Python `os` module — Cheatsheet & Infovisual

A compact reference and visual-style cheatsheet of common `os` module functions, real-world usage, and gotchas. Use this to build scripts, automation, and maintain cross-platform behavior.

Quick legend

- ✅= Common / safe to use
- ⚠️= Caution (may raise exception or be platform-dependent)
- 🔁= Often used in loops / repeated runs

---

1. Path helpers (use these instead of hardcoding separators)

Function	Purpose	Example	Notes
`os.path.join(*parts)` ✅	Join path parts using OS separator	`p = os.path.join('data', '2025', 'file.txt')`	Use instead of `+/` or `\\`
`os.path.exists(path)` ✅	Check if path exists (file or dir)	`os.path.exists(p)`	Returns `False` for broken symlinks on some OSes
`os.path.isfile(path)` ✅	Is regular file?	`if os.path.isfile(p): ...`	True only for files
`os.path.isdir(path)` ✅	Is directory?	`os.path.isdir(d)`	Useful after `exists()`
`os.path.basename(path)` ✅	Get filename	`os.path.basename('/a/b.txt') -> 'b.txt'`	
`os.path.dirname(path)` ✅	Get parent folder	`os.path.dirname('/a/b.txt') -> '/a'`	
`os.path.abspath(path)` ✅	Absolute path resolution	`os.path.abspath('.')`	Resolves `.` and `..`

Function	Purpose	Example	Notes
`os.path.realpath(path)` ⚠️	Resolve symlinks	`os.path.realpath('link')`	Use when symlinks matter

---

2. Directory creation / removal

Function	Purpose	Example	Exceptions / Tips
`os.mkdir(path)` ✅	Create single directory	`os.mkdir('reports')`	`FileExistsError` if exists; `FileNotFoundError` for missing parents
`os.makedirs(path, exist_ok=False)` ✅🔁	Create nested dirs	`os.makedirs('a/b/c', exist_ok=True)`	`exist_ok=True` avoids `FileExistsError`
`os.rmdir(path)` ⚠️	Remove empty directory	`os.rmdir('tmp')`	`OSError` if not empty or no permission
`os.remove(path)` / `os.unlink(path)` ✅	Delete a file	`os.remove('a.txt')`	`FileNotFoundError` if missing; use `os.path.exists()` first
`shutil.rmtree(path)` ⚠️	Recursively delete dir tree	`import shutil; shutil.rmtree('build')`	**DANGEROUS** — permanent deletion; prefer confirmation or dry-run

Tip: For safe deletions use checks and backups. `rmdir` only for empty dirs.

---

3. Directory listing & traversal

Function	Purpose	Example	Notes
`os.listdir(path)` ✅	List names in directory	`os.listdir('.')`	Returns names (not full paths)

Function	Purpose	Example	Notes
`os.scandir(path)` ✅🔁	Efficient iterator returning DirEntry	`with os.scandir('.') as it: for e in it: ...`	`DirEntry` has `.is_file()` and `.stat()` without extra syscall
`os.walk(top)` ✅🔁	Walk directory tree	`for root, dirs, files in os.walk('data'): ...`	Great for backups, searches; yields top-down by default

4. File metadata & permissions

Function	Purpose	Example	Notes
`os.stat(path)` ✅	Get file metadata (size, mtime, mode)	`st = os.stat('a.txt')`	`st.st_size`, `st.st_mtime`
`os.chmod(path, mode)` ⚠️	Change file mode (permissions)	`os.chmod('script.sh', 0o755)`	Platform-specific behavior on Windows
`os.utime(path, times=None)` ✅	Set access & modified times	`os.utime('a.txt', (atime, mtime))`	Useful for caching/ timestamping

5. Environment & process (safe use for scripts)

Function	Purpose	Example	Notes
`os.getenv('VAR', default=None)` ✅	Read env var	`db = os.getenv('DB_URL')`	Use for secrets/configs; prefer `os.environ.get()`
`os.environ` ✅	Mutable mapping of env vars	`os.environ['DEBUG']='1'`	Changes affect subprocesses launched after change
`os.chdir(path)` ✅	Change current working directory	`os.chdir('/app')`	`OSError` if path missing; consider `with` pattern via `pathlib`

Function	Purpose	Example	Notes
`os.getcwd()` ✅	Current working directory	`cwd = os.getcwd()`	Useful for logging & debugging
`os.getpid()` ✅	Current process id	`pid = os.getpid()`	Useful for pidfiles, supervisord
`os.kill(pid, sig)` ⚠️	Send signal to process	`os.kill(pid, signal.SIGTERM)`	Windows supports limited signals; permission required
`os.system(cmd)` ⚠️	Run shell command	`os.system('ls -la')`	Prefer `subprocess` module for safety and control

6. Atomic operations & renaming

Function	Purpose	Example	Notes
`os.rename(src, dst)` ✅	Rename/ move file or dir	`os.rename('tmp.txt','data.txt')`	Overwrites destination on POSIX; raises `FileExistsError` on some OSes
`os.replace(src, dst)` ✅	Atomic replace	`os.replace('tmp.new', 'app.conf')`	Guarantees replace (atomic on same filesystem) — safe write pattern

Safe write pattern (atomic save): 1. Write to temporary file `file.tmp` 2. `os.replace('file.tmp', 'file')`

This avoids partial writes and race conditions.

7. Pathlib note (modern alternative)

`pathlib.Path` offers object-oriented, cross-platform APIs that wrap many `os`/`os.path` functions. Example:

```
from pathlib import Path
p = Path('data') / 'file.txt'
p.mkdir(parents=True, exist_ok=True)
if p.exists():
    p.unlink()
```

Consider `pathlib` for new code.

---

8. Common exceptions & how to handle them

- `FileNotFoundError` — path missing (use `os.path.exists()` or handle exception)
- `FileExistsError` — attempt to create something that already exists
- `PermissionError` — insufficient privileges (avoid writing to system dirs, use correct UID)
- `OSError` — generic OS-level error (check `errno` for finer detail)

Pattern: prefer explicit checks where necessary, but also use EAFP (`try/except`) when race conditions exist.

```
# race-safe create-folder
import os
from errno import EEXIST
try:
    os.makedirs('cache', exist_ok=False)
except OSError as e:
    if e.errno != EEXIST:
        raise
```

---

9. Handy one-liners

- Create nested folder safely:

  ```
  os.makedirs('out/images', exist_ok=True)
  ```

- List only files in a dir:

  ```
  files = [f for f in os.listdir('data') if
  os.path.isfile(os.path.join('data', f))]
  ```

- Walk and find large files (>100MB):

```python
for root, _, files in os.walk('data'):
    for f in files:
        p = os.path.join(root, f)
        if os.path.getsize(p) > 100*1024*1024:
            print(p)
```

- Atomic write:

```python
with open('file.tmp','wb') as tmp: tmp.write(b'data')
os.replace('file.tmp','file.bin')
```

---

10. Visual summary (mental map)

- **Path helpers**: `join`, `exists`, `isfile`, `isdir`, `abspath`
- **Create/Remove**: `mkdir`, `makedirs`, `rmdir`, `remove`, `shutil.rmtree`
- **Traverse**: `listdir`, `scandir`, `walk`
- **Metadata**: `stat`, `chmod`, `utime`
- **Process/env**: `getenv`, `environ`, `chdir`, `getcwd`, `getpid`
- **Safe ops**: `replace` (atomic), `makedirs(..., exist_ok=True)`

If you want, I can: - Export this as a printable PDF or PNG infographic - Convert it to a one-page A4 layout with icons - Add color-coded sections for beginner vs advanced

Tell me which format you prefer.