



A `Path` is only a name until you read or write it, and text files should always be opened with `encoding="utf-8"`. Examples run in an empty folder, so each file is created before it is read.

READ AND WRITE IN ONE LINE

```
from pathlib import Path
p = Path("a.txt")

p.write_text("a\nb\n", encoding="utf-8")
→ 4
creates or replaces the file; returns chars written

p.read_text(encoding="utf-8") → 'a\nb\n'
the whole file as one string

text = p.read_text(encoding="utf-8")
text.splitlines()
→ ['a', 'b']
lines without the \n

Path("b.bin").write_bytes(b"\x00\xff")
→ 2
bytes in, no encoding

Path("b.bin").read_bytes() → b'\x00\xff'
```

OPEN() AND WITH

```
from pathlib import Path
p = Path("a.txt")

with open(p, encoding="utf-8") as f:
    text = f.read()
the file closes at the end, even after an error

with open(p, encoding="utf-8") as f:
    for line in f:
        print(line.rstrip("\n"))
one line at a time: fine for huge files

with open(p, encoding="utf-8") as f:
    first = f.readline()
first
→ 'a\n'
the next line, \n included

with open("log.txt", "a",
          encoding="utf-8") as f:
    f.write("start\n")
    print("ready", file=f)
"a" adds to the end; write() adds no \n, print() does

errors="replace"
open() option: bad bytes no longer raise
```

MODES

"r"	read (default); the file must exist
"w"	write; empties the file first
"a"	append to the end
"x"	create; FileExistsError if it exists
"r+"	read and write, no emptying
"rb", "wb"	binary: bytes, never encoding=

HANDY HELPERS

```
Path(__file__).parent / "data.txt"
a file next to the script, wherever you run from

with TemporaryDirectory() as tmp:
    print(Path(tmp).exists())
a scratch folder, deleted afterwards (tempfile)

os.path.join("data", "a.txt")
older code; Path / does the same
```

PATH PARTS

```
from pathlib import Path
p = Path("data/sales/report.csv")

p.name → 'report.csv'
p.stem → 'report'
the name without its suffix

p.suffix → '.csv'

p.parent.as_posix() → 'data/sales'
the folder it lives in

p.parts → ('data', 'sales', 'report.csv')

p.with_suffix(".json").name
→ 'report.json'
/ joins parts with the right separator for the OS

Path("a.tar.gz").suffixes
→ ['.tar', '.gz']

Path.cwd(), Path.home()
where you are, your home folder

p.resolve() the full absolute path
```

CHECK, LIST AND TIDY

```
d = Path("data"), holding a.csv, b.csv and sub/x.md

d.is_dir() → True

(d / "a.csv").is_file() → True

(d / "zz.csv").exists() → False

(d / "a.csv").stat().st_size → 1
size in bytes

sorted(q.name for q in d.glob("*.csv"))
→ ['a.csv', 'b.csv']
glob order isn't fixed: sort it

sorted(q.name for q in d.iterdir())
→ ['a.csv', 'b.csv', 'sub']
files and folders inside

[q.as_posix() for q in d.rglob("*.md")]
→ ['data/sub/x.md']
rglob searches subfolders too

Path("out").mkdir(exist_ok=True)
no error if it exists; parents=True makes a/b/c

(d / "b.csv").rename(d / "c.csv")
rename or move

shutil.copy(d / "a.csv", "out")
copy a file (import shutil)

(d / "c.csv").unlink(missing_ok=True)
delete a file

shutil.rmtree("out")
delete a folder and everything in it
```

CSV

```
import csv
rows = [{"name", "age"}, ["Ada", 36]]

with open("p.csv", "w", newline="",
          encoding="utf-8") as f:
    csv.writer(f).writerows(rows)
newline="" stops blank rows on Windows

with open("p.csv", newline="",
          encoding="utf-8") as f:
    back = list(csv.DictReader(f))
back
→ [{'name': 'Ada', 'age': '36'}]
header row = keys; values come back as text

csv.reader(f)
plain lists, header row included
```

JSON

```
import json
from pathlib import Path
p = Path("cfg.json")
cfg = {"name": "Ada", "tags": ["x"]}

p.write_text(json.dumps(cfg, indent=2),
              encoding="utf-8")
indent=2 makes it readable

json.loads(p.read_text(encoding="utf-8"))
→ {'name': 'Ada', 'tags': ['x']}
loads/dumps work on strings

with open(p, encoding="utf-8") as f:
    cfg = json.load(f)
load/dump work on open files

json.dumps({"a": None, "b": True})
→ '{"a": null, "b": true}'
None and True become null and true

json.dumps("é", ensure_ascii=False)
→ '"é"'
keeps accents as they are, not \u00e9
```

BIG BINARY FILES

```
with open("b.bin", "rb") as f:
    while chunk := f.read(65536):
        print(len(chunk))
read in pieces, never all at once

f.seek(0), f.tell()
jump to a byte; where you are now
```

ERRORS YOU WILL MEET

```
open("nope.txt")
→ raises FileNotFoundError
wrong name, or python runs from another folder

open("a.txt", "x")
→ raises FileExistsError
"x" refuses to overwrite

Path("data").mkdir()
→ raises FileExistsError
add exist_ok=True

Path("data").rmdir() → raises OSError
only empty folders; shutil.rmtree for the rest

PermissionError
open in another program, or no rights there
```

WATCH OUT

```
open("data.txt")
no encoding: the Windows default is often not UTF-8

open("notes.txt", "w")
"w" wipes the file the moment it opens; use "a" to add

open("C:\new\table.csv")
\n and \t turn into a newline and a tab: use r"..." or /

text = f.read() # after the with
ValueError: the file is closed. Read inside the block

f.read(); f.read()
the second read gives "": you are at the end. f.seek(0)

open("data.txt") # next to the script
relative paths start where you run python, not at the script

if line == "yes": # line from a file
never true: line ends in "\n". Compare line.strip()
```