Contents

Toggle Table of Contents

Python Guides

Python Modules

Python Modules with Syntax & Examples [Latest]

Python Modules with Syntax & Examples [Latest]

by Sagar Sharma Published: March 31, 2022 Updated: September 27, 2022

1. Python os Module

In this tutorial on Python os Module, we will go learn how to Access modules and its methods. Moreover, you will study system environment vars in Module in Python Programming language.

So, let's start Python Module Tutorial.

Table of Contents

1. Python os Module

The Python OS module lets us work with files and directories. We have been using it at various places in the Modules as we were importing it. Before we move forward, let's discuss the important functions/methods in Python. In case of any doubts, please ask us in comments. Let's check the step by step method.

>>> dir(os)

['CLD_CONTINUED', 'CLD_DUMPED', 'CLD_EXITED', 'CLD_TRAPPED', 'EX_CANTCREAT', 'EX_CONFIG', 'EX_DATAERR', 'EX_IOERR', 'EX_NOHOST', 'EX_NOINPUT', 'EX_NOPERM', 'EX_NOUSER', 'EX_OSERR', 'EX_OSFILE', 'EX_PROTOCOL', 'EX_SOFTWARE', 'EX_TEMPFAIL', 'EX_UNAVAILABLE', 'EX_USAGE', 'F_LOCK', 'F_OK', 'F_TEST', 'F_TLOCK', 'F_ULOCK', 'GenericAlias', 'MutableMapping', 'NGROUPS_MAX', 'O_ACCMODE', 'O_APPEND', 'O_ASYNC', 'O_BINARY', 'O_CLOEXEC', 'O_CREAT', 'O_DIRECT', 'O_DIRECTORY', 'O_DSYNC', 'O_EXCL', 'O_LARGEFILE', 'O_NDELAY', 'O_NOATIME', 'O_NOCTTY', 'O_NOFOLLOW', 'O_NOINHERIT', 'O_NONBLOCK', 'O_PATH', 'O_RANDOM', 'O_RDONLY', 'O_RDWR', 'O_RSYNC', 'O_SEQUENTIAL', 'O_SHLOCK', 'O_SHORT_LIVED', 'O_SYMLINK', 'O_SYNC', 'O_TEMPORARY', 'O_TEXT', 'O_TMPFILE', 'O_TRUNC', 'O_WRONLY', 'P_DETACH', 'P_NOWAIT', 'P_NOWAITO', 'P_OVERLAY', 'P_WAIT', 'PRIO_PGRP', 'PRIO_PROCESS', 'PRIO_USER', 'P_all', 'P_pgid', 'P_pid', 'P_PGID', 'P_PID', 'PIDFD_NONBLOCK', 'POSIX_FADV_DONTNEED', 'POSIX_FADV_NOREUSE', 'POSIX_FADV_NORMAL', 'POSIX_FADV_RANDOM', 'POSIX_FADV_SEQUENTIAL', 'POSIX_FADV_WILLNEED', 'PRIO_MAX', 'PRIO_MIN', 'P_PIDFD', 'pwrite', 'RTLD_DEEPBIND', 'RTLD_GLOBAL', 'RTLD_LAZY', 'RTLD_LOCAL', 'RTLD_NODELETE', 'RTLD_NOLOAD', 'RTLD_NOW', 'SCHED_BATCH', 'SCHED_FIFO', 'SCHED_IDLE', 'SCHED_OTHER', 'SCHED_RESET_ON_FORK', 'SCHED_RR', 'SCHED_SPORADIC', 'SEEK_DATA', 'SEEK_HOLE', 'ST_APPEND', 'ST_MANDLOCK', 'ST_NOATIME', 'ST_NODEV', 'ST_NODIRATIME', 'ST_NOEXEC', 'ST_NOSUID', 'ST_RDONLY', 'ST_RELATIME', 'ST_SYNCHRONOUS', 'ST_WRITE', 'TMP_MAX', 'WCONTINUED', 'WCOREDUMP', 'WEXITED', 'WEXITSTATUS', 'WIFCONTINUED', 'WIFEXITED', 'WIFSIGNALED', 'WIFSTOPPED', 'WNOHANG', 'WNOWAIT', 'WSTOPPED', 'WSTOPSIG', 'WTERMSIG', 'WUNTRACED', '__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__path__', '__spec__', '_check_methods', '_execvpe', '_exists', '_exit', '_fspath', '_get_exports_list', '_putenv', '_unicodefromfd', '_unsetenv', '_wrap_close', 