

[220 / 319] Files

Meena Syamkumar
Andy Kuemmel

Relation to Recent Topics...

1

file objects are a type of iterator (lecture 25)

2

exceptions occur a lot with files (lecture 26)
not found, out of space, no permissions, etc

Learning Objectives Today

Basic file interactions

- opening/closing
- reading/writing

OS module

- listdir, mkdir, exists, isdir, isfile, join

File exceptions

Encodings

File objects

main.py:

```
f = open(path)
```

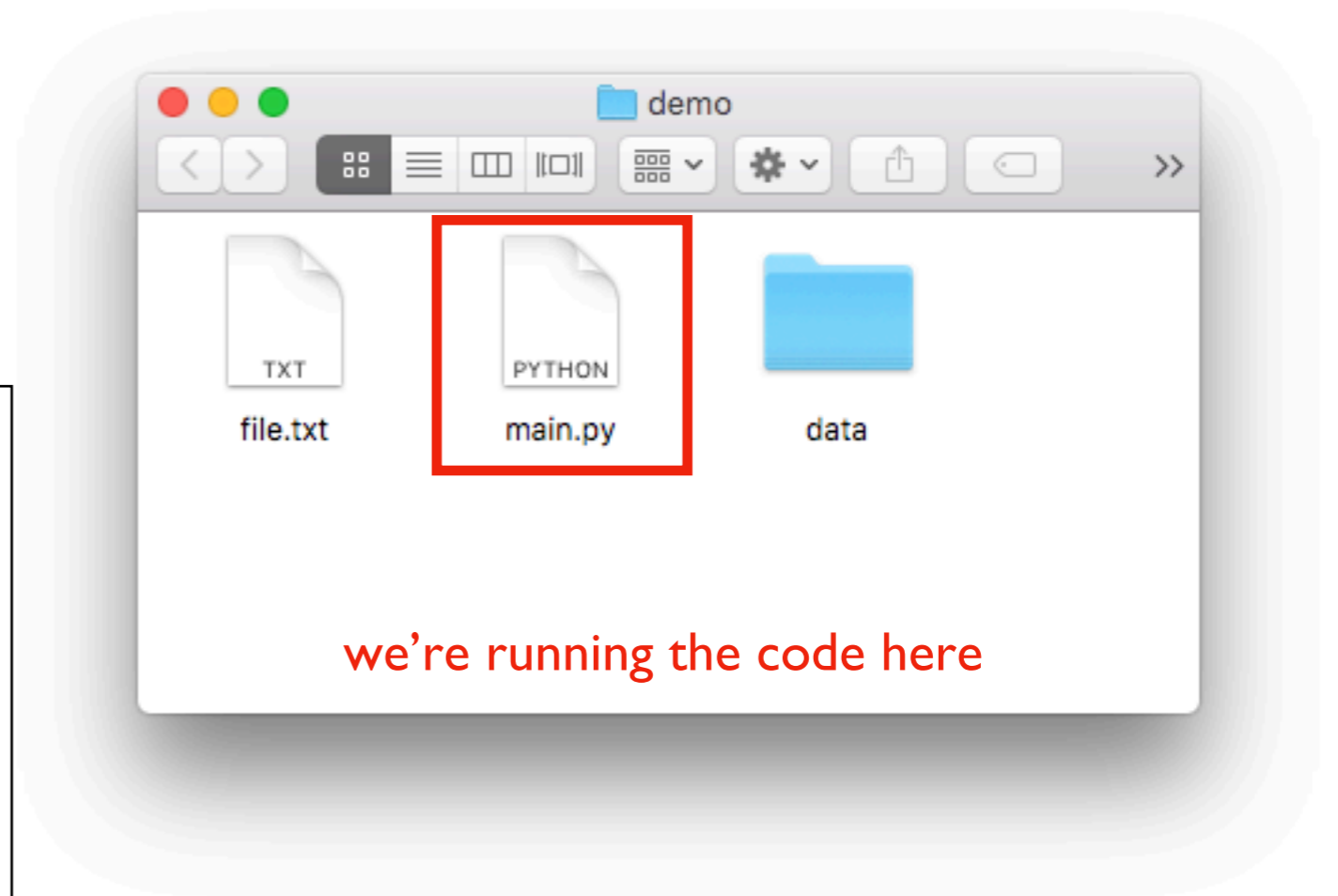
file object

file path

```
# read data from f  
# OR  
# write data to f
```

```
f.close()
```

