Files 101!

File Systems

- File System - a subsystem of your computer's operating system

- Made of *directories* and *files* that are *persisted* to a storage device
  - Directories are often referred to and visualized as Folders in user interfaces.
  - A computer/phone's storage device is different from memory in that when a device is shutdown the storage is maintained, but memory is reset/lost.
  - Why the difference? Efficiency and speed! Storage is slower than memory.

- Directories are lists of files and... other *directories*.
  - Thus forming a hierarchy.
  - A directory's ability to relate with another directory makes it a *recursive structure(!)*

- Directories and files have textual names *for humans*
  - Under the hood, data structures map these textual names to indexed addresses.

Paths

- When a program is running it is called a *process*
- Each *process* is running in "working directory"
  - When python runs in your workspace, this is the workspace's directory

Opening a Text File for Reading

- There are many file types (plain text, rich text, photos, programs)
  - Plain text documents have no formatting besides plain-old character data.
  - Rich text documents are those like Word documents which encode formatting
  - Our focus is plain text. Their direct translation to string values makes it easiest to work with.

- Plain text files can have different encodings. We'll use UTF-8.
  - Why different encodings? Related to the earlier idea of character codes.
  - Short story: when space was more limited, efficient encoding mattered more. These days UTF-8 is a universal standard that works for most every written language, *as well as emoji*.

Opening a Plain Text file with UTF-8 Encoding

- Python's built-in `open` function opens a file and results in an IO object
  - IO is an abbreviation for Input/Output

- Example:

```
path = "some/path/to/file.txt"
mode = "r"  # Read-only
io_handle = open(path, mode, encoding="utf8")
```

- The last parameter is a *keyword argument* and noticeably strange.
  - It is beyond our concern right now, but the short story is the open function has a lot of optional parameters and by specifying *encoding* we are giving a specific one.

Reading the Contents of a File (1 / 2)

- A readable IO Handle object's `read` method returns the file's content.
  - The `read` method returns a `str`.

- Example:

```
contents = io_handle.read()
```

- Warning: An I/O Handle is much like an iterator in that it is stateful.
  - Generally: Storage I/O is a side-effect since it's either reading from or writing to storage outside of your program.
  - Subsequent calls to read will result in an empty string because the handle has reached the end of file.

Reading the Contents of a File (2 / 2)

- A readable IO Handle to a plain text file is is ***iterable***
  - Each *line*, including the new line character at its end, is iterated over
  - Can be used with for..in
- Example:

```
for line in io_handle:
    ...
```

- Warning: The new-line character "\n" is included at the end of each line.
  - The str type's strip() method will get rid of it, as well as any other spaces on either end of the string.

Closing an I/O Handle

- You must close an I/O handle once your need for it expires

```
io_handle.close()
```

- Why we call these *handles* is they are *handles on a resource*
  - The operating system keeps track of what files are currently open and serves as a brokerage between your process and the file system.
  - Closing an IO Handle frees up resources. If you had *lots* of file handles open and failed to close them your program would eventually error out.