'abc', 'abort', 'access', 'altsep', 'chdir', 'chmod', 'chown', 'chroot', 'close', 'closerange', 'confstr', 'confstr_names', 'cpu_count', 'ctermid', 'curdir', 'defpath', 'device_encoding', 'devnull', 'dup', 'dup2', 'environ', 'environb', 'error', 'eventfd', 'eventfd_read', 'eventfd_write', 'execl', 'execle', 'execlp', 'execlpe', 'execv', 'execve', 'execvp', 'execvpe', 'extsep', 'fchdir', 'fchmod', 'fchown', 'fdatasync', 'fdopen', 'fork', 'forkpty', 'fpathconf', 'fsdecode', 'fsencode', 'fstat', 'fstatvfs', 'fsync', 'ftruncate', 'get_blocking', 'get_exec_path', 'get_handle_inheritable', 'get_inheritable', 'get_terminal_size', 'getcwd', 'getcwdb', 'getegid', 'getenv', 'getenvb', 'geteuid', 'getgid', 'getgrouplist', 'getgroups', 'getloadavg', 'getlogin', 'getpgid', 'getpgrp', 'getpid', 'getppid', 'getpriority', 'getrandom', 'getresgid', 'getresuid', 'getsid', 'getuid', 'getxattr', 'grantpt', 'initgroups', 'isatty', 'kill', 'killpg', 'lchflags', 'lchmod', 'lchown', 'linesep', 'link', 'listdir', 'listxattr', 'lockf', 'lseek', 'lstat', 'major', 'makedev', 'makedirs', 'memfd_create', 'minor', 'mkdir', 'mkdirat', 'mkfifo', 'mknod', 'name', 'nice', 'open', 'openpty', 'pardir', 'pathconf', 'pathconf_names', 'pathsep', 'pidfd_open', 'pipe', 'pipe2', 'plock', 'popen', 'posix_fadvise', 'posix_fallocate', 'pread', 'preadv', 'putenv', 'pwrite', 'pwritev', 'read', 'readlink', 'readv', 'register_at_fork', 'remove', 'removedirs', 'rename', 'renames', 'replace', 'rmdir', 'scandir', 'sched_get_priority_max', 'sched_get_priority_min', 'sched_getaffinity', 'sched_getparam', 'sched_getscheduler', 'sched_param', 'sched_rr_get_interval', 'sched_setaffinity', 'sched_setparam', 'sched_setscheduler', 'sched_yield', 'sendfile', 'sep', 'set_blocking', 'set_inheritable', 'setegid', 'seteuid', 'setgid', 'setgroups', 'setpgid', 'setpgrp', 'setpriority', 'setregid', 'setresgid', 'setresuid', 'setreuid', 'setsid', 'setuid', 'setxattr', 'spawnl', 'spawnle', 'spawnv', 'spawnve', 'st', 'stat', 'stat_result', 'statvfs', 'statvfs_result', 'strerror', 'supports_bytes_environ', 'supports_dir_fd', 'supports_effective_ids', 'supports_fd', 'supports_follow_symlinks', 'symlink', 'sync', 'sysconf', 'sysconf_names', 'system', 'tcgetpgrp', 'tcsetpgrp', 'terminal_size', 'times', 'times_result', 'truncate', 'ttyname', 'umask', 'uname', 'uname_result', 'unlink', 'unsetenv', 'urandom', 'utime', 'wait', 'wait3', 'wait4', 'waitid', 'waitid_result', 'waitpid', 'walk', 'write', 'writev']

Module: Python Os Module (Full)

2. mkdir(pathname)

This method can be used with the path parameter to create a new path. If no error is allowed, it creates it then. But, it returns if also. The file exception is the path, the second is the mode. This mode can determine all these values.