built-in open function



we're running the code here

File objects

main.py:

```
f = open("file.txt")
```

built-in open function

file object

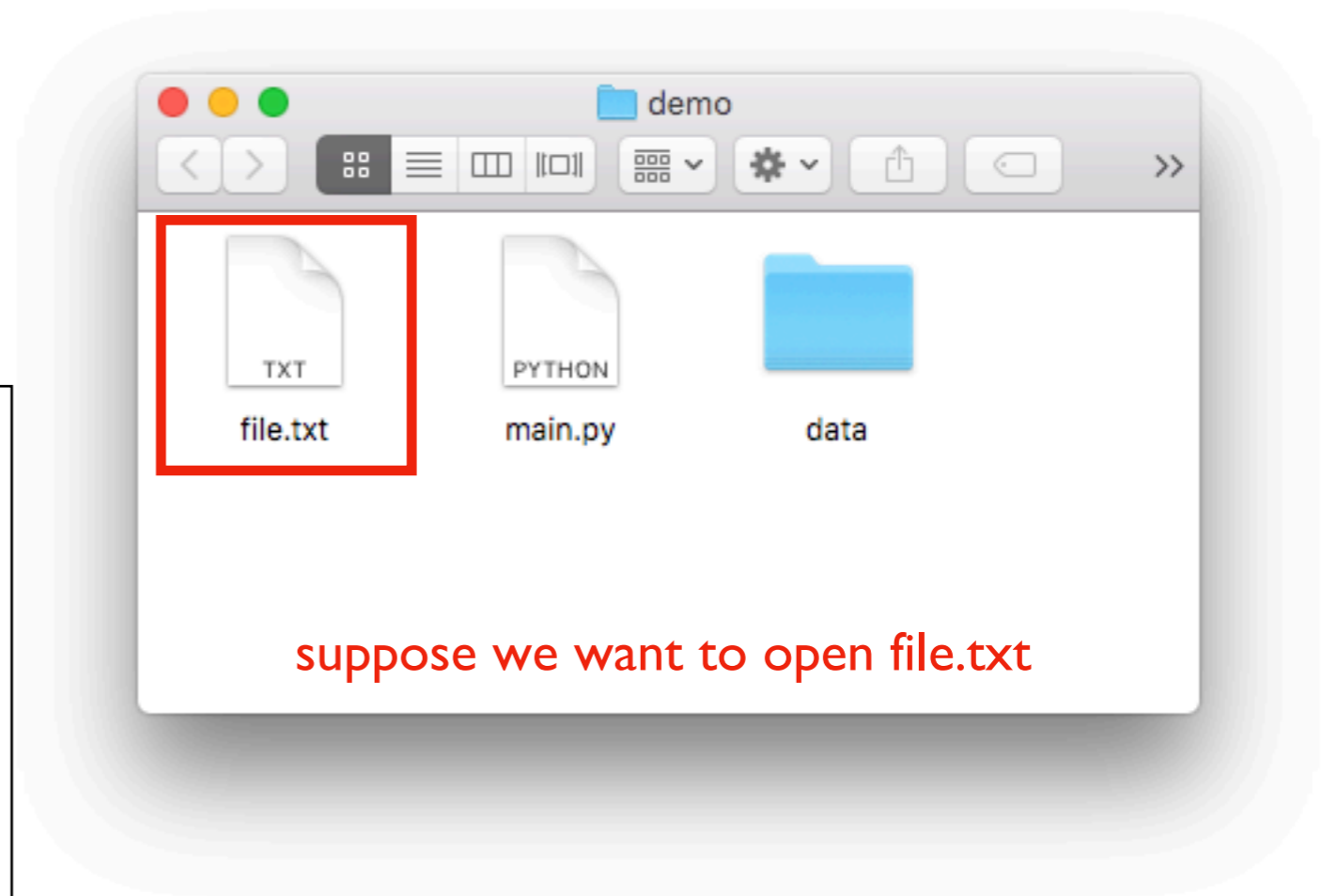
file path

```
# read data from f
```

```
# OR
```

```
# write data to f
```

```
f.close()
```



File objects

main.py:

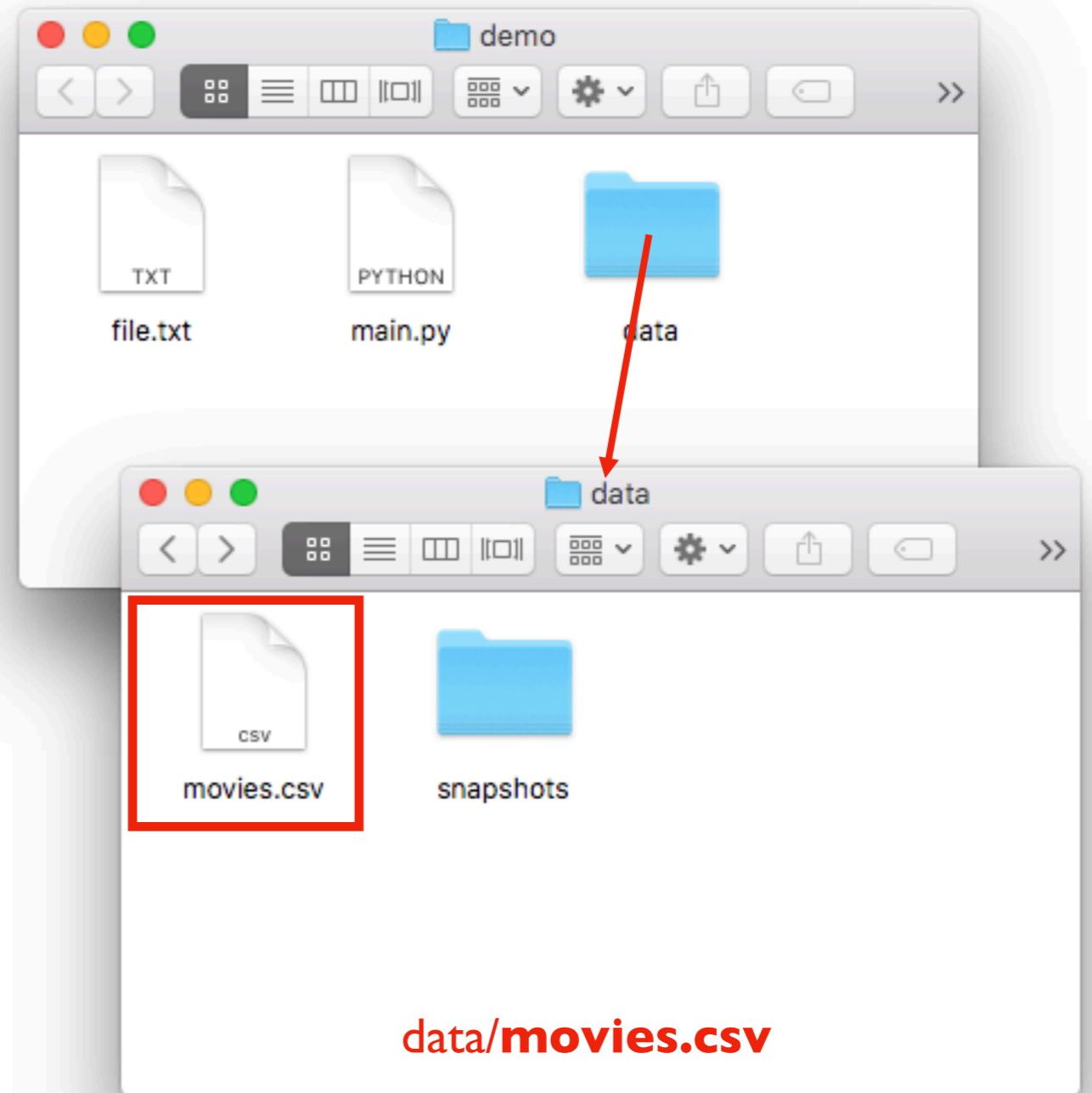
```
f = open(  
    "data/movies.csv")
```

