you can apply them instantly.

. . .

📁 1. Directory Management

These commands help you **navigate** and **organize** file systems.

1. `os.getcwd()` : Get current working directory
2. `os.chdir(path)` : Change current directory
3. `os.listdir(path='.')` : List files in a directory
4. `os.mkdir(path)` : Create a new directory
5. `os.makedirs(path)` : Create nested directories
6. `os.rmdir(path)` : Remove a directory
7. `os.removedirs(path)` : Remove nested directories
8. `os.rename(src, dst)` : Rename a file or folder

💡 Real-World Example:

```
import os

# Automatically organize downloaded images into a new folder
downloads = os.path.expanduser("~/Downloads")
images_folder = os.path.join(downloads, "Images")

if not os.path.exists(images_folder):
    os.mkdir(images_folder)

for file in os.listdir(downloads):
    if file.endswith(".png") or file.endswith(".jpg"):
        os.rename(
            os.path.join(downloads, file),
            os.path.join(images_folder, file)
        )
```

. . .

📄 2. File Management

These are must-haves when you're working with logs, configs, and other local files.

1. `os.remove(path)` : Delete a file
2. `os.path.exists(path)` : Check if a file/directory exists
3. `os.path.isfile(path)` : Check if a path is a file
4. `os.path.isdir(path)` : Check if a path is a directory
5. `os.path.abspath(path)` : Get absolute path
6. `os.path.join(a, b)` : Join paths safely
7. `os.path.getsize(path)` : Get file size
8. `os.path.getmtime(path)` : Last modified time

💡 Real-World Example:

```
import os
import time

# Monitor log file and alert if too big
log_path = "/var/log/app.log"

if os.path.exists(log_path):
    size = os.path.getsize(log_path)
    if size > 10 * 1024 * 1024:  # 10MB
        print("⚠️ Log file too large. Consider rotating it.")
    else:
        print(f"✅ Log file size is okay: {size} bytes")
```

. . .

🌎 3. Environment Variables

Useful when managing credentials, configs, or Docker-style `.env` files.

1. `os.environ.get('KEY')` : Get environment variable
2. `os.environ['KEY'] = 'val'` : Set environment variable
3. `os.environ` : Get all environment variable (returns dict-like object)

💡 Real-World Example:

```
import os

db_url = os.environ.get("DATABASE_URL")
if not db_url:
    print("⚠️ DATABASE_URL not set!")
else:
    print(f"🔗 Connecting to DB at: {db_url}")
```

. . .

🔄 4. Process Management

Run shell commands, get process IDs, or fork processes.

1. `os.system('command')` : Run a shell command
2. `os.getpid()` : Current process ID
3. `os.getppid()` : Parent process ID
4. `os.fork()` : Fork process (Unix only)

5. `os._exit(status)` : Exit from a child process

💡 Real-World Example:

```
import os

print("📂 Listing current directory files:")
os.system("ls -l")
```

. . .

⚙️ 5. OS & Platform Info

Good for platform-specific logic (e.g., Windows vs. Unix).

1. `os.name` : Return OS name (`posix`, `nt`)
2. `os.uname()` : System info (Unix only)
3. `os.getuid()` : Get user ID (Unix only)

💡 Real-World Example:

```
import os

if os.name == 'nt':
    print("🪟 You're on Windows")
else:
    print("🐧 You're on Unix/Linux")
```

. . .

🔐 6. Permissions & Metadata

🔐 6. Permissions & Metadata

Manage access, ownership, and detailed info.

1. `os.chmod(path, mode)` : Change file permissions
2. `os.chown(path, uid, gid)` : Change owner (Unix)
3. `os.stat(path)` : Get file metadata

💡 Real-World Example:

```
import os
import stat

# Make a script executable
file = "deploy.sh"
st = os.stat(file)
os.chmod(file, st.st_mode | stat.S_IEXEC)
```

. . .

📄 7. Path Utilities (via os.path)

When working with file systems, paths can be tricky — especially when your code runs across different platforms (Linux, Windows, macOS). The `os.path` utilities help you manage file paths safely and consistently.

1. `os.path.isabs(path)` : Check if path is absolute
2. `os.path.splitext(path)` : Split filename and extension
3. `os.path.basename(path)` : Get base filename
4. `os.path.dirname(path)` : Get directory name from path
5. `os.path.normpath(path)` : Normalize a path (removes redundant `..`, `.`)

💡 Real-World Example:

```
import os

file_path = "../project/data/file.csv"

# Check if path is absolute
print("Is Absolute Path?", os.path.isabs(file_path))

# Normalize the path
normalized_path = os.path.normpath(file_path)
print("Normalized Path:", normalized_path)

# Split directory and file name
dir_name = os.path.dirname(file_path)
file_name = os.path.basename(file_path)

print("📁 Directory:", dir_name)
print("📄 File:", file_name)

# Get file extension
name, ext = os.path.splitext(file_path)
print("📝 Extension:", ext)
```

🧠 Use Case:

> You're building a CLI that accepts file inputs, and you want to:

- validate if the path is absolute,
- ensure the path is valid regardless of OS,
- separate the file name from its directory and extension for further processing like logging or uploading.

. . .

🖼️ Download the Visual Cheat Sheet

You can also **save or share this visual cheat sheet:**