>>> os.mkdir("/home/sagar/Desktop/Test")

- os.R_OK — Readable

- os.W_OK — Writable

- os.X_OK — Executable

Many of these are example

```
>>> os.chdir('C:\\Users\\Tutorial\\Desktop')
>>> os.access("Testing.txt", os.R_OK)
```

True

```
>>> os.access("Testing.txt", os.W_OK)
```

True

```
>>> os.access("Testing.txt", os.X_OK)
```

True

```
>>> os.access("Testing.txt", os.F_OK)
```

True

Result [Python Checking Access to a File - Real, Effective and Existence](#)

os.chdir(path)

This Python os module changes the current working [Directory](#) to the path we specify.
Many of the examples below include it.

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

It returns None.

Result [Python Changing a Current Working Directory \(chdir\)](#)

os.chflags(path,flags)

chflags() sets path flags to the numeric flags. These flags may be combined (bitwise-ORed) of the following values:

- stat.UF_NODUMP - Don't dump the file

- stat.UF_IMMUTABLE - The file may not change.

- stat.UF_APPEND - The file may only append to the file

- stat.UF_OPAQUE - The file may not be viewed in Union the file

- stat.UF_NOUNLINK - The file may not be renamed or deleted the file

- stat.UF_COMPRESSED - The file is a compressed store one or more through source stock

- stat.UF_HIDDEN - This may or may not be the file

- stat.SF_ARCHIVED - The file may or may not change the file

- stat.SF_IMMUTABLE - The file may not change the file

- stat.SF_APPEND - The file may only append to the file

- stat.SF_NOUNLINK - The file may not be renamed or deleted the file

- stat.SF_SNAPSHOT - This is a snapshot file

Many flags are used in the unix file, but I am not. so change. Once, many, many flags don't work on all systems.

Example except:

```
>>> os.chflags("Testing .txt", os.UF_IMMUTABLE)
```

Result [Python Checking Permissions and Access with chmod\(\)](#)

os.chmod(path,mode)

This Python os file alters the mode of the path to the passed numeric mode. The mode may be one of the following valueor a bit-wise ORcombination of them):

- stat.S_ISUID - Set user ID on execution

- stat.S_ISGID - Set group ID on execution

- stat.S_ENFMT - Enforced record file locing

- stat.S_ISVTX - Orbit Execution, now not integr

- 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

Example except:

```
>>> import os, stat
>>> os.chmod("Testing .txt", stat.S_IWRITE)
```

This method does not return any value.

os.chown(path,uid,gid)

chown() Python os. Module alters the owner and group of the path to the numeric uid and gid. To use this, we need higher-level privileges.

This will be the owner ID for the file. The gid is the group ID to be set for the file.

Example except:

```
>>> os.chown("Testing .txt", 1000, -1)
```

This method returns no value.

Result [Python Changing Owner and Group](#)

os.chroot(path)

chroot() Python os. Module alters the current process' root directory to the given path. To use this, we need higher-level privileges.

Example except:

```
>>> os.chroot("C:\\Windows")
```

This method returns no value.

os.close(fd)

This Python os method closes the associated File with descriptor fd.

```
>>> file=os.open("Testing .txt", os.O_RDWR)
>>> os.close(file)
```

Python not return any value.

os.closerange(fd1,fd2,/*ignore*/)

closerange() closes all file descriptors from fd1 (one fd_Right. Mine, fd2 (as one less one), and fd_Right one less one. Mine, fd_2 one of the closes the descriptors to be closed, while fd_Right as the highest. This method ignores errors.

Example except:

```
>>> fd = os.open( "Testing .txt", os.O_RDWR)
>>> fd_2 = os.dup( fd )
>>> os.closerange( fd, fd_2 )
```

This method does not return any value.

Result [Python Closing All Open File Descriptors with closerange\(\) Function](#)

os.dup(fd)

Python os Window dup(fd) returns a duplicate of the file descriptor fd.
Example except:

```
>>> fd = os.open( "Testing .txt", os.O_RDWR)
>>> fd_2 = os.dup( fd )
>>> os.write( fd, b"Testing" )
>>> os.closerange( fd, fd_2 )
```

This method returns a duplicate one of the file descriptor.

os.dup2(fd2fd1)

dup2() duplicate file descriptor fd to fd1, and if necessary, a closes fd2 first.