file object

file path

```
# read data from f  
# OR  
# write data to f
```

```
f.close()
```



File objects

main.py:

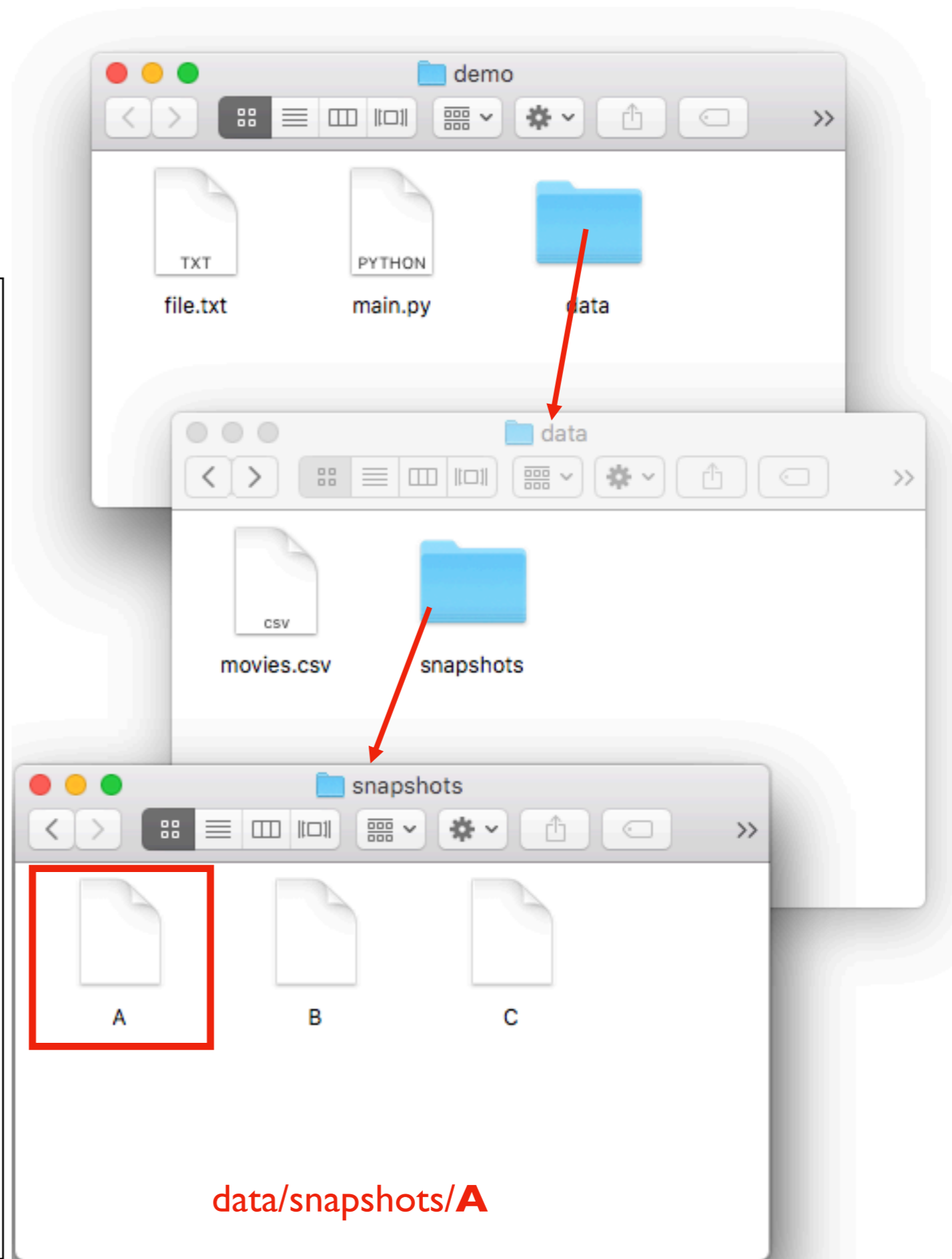
```
f = open(  
    "data/snapshots/A" )
```

