Python File IO

Table of contents

Opening a file

```
file = open("path", "<mode>")
```

Mode	Description
`r`	read
`w`	write (creates or **overwrites**)
`a`	append (creates or adds to the end)

Reading

`file.readline()` returns one line (including its trailing `\n`, except possibly the last line of the file), then `''` forever once exhausted. A for-loop iterates line by line and is the idiomatic way to consume a whole file:

```
with open("path", "r") as file:
    for line in file:
        ...
```

Reading always gives back **strings** — cast (`int(...)`, `float(...)`, …) as needed.

A file-pointer only moves forward: once a `for` loop (or repeated `readline()` calls) has consumed a file, iterating again yields nothing further, unless the file is reopened (or the pointer is seeked back to the start).

Closing

`file.close()` releases the file; forgetting to close leaves it vulnerable to side effects (missing or extra data). `with open(...) as file:` closes it automatically — even if the block raises an exception — so it's preferred over manual `open`/`close`.

Writing and appending

```
with open("path", "w") as file:
    file.write("a single string\n")
    file.writelines(["one string\n", "per element\n"])
```

`"w"` overwrites any existing file; `"a"` instead appends to the end without touching existing content.