The assignee assigns the one file descriptor early when it is available.

Example except:

```
>>> fd = os.open( "Testing .txt", os.O_RDWR)
>>> os.write(fd, "Hello!\n")
>>> fd_2 = os.dup( fd )
>>> os.dup2(fd, fd2)
>>> os.write(fd_2, b"Hello!\n")
>>> os.close( fd )
>>> os.close( fd_2 )
```

This method does not return any value.

Result [Python Duplicating, Closing and Copying a File Descriptor](#)

os.fchdir(fd)

fchdir() alters the current working directory to the directory described by the file descriptor fd opposites. For this, it is mandatory. Otherwise,fchdir() raises will a is accepted. Access by, and need not again fails.

Example except:

```
>>> os.chdir("C:\\Users\\Tutorial\\Desktop")
>>> print("Current working dir : %s" % os.getcwd())
>>> fd = os.open("C:\\Users\\Tutorial", os.O_RDONLY)
>>> os.fchdir(fd)
>>> print("Current working dir : %s" % os.getcwd())
>>> os.close( fd )
```

This method does not return any value.

os.fchmod(fd,mode)

This Python os file alters the mode of the file, specified by fd, to the numeric mode. We mode may be one of the following (or ors ORed combinations of):

- stat.S_ISUID - Set user ID on execution

- stat.S_ISGID - Set group ID on execution

- stat.S_ENFMT - Record locing enforce on

- stat.S_ISVTX - Sticon bits (Mega 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

Example except:

```
>>> import os, stat
>>> fd = os.open( "Testing .txt", os.O_RDWR)
>>> os.fchmod(fd, stat.S_IWRITE)
>>> os.close( fd )
```

This method does not return any value.

Result [Python Changing Permissions with chmod\(\) and fchmod\(\) Functions \(Windows\)](#)

os.fchown(fd,uid,gid)

fchown() alters the owner and the group id of the files specified by. this file receives' will and. gid. Setting an of to -1 leaves it unchanged.
Example except:

```
>>> fd = os.open( "Tutorial", os.O_RDONLY )
>>> os.fchown( fd, 1000, -1 )
>>> os.fchown( fd, -1, 10 )
>>> print( "Changed! Owner:%d, Group:%d" %(os.fstat(fd).st_uid, os.fstat(fd).st_gid))
>>> os.close( fd )
```

This method does not return any value.

os.fdatasync(fd)

Forces a file to write with data fdatasync() to the disk. This function doesn't force update on metadata. This can be like on flush your buffer.

Example except:

```
>>> fd = os.open( "Testing .txt", os.O_RDWR)
>>> os.write(fd, "Testing.txt")
>>> os.fdatasync(fd)
>>> print( "Changed! Owner:%d, Group:%d" %(os.fstat(fd).st_uid, os.fstat(fd).st_gid))
>>> os.close( fd )
```

This method doesn't return any value.

Result [Python Forcing Writing to Disk with fsync and fdatasync](#)

os.fdopen(fd,mode='r',buffering=-1)

Returns a file object, which is associated with the file descriptor fd. This is identical to the open() function. This can be like on flush your buffer.

Example except:

```
>>> fd = os.open( "Testing .txt", os.O_RDWR)
>>> os.write(fd, "Testing.txt")
>>> os.fdatasync(fd)
>>> print( "Changed! Owner:%d, Group:%d" %(os.fstat(fd).st_uid, os.fstat(fd).st_gid))
>>> os.close( fd )
```

This method doesn't return any value.

Result [Python Forcing Writing to Disk with fsync and fdatasync](#)

os.fpathconf(fd,name)

Returns system configuration information that is relevant to an open file. The name parameter is the value you want to retrieve from the open file. It may be a string whose name is known, or an integer whose value is known.

Example except:

```
>>> fd = os.open( "Testing .txt", os.O_RDWR)
>>> print ("File path:%s" %os.pathconf("Testing .txt", os.PC_PATH_MAX))
>>> print ("Maximum number of links to file:%d" %os.fpathconf(fd, os.PC_LINK_MAX))
>>> print ("Maximum number of links to file:%d" %os.fpathconf(fd, "PC_LINK_MAX"))
>>> os.close( fd )
```

Passing in name or a pair the object that is connected to the file we are pon

os.fpathconf(fd,name)

fpathconf() returns system configuration information that is relevant to an open file. This is pon similar to the name system call (python.h), in does not give another arguments.