file object

file path

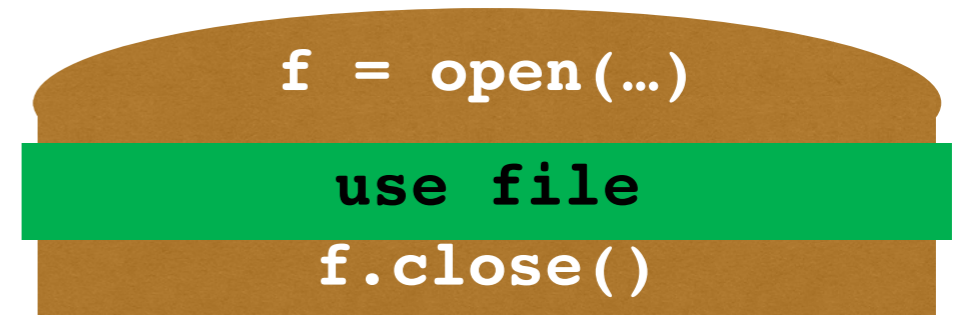
```
# read data from f  
# OR  
# write data to f
```

```
f.close()
```



File objects

imagine a *file object* as a *sandwich*...



main.py: `f = open("file.txt")`

```
# read data from f  
# OR  
# write data to f
```

