Paths Cheat Sheet 🐍

References:

- https://docs.python.org/3/library/pathlib.html
- https://docs.python.org/3/library/os.path.html
- https://docs.python.org/3/library/sys.html#sys.path

Modules summary:

- `os`: Provides functions to interact with the operating system, including working with file paths in a platform-independent way.
- `sys`: Allows interaction with Python runtime and environment, can also be used to manipulate Python's search paths (i.e. `sys.path`).
- `pathlib`: A modern, object-oriented approach to handling paths. Easier to use and more readable than `os` for file and directory manipulations.

> Use `pathlib` for modern projects and `os` for compatibility with older Python versions.

Common Operations

---

1. Working with Paths

- **Constructing File Paths**
  - `os.path.join()`: Builds file paths in a platform-independent way.

```
import os
os.path.join("users", "data.txt")
```

  - `pathlib.Path()`: Use `/` to concatenate paths.

```
from pathlib import Path
Path("users") / "data.txt"
```

2. Checking if a Path Exists

- `os.path.exists()`: Checks if a file or directory exists.

```
os.path.exists("path/to/file")
```

- `pathlib.Path.exists()`: Object-oriented way to check path existence.

```
Path("path/to/file").exists()
```

3. Getting Path Components (Directory, Filename)

- `os.path.dirname()` **and** `os.path.basename()`: Extracts the directory and filename from a path.

```
os.path.dirname("/folder/subfolder/file.txt")  # Output:
"/folder/subfolder"
os.path.basename("/folder/subfolder/file.txt")  # Output:
"file.txt"
```

- `pathlib.Path()` **Attributes**:

  Use `parent` for directory and `name` for filename.

```
path = Path("/folder/subfolder/file.txt")
path.parent  # Output: "/folder/subfolder"
path.name    # Output: "file.txt"
```

4. Changing and Getting the Current Working Directory

- `os.getcwd()`: Gets the current working directory.

```
os.getcwd()
```

- `os.chdir()`: Changes the current working directory.

```
os.chdir("/new/directory")
```

- `pathlib.Path.cwd()`: Gets the current working directory.

```
Path.cwd()
```

---

5. Listing Files in a Directory

- `os.listdir()`: Lists all files and directories inside a specified directory.

```
os.listdir("path/to/directory")
```

- `pathlib.Path.iterdir()`: Iterates over files in a directory.

```
Path("path/to/directory").iterdir()
```

---

6. Adding a Directory to Python's Search Path

- `sys.path.append()`: Adds a new path to the Python module search path (useful when importing custom modules).

```
import sys
sys.path.append("/path/to/your/module")
```

---

7. Creating and Removing Directories

- `os.makedirs()`: Creates directories (including nested ones).

```
os.makedirs("new_directory/nested_directory", exist_ok=Tru
```

```
e)
```

- `os.rmdir()`: Removes an empty directory.

```
os.rmdir("empty_directory")
```

- `pathlib.Path.mkdir()`: Creates a directory.

```
Path("new_directory").mkdir(parents=True, exist_ok=True)
```

Conclusion

- `os`: Useful for interacting with file paths and directories in a cross-platform way.
- `sys`: Helpful for managing Python's environment and modifying the module search path.
- `pathlib`: A cleaner, more Pythonic alternative for handling file paths with improved readability.