Example except:

```
>>> fd = os.open( "Testing .txt", os.O_RDWR)
>>> print ("Link pathconf:%d" %os.fpathconf(fd, "PC_LINK_MAX"))
>>> print ("Maximum number of links to file:%d" %os.fpathconf(fd, "PC_LINK_MAX"))
>>> print ("Maximum number of links to file:%d" %os.fpathconf(fd, os.PC_LINK_MAX))
>>> os.close( fd )
```

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

Result [Python Accessing Information about Files](#)

os.fstat(fd)

Python os Window fstat() returns information about the file pointed by an file fd. Isn't unless most. of 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_blocks – Number for filesystem I/O

- st_blocks – number of file this allocated

- st_atime – time of last access

- st_mtime – time of last modification

- st_ctime – time of last status change

Example except:

```
>>> fd = os.open( "Testing .txt", os.O_RDWR)
>>> print ("Link pathconf:%d" %os.fstat(fd).st_nlink)
>>> print ("Maximum number of links to file:%d" %os.fstat(fd).st_uid)
>>> print ("Maximum number of links to file:%d" %os.fstat(fd).st_gid)
>>> os.close( fd )
```

Many times information about the file. Name and the fd.

os.fstatvfs(fd)

This Python os module returns information about the file system containing the file fstatvfs() and the descriptor fd. This is the same as stat.vfs.

- 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_flag – mount flags

- f_namemax – maximum filename length

Default usage:

```
$ ffz --dir ./gallery --format "{index}_{ext}" --dry-run
would move ./gallery/1.jpg -> ./gallery/1_jpg.jpg
would move ./gallery/2.png -> ./gallery/2_png.png
would move ./gallery/3.webp -> ./gallery/3_webp.webp
```

Preview only, no changes made.

Custom format with padding:

```
$ ffz --dir ./gallery --format "{index:04}_{ext}" --dry-run
would move ./gallery/1.jpg -> ./gallery/0001_jpg.jpg
would move ./gallery/2.png -> ./gallery/0002_png.png
would move ./gallery/3.webp -> ./gallery/0003_webp.webp
```

Four-digit padding.

Starting from 100:

```
$ ffz --dir ./gallery --format "{index}_{ext}" --start 100 --dry-run
would move ./gallery/1.jpg -> ./gallery/100_jpg.jpg
would move ./gallery/2.png -> ./gallery/101_png.png
would move ./gallery/3.webp -> ./gallery/102_webp.webp
```

Start counting at 100.

Custom separator:

```
$ ffz --dir ./gallery --format "{index}-{ext}" --dry-run
would move ./gallery/1.jpg -> ./gallery/1-jpg.jpg
would move ./gallery/2.png -> ./gallery/2-png.png
would move ./gallery/3.webp -> ./gallery/3-webp.webp
```

Use a hyphen instead of underscore.

Actually rename files:

```
$ ffz --dir ./gallery --format "{index}_{ext}"
renamed ./gallery/1.jpg -> ./gallery/1_jpg.jpg
renamed ./gallery/2.png -> ./gallery/2_png.png
renamed ./gallery/3.webp -> ./gallery/3_webp.webp
```

Rename files. No undo.

Recursive:

```
$ ffz --dir ./gallery --format "{index}_{ext}" --recursive --dry-run
would move ./gallery/1.jpg -> ./gallery/1_jpg.jpg
would move ./gallery/sub/1.jpg -> ./gallery/sub/1_jpg.jpg
would move ./gallery/sub/2.png -> ./gallery/sub/2_png.png
```

Include subdirectories. Each directory is numbered independently.

Combined:

```
$ ffz --dir ./gallery --format "{index:04}_{ext}" --start 100 --recursive --dry-run
would move ./gallery/1.jpg -> ./gallery/0100_jpg.jpg
would move ./gallery/sub/1.jpg -> ./gallery/sub/0100_jpg.jpg
would move ./gallery/sub/2.png -> ./gallery/sub/0101_png.png
```

Padding, custom start, and recursion together.