using file

```
f.close()
```

cleanup

Reasons for closing

- avoid data loss
- limited number of open files

Learning Objectives Today

Basic file interactions

- opening/closing
- reading/writing

OS module

- listdir, mkdir, exists, isdir, isfile, join

File exceptions

Encodings

Reading a file

```
f = open("file.txt")
```

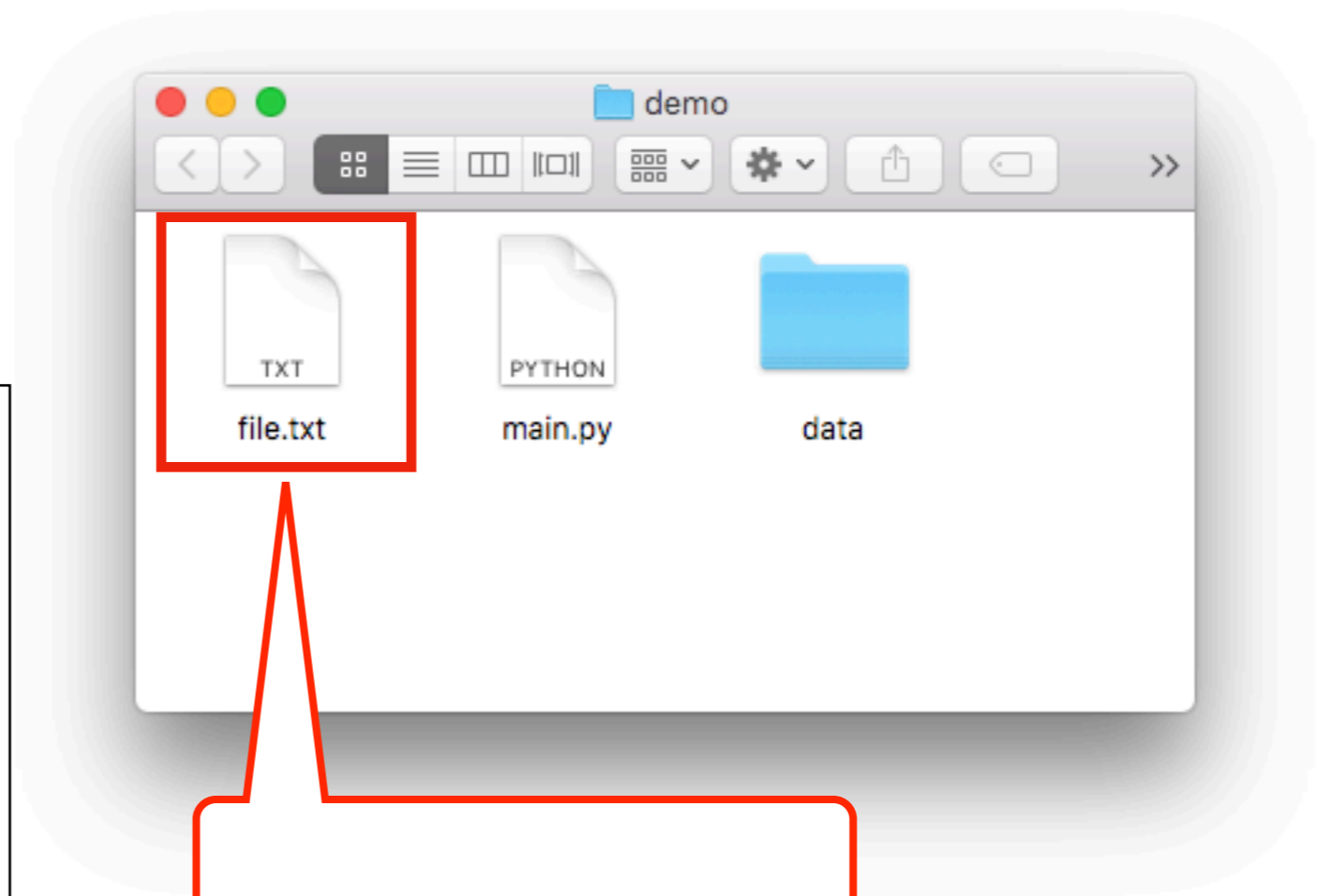
```
data = f.read()
```

Option 1

```
print(data)
```

data is: "I promise\n to always\nclose my files"

```
f.close()
```



I promise
to always
close my files

read() method

- fetch entire file contents
- return as a string

Reading a file

```
f = open("file.txt")
```

```
# read data from f
```

```
# OR
```

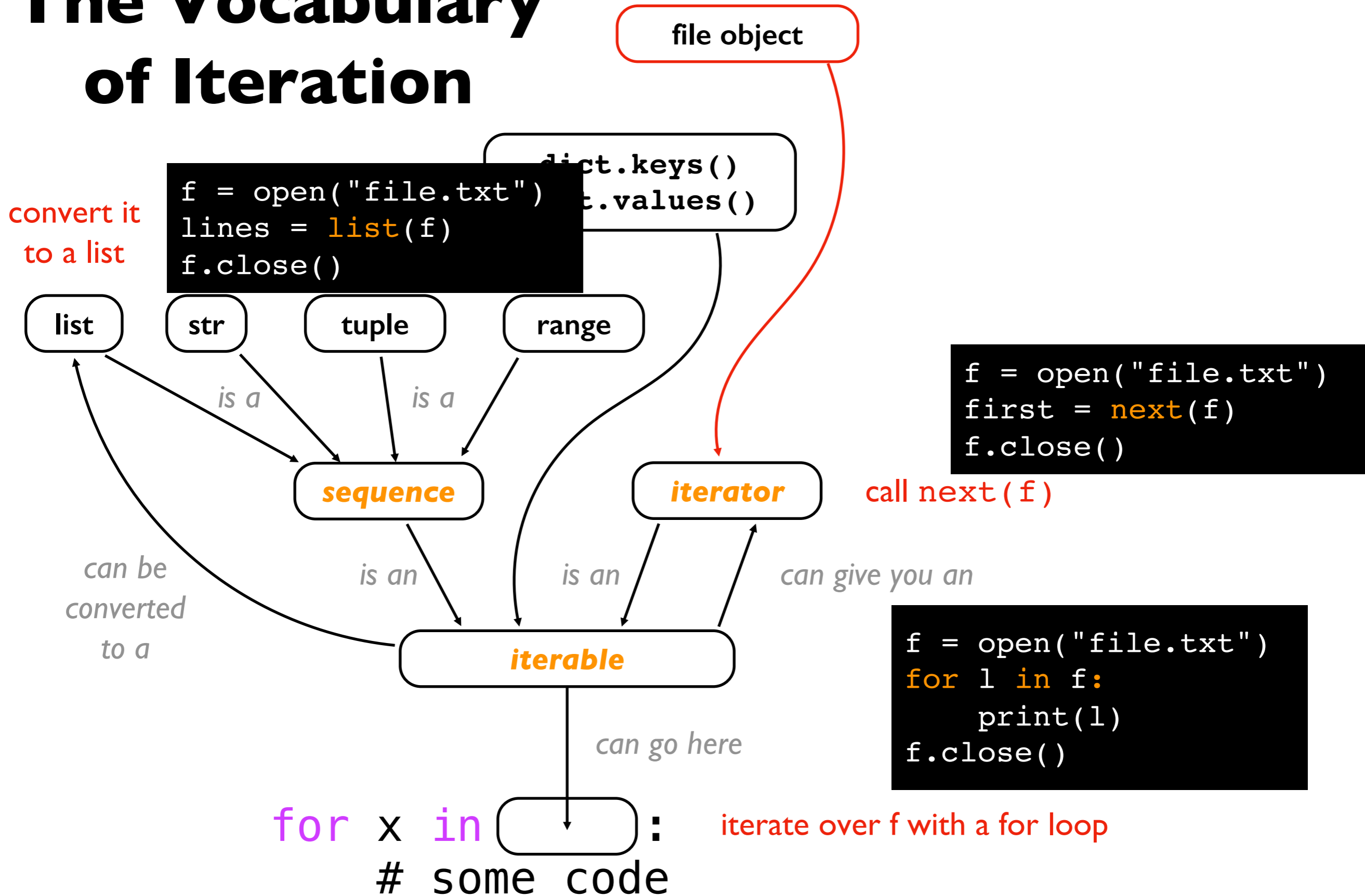
```
# write data to f
```

```
f.close()
```

Option 2


file objects are iterators!

