The `os` Module

Reference: https://docs.python.org/3/library/os.html.

Use the `os` module perform command-line-style file and directory operations, and to access system environment variables.

> NOTE: Windows users may need to modify the filepaths below by using different slashes.

Directory Operations

Detect the path of the current working directory (in scripts, this reflects the dir from which the command is being run):

```
os.getcwd() #> '/Users/mjr/Desktop/my-dir'
```

In scripts, detect the path of the directory where the script file exists:

```
os.path.dirname(__file__))
```

Change directory:

```
os.chdir("/path/to/Desktop")
```

Make a new directory:

```
os.mkdir("/path/to/Desktop/my-dir")
```

List all files in a given directory:

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

File Operations

Delete a file:

```
os.remove("demofile.txt")
```

Detect whether a specific file exists:

```
os.path.isfile("/path/to/Desktop/some_file.txt") #> returns True or False
```

Compile file paths by joining the directory of the current file with a relative file path:

```
os.path.join(os.path.dirname(__file__), "../data/monthly_sales.csv")

# use `os.path.join` in conjunction with commas to standardize paths across
operating systems:
os.path.join(os.path.dirname(__file__), "..", "data", "monthly_sales.csv")
```

More examples of how to assemble file paths:

```
#
# /Users/mjr/Desktop/my-dir/paths.py
#
# Assumes the following files and directories exist on your computer:
#
#   /Users/mjr/Desktop/desktop_message.txt
#   /Users/mjr/Desktop/my-dir
#   /Users/mjr/Desktop/my-dir/paths.py (this file)
#   /Users/mjr/Desktop/my-dir/my_message.txt
#   /Users/mjr/Desktop/my-dir/subdir/
#   /Users/mjr/Desktop/my-dir/subdir/other_message.txt
#

import os

# what's the name of this file?
print(__file__)
#> /Users/mjr/Desktop/my-dir/paths.py

# what directory is this file in?
print(os.path.dirname(__file__))
#> /Users/mjr/Desktop/my-dir

# examples of constructing paths to the various files...

print(os.path.join(os.path.dirname(__file__), "my_message.txt"))
#> /Users/mjr/Desktop/my-dir/my_message.txt
```

```
print(os.path.join(os.path.dirname(__file__), "subdir"))
#> /Users/mjr/Desktop/my-dir/subdir

print(os.path.join(os.path.dirname(__file__), "subdir", "other_message.txt"))
#> /Users/mjr/Desktop/my-dir/subdir/other_message.txt

print(os.path.join(os.path.dirname(__file__), "..", "desktop_message.txt"))
#> /Users/mjr/Desktop/my-dir/../desktop_message.txt

print(os.path.isfile(os.path.join(os.path.dirname(__file__), "..",
"desktop_message.txt")))
#> True
```

Environment Variables

> Prerequisite: Environment Variables

Get the entire environment:

```
import os

my_env = os.environ

print("------------")
print(type(my_env)) #> <class 'os._Environ'>
print(my_env)

# can be converted to a dictionary:
print("------------")
print(type(dict(my_env))) #> <class 'dict'>
```

Get a specific environment variable (e.g. `MY_SECRET_MESSAGE`, only after you have set it):

```
# using a dictionary-like approach:
my_var = os.environ["MY_SECRET_MESSAGE"]
print(my_var) #> SecretPassword123

# using a getter function:
my_var = os.environ.get("MY_SECRET_MESSAGE")
print(my_var) #> SecretPassword123

# using the newer getter function (recommended):
my_var = os.getenv("MY_SECRET_MESSAGE", default="This is a default / fallback
message.")
print(my_var) #> SecretPassword123
```