Invalid format:

```
$ ffz --dir ./gallery --format "{name}_{ext}" --dry-run
error: unknown placeholder {name}
```

Unknown placeholder. Only {index} and {ext} are supported.

Missing placeholder:

```
$ ffz --dir ./gallery --format "{index}" --dry-run
error: format must contain {ext}
```

Format must contain {ext}.

Nonexistent directory:

```
$ ffz --dir ./nope --format "{index}_{ext}" --dry-run
error: directory not found: ./nope
```

Directory not found.

Empty directory:

```
$ ffz --dir ./empty --format "{index}_{ext}" --dry-run
no files found in ./empty
```

No files found.

Already-named files are skipped:

```
$ ffz --dir ./gallery --format "{index}_{ext}" --dry-run
would move ./gallery/1.jpg -> ./gallery/2_jpg.jpg
skipped ./gallery/2_png.png (already named correctly)
would move ./gallery/3.webp -> ./gallery/4_webp.webp
```

Files already named correctly are skipped.

Collision:

```
$ ffz --dir ./gallery --format "{index}_{ext}" --dry-run
error: ./gallery/1_jpg.jpg already exists
```

Refuses to overwrite existing files.

Hidden files ignored:

```
$ ffz --dir ./gallery --format "{index}_{ext}" --dry-run
would move ./gallery/1.jpg -> ./gallery/1_jpg.jpg
skipped ./gallery/.DS_Store (hidden)
would move ./gallery/2.png -> ./gallery/2_png.png
```

Hidden files are skipped.

Symlinks ignored:

```
$ ffz --dir ./gallery --format "{index}_{ext}" --dry-run
would move ./gallery/1.jpg -> ./gallery/1_jpg.jpg
skipped ./gallery/shortcut (symlink)
would move ./gallery/2.png -> ./gallery/2_png.png
```

Symlinks are skipped.

Permission denied:

```
$ ffz --dir ./locked --format "{index}_{ext}" --dry-run
error: permission denied: ./locked
```

Cannot read directory.

Read-only file:

```
$ ffz --dir ./gallery --format "{index}_{ext}"
renamed ./gallery/1.jpg -> ./gallery/1_jpg.jpg
error: permission denied: ./gallery/2.png
renamed ./gallery/3.webp -> ./gallery/3_webp.webp
```

Skips files it can't write to.

Dry run:

```
$ ffz --dir ./gallery --format "{index}_{ext}" --dry-run
would move ./gallery/1.jpg -> ./gallery/1_jpg.jpg
would move ./gallery/2.png -> ./gallery/2_png.png
would move ./gallery/3.webp -> ./gallery/3_webp.webp
```

Preview changes without renaming.

Verbose:

```
$ ffz --dir ./gallery --format "{index}_{ext}" --verbose
scanning ./gallery
found 3 files
skipped ./gallery/.DS_Store (hidden)
renamed ./gallery/1.jpg -> ./gallery/1_jpg.jpg
renamed ./gallery/2.png -> ./gallery/2_png.png
renamed ./gallery/3.webp -> ./gallery/3_webp.webp
done: 3 renamed, 1 skipped, 0 errors
```

Show every file, including skipped ones.

Quiet:

```
$ ffz --dir ./gallery --format "{index}_{ext}" --quiet
```

Suppress all output. Exit code only.

Undo:

```
$ ffz --undo
undoing 3 renames
restored ./gallery/1_jpg.jpg -> ./gallery/1.jpg
restored ./gallery/2_png.png -> ./gallery/2.png
restored ./gallery/3_webp.webp -> ./gallery/3.webp
```

Reverses the last operation.

Undo nothing:

```
$ ffz --undo
error: nothing to undo
```

No history to undo.

Undo after files changed:

```
$ ffz --undo
error: ./gallery/1_jpg.jpg no longer exists
restored ./gallery/2_png.png -> ./gallery/2.png
restored ./gallery/3_webp.webp -> ./gallery/3.webp
```

Skips files that are gone, reverses the rest.

Undo twice:

```
$ ffz --undo
error: nothing to undo
```

Only one level of undo.