The Vocabulary of Iteration



Write a file

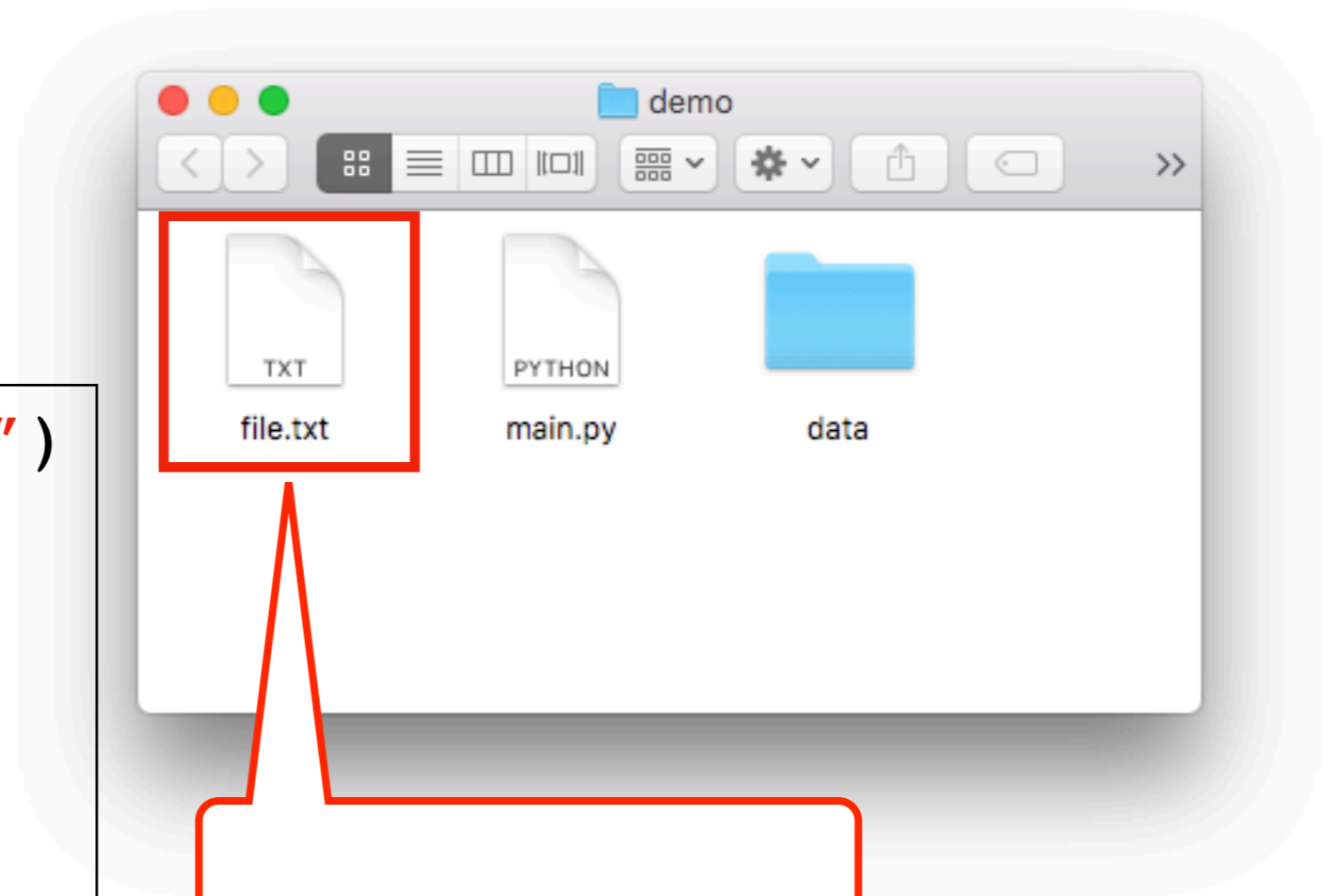
“w” mode indicates we want to write to this file



```
f = open("file.txt", "w")
```

```
f.write("hello")  
f.write(" world\n")  
f.write("!!!!\n")
```

```
f.close()
```



