path.join(root, name))
```

```
for name in dirs:
print(os.path.join(root, name))
```

62. write(fd,str)

This Python os Module will write the specified string to descriptor fd. It returns the number of bytes that it actually wrote.

Sample usage:

```
>>> fd = os.open("f1.txt",os.O_RDWR|os.CREAT)
>>> ret = os.write(fd,"This is test")
>>> print(f"the number of bytes written: {ret}")
>>> print("written successfully")
>>> os.close(fd)
```

So, this was all about Python os Module. Hope you like our explanation.

Python Interview Questions on OS Modules

1. What is OS Module in Python?
2. How to open an OS Module in Python?
3. What is the use of OS Module in Python?
4. What is OS Path Module in Python?
5. What is import OS SYS in Python?

Conclusion

Hence, we cover all the Python os module. This will allow you to work your way around the directories without any problem.

Did you know we work 24x7 to provide you best tutorials
Please encourage us - write a review on **Google** | **Facebook**

Tags: List of Python os Module Os Module in Python Python Modules Python os Modules

2 RESPONSES

Comments 2 Pingbacks 0

Kiran August 8, 2019 at 8:44 pm
Hi..how to install applications using Python in Windows remote servers
Reply

DataFlair September 27, 2021 at 9:16 pm
You can use pip to install and manage software packages using python on windows.
Reply

LEAVE A REPLY

Comment *

Name * This site is **Email** *

protected by reCAPTCHA and the Google Privacy Policy and Terms of Service apply.

Post Comment