Help:

```
$ ffz --help
ffz — batch rename files in a directory

USAGE:
  ffz --dir <path> --format "<template>" [options]

OPTIONS:
  --dir <path>          Directory to rename files in
  --format "<template>" Rename template (required)
  --start <n>           Starting index (default: 1)
  --recursive           Include subdirectories
  --dry-run             Preview changes without renaming
  --verbose             Show all files, including skipped
  --quiet               Suppress all output
  --undo                Reverse the last operation
  --help                Show this help
  --version             Show version

TEMPLATE PLACEHOLDERS:
  {index}               Sequential number
  {index:04}            Zero-padded number (any width)
  {ext}                 File extension
```

Version:

```
$ ffz --version
ffz 0.1.0
```

Why

Every developer has a script like this. Every one is slightly different. Every one is a little bit wrong.

`ffz` is the one you install once and stop thinking about.

Design

- **Safe by default.** Dry run is the default. You must explicitly pass `--apply` to touch anything.
- **Undo.** Every operation is logged. `ffz undo` reverses the last one.
- **Fast.** Written in Rust. Renames 10,000 files in under a second.
- **Predictable.** Same input, same output. No surprises.
- **Composable.** Reads from stdin, writes to stdout, plays well with pipes.

Install

```
cargo install ffz
```

Or grab a binary from the releases page.

Usage

```
ffz [OPTIONS] [PATH]
```

If `PATH` is omitted, operates on the current directory.

Options

Flag	Description
`--apply`	Actually perform the operations (default is dry run)
`--undo`	Reverse the last applied operation
`--history`	Show the operation log
`--clear-history`	Delete the operation log
`--recursive`	Descend into subdirectories
`--depth <n>`	Limit recursion depth
`--include <glob>`	Only match files matching the glob
`--exclude <glob>`	Skip files matching the glob
`--hidden`	Include dotfiles
`--case <mode>`	Transform case: `lower`, `upper`, `snake`, `kebab`, `camel`, `pascal`
`--ext <ext>`	Only operate on files with this extension
`--find <pattern>`	Find pattern for substitution
`--replace <text>`	Replace matches of `--find`
`--regex`	Treat `--find` as a regex
`--prepend <text>`	Add text to the start of the filename
`--append <text>`	Add text to the end of the filename
`--number <mode>`	Add sequential numbers: `prefix`, `suffix`
`--start <n>`	Starting number for `--number` (default: 1)
`--pad <n>`	Zero-pad numbers to width `n`
`--sort <mode>`	Sort order: `name`, `mtime`, `size`, `created`
`--reverse`	Reverse sort order
`--limit <n>`	Only process the first `n` files
`--interactive`	Confirm each operation
`--quiet`	Suppress all output except errors
`--verbose`	Show detailed output
`--color <mode>`	Color output: `auto`, `always`, `never`
`--version`	Print version
`--help`	Print help

Examples

Lowercase all filenames:

```
ffz --case lower --apply
```

Replace spaces with underscores:

```
ffz --find " " --replace "_" --apply
```

Add a date prefix to photos:

```
ffz --prepend "2024-" --ext jpg --apply
```

Number files in sort order:

```
ffz --number prefix --pad 3 --apply
```

Produces `001_`, `002_`, etc.

Clean up a messy downloads folder:

```
ffz ~/Downloads --case lower --find " " --replace "_" --exclude "*.part" --apply
```

Regex rename:

```
ffz --regex --find "^(\d{4})-(\d{2})-(\d{2})" --replace "$3-$2-$1" --apply
```

Reorganize `2024-01-15` into `15-01-2024`.

Preview before committing:

```
ffz --case snake --recursive
ffz --apply
```

Always dry run first.

Undo a mistake:

```
ffz --undo
```

Made a mistake? Undo it.

Philosophy

`ffz` does one thing well. It renames files. It doesn't try to be a build system, a package manager, or a media converter. It composes with the tools you already have.

It never asks you to learn a configuration language. Flags are enough.

It never surprises you. Dry run by default. Undo always available. Errors are loud and clear.

It never loses your work. Operations are logged before they're applied.