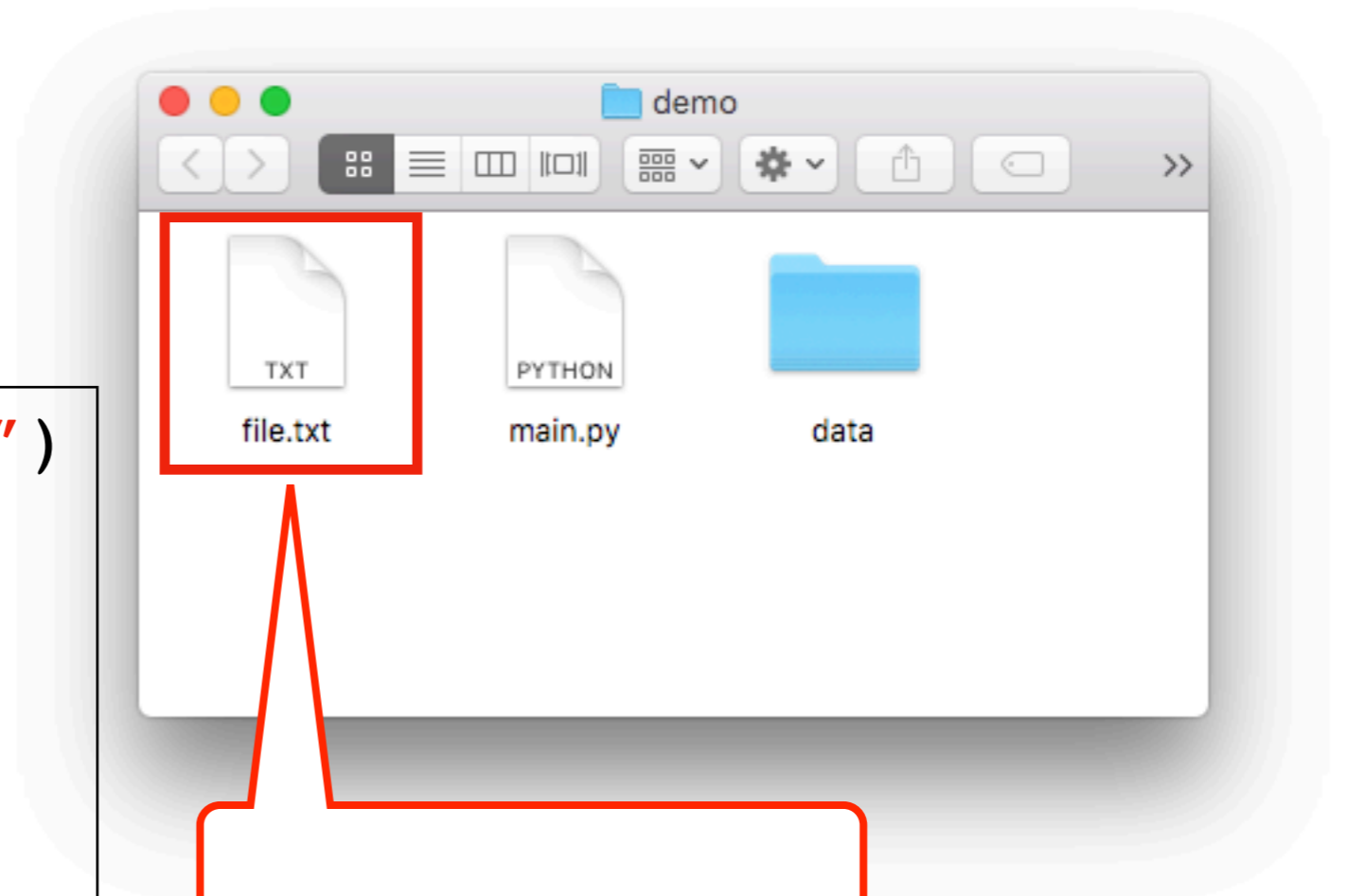
I promise
to always
close my files

let's run it!

Write a file

“w” mode indicates we want to write to this file

```
f = open("file.txt", "w")  
  
f.write("hello")  
f.write(" world\n")  
f.write("!!!!\n")  
  
f.close()
```



opening with “w” is dangerous. It immediately wipes out your file.

(or creates a new one if there isn't already a file.txt)