Contributing

Issues and pull requests welcome at https://github.com/yourname/ffz

License

MIT

Python Tutorials

- Python – Introduction
- Python – Features
- Python – Pros and Cons
- Python – Applications
- Python – Career Opportunities
- Python – Tools
- Python – Data Science Tutorial
- Python – For Data Science
- Python – Data Science Installation
- Python 3.6 Features

Python – Install on Windows
Python – Syntax
Python – Comments, Indentations and Statements
Python – Assert Statements
Python – Number Types
Python – Random Number
Python – Variables and Data Types
Python – Variable Scope
Python – Identifiers
Python – Strings
Python – Interpreter
Python – Operators
Python – Bitwise Operators
Python – Comparison Operators
Python – Operator Overloading
Python – Ternary Operator
Python – Operator Precedence
Python – Namespaces
Python – Decision Making
Python – Loops in Python
Python – Implement Switch Case
Python – Functions
Python – Function Arguments
Python – Built-in Functions
Python – Methods
Python – Methods vs Functions
Python – Constructors
Python – Range() Function
Python – Zip Function
Python – Eval Function
Python – exec Function
Python – repr Function
Python – Data Structures
Python – Lists
Python – Tuples
Python – Tuples vs Lists
Python – Collections Module

Python – Counters

Python – Namedtuples

Python – DefaultDict

Python – OrderedDict

Python – Dictionaries

Python – DateTime

Python – Modules

Python – Serialization

Python – Packages

Python – Python OS Module

Python – Virtual Environment

Python – Modules vs Packages

Python – Sets & Booleans

Python – List Comprehension

Python – Date and Time

Python – Calendar Module

Python – Recursion

Python – Lambda Expression

Python – Decorators

Python – Generators

Python – Iterators

Python – Generators vs Iterators

Python – Closures

Python – Classes

Python – Object

Python – Inheritance

Python – Multiple Inheritance

Python – Compilers & Interpreters

Python – ZipFile

Python – File I/O

Python – File Handling

Python – Copy A File

Python – Shallow & Deep Copy

Python – Rename A File

Python – Data File Formats

Python – Errors and Exceptions

Python – Exception Handling

Python – Directories
Python – Itertool
Python – Property
Python – Sequences and Collections
Python – Stemming & Lemmatization
Python – Aggregation & Data Wrangling
Python – Data Cleansing & Operations
Python – Multithreading
Python – Regular Expressions
Python – Debugger
Python – Multi Processing
Python – XML Processing
Python – CGI Programming
Python – Library
Python – Matplotlib
Python – SciPy
Python – NumPy
Python – Pandas
Python – PyQT
Python – Array Module
Python – Database Access
Python – Programming with C
Python – NoSQL Database
Python – Relational Database
Python – Frameworks
Python – Flask
Python – Django
Python – Forensics
Python – Network Programming
Python – Image Processing
Python – Sending Email
Python – GUI Programming
Python – Statistics
Python – Descriptive Statistics
Python – Probability Distributions
Python – Linear Regression
Python – Histogram & Bar Plot

Python – Bubble & 3D Charts
Python – Scatter & Box Plots
Python – Heatmap & Word Cloud
Python – Geographical Map & Graph
Python – Time Series
Python – Unittest
Python – Logging
Python – Slice
Python – Subprocess Module
Python – sys Module
Python ML – Tutorial
Python ML – Environment Setup
Python ML – Data Pre-Processing
Python ML – Train and Test Set
Python ML – Techniques
Python ML – Application
Python ML – Algorithms
Python Deep Learning Tutorial
Python DL – Environment Setup
Python DL – Application
Python DL – Python Libraries
Python DL – Deep Neural Networks
Python DL – Computational Graphs
Python AI Tutorial
Python AI – NLTK
Python AI – Speech Recognition
Python AI – NLP Tutorial
Python AI – Heuristic Search
Python AI – Genetic Algorithms
Python AI – Computer Vision
Python AI – Logic Programming
Python – Best Python Books
Python – Reasons Why Learn Python
Python – Terminologies Part 1
Python – Terminologies Part 2
Python vs Scala
Python vs Java

Python vs R

Python Interview Questions

Python – Interview Questions Part 1

Python – Interview Questions Part 2

Python – Interview Questions Part 3

Python – Interview Questions Part 4

Python – Interview Questions Part 5

Python – Interview Questions Part 6