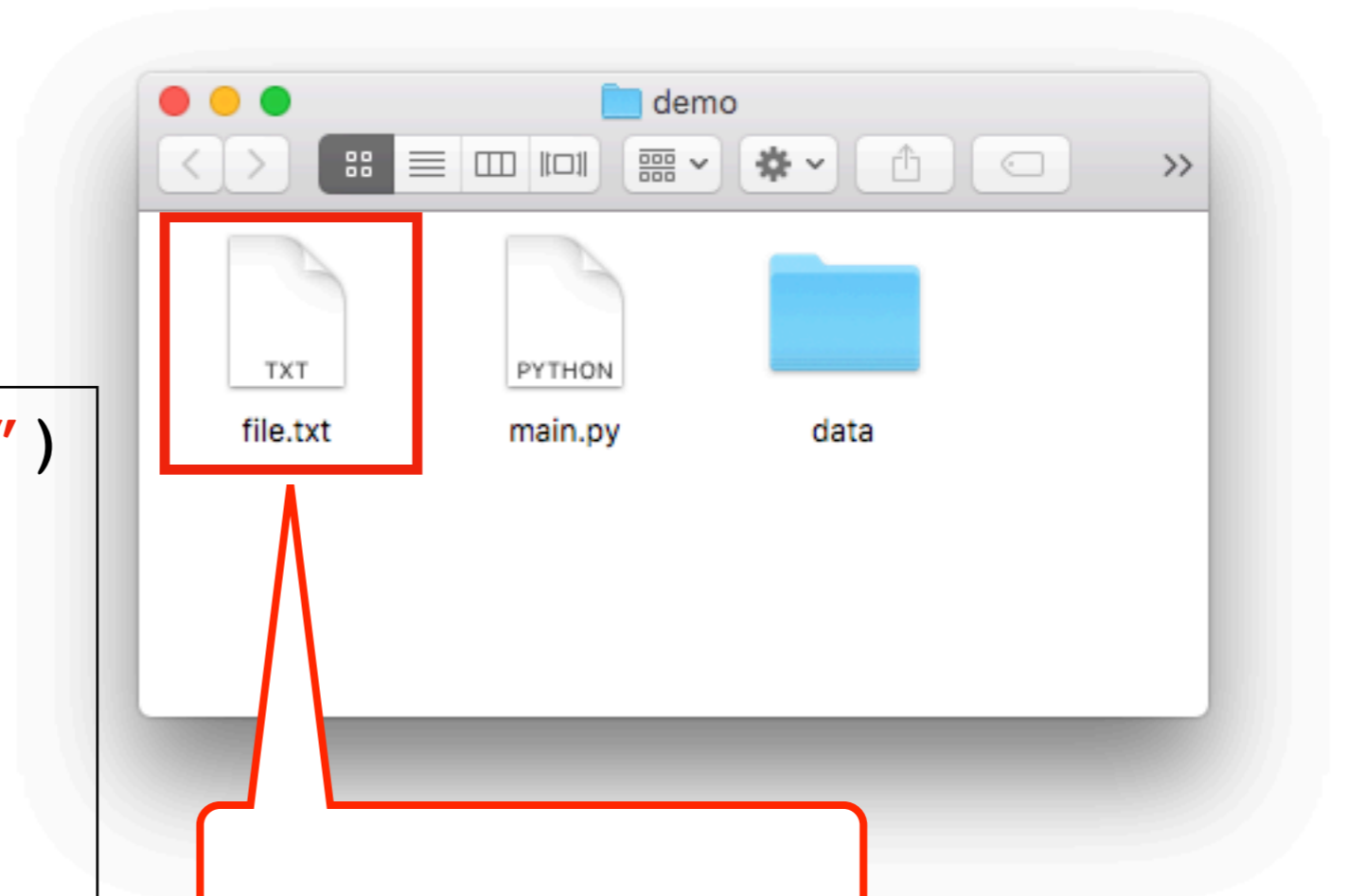
Write a file

“w” mode indicates we want to write to this file

```
f = open("file.txt", "w")
```

```
f.write("hello")  
f.write(" world\n")  
f.write("!!!!\n")
```

```
f.close()
```



hello

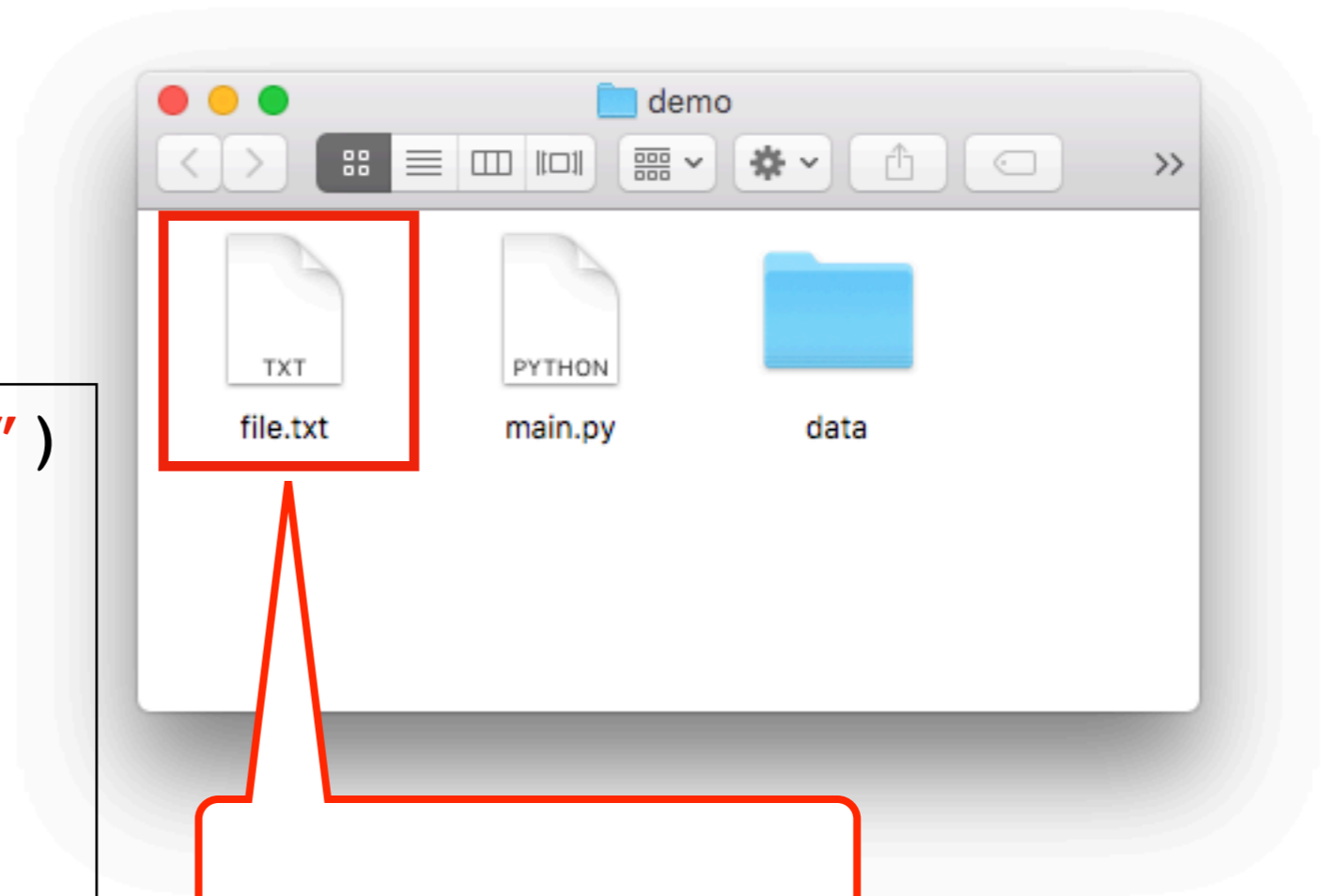
Write a file

“w” mode indicates we want to write to this file

```
f = open("file.txt", "w")
```

```
f.write("hello")  
f.write(" world\n")  
f.write("!!!!\n")
```

```
f.close()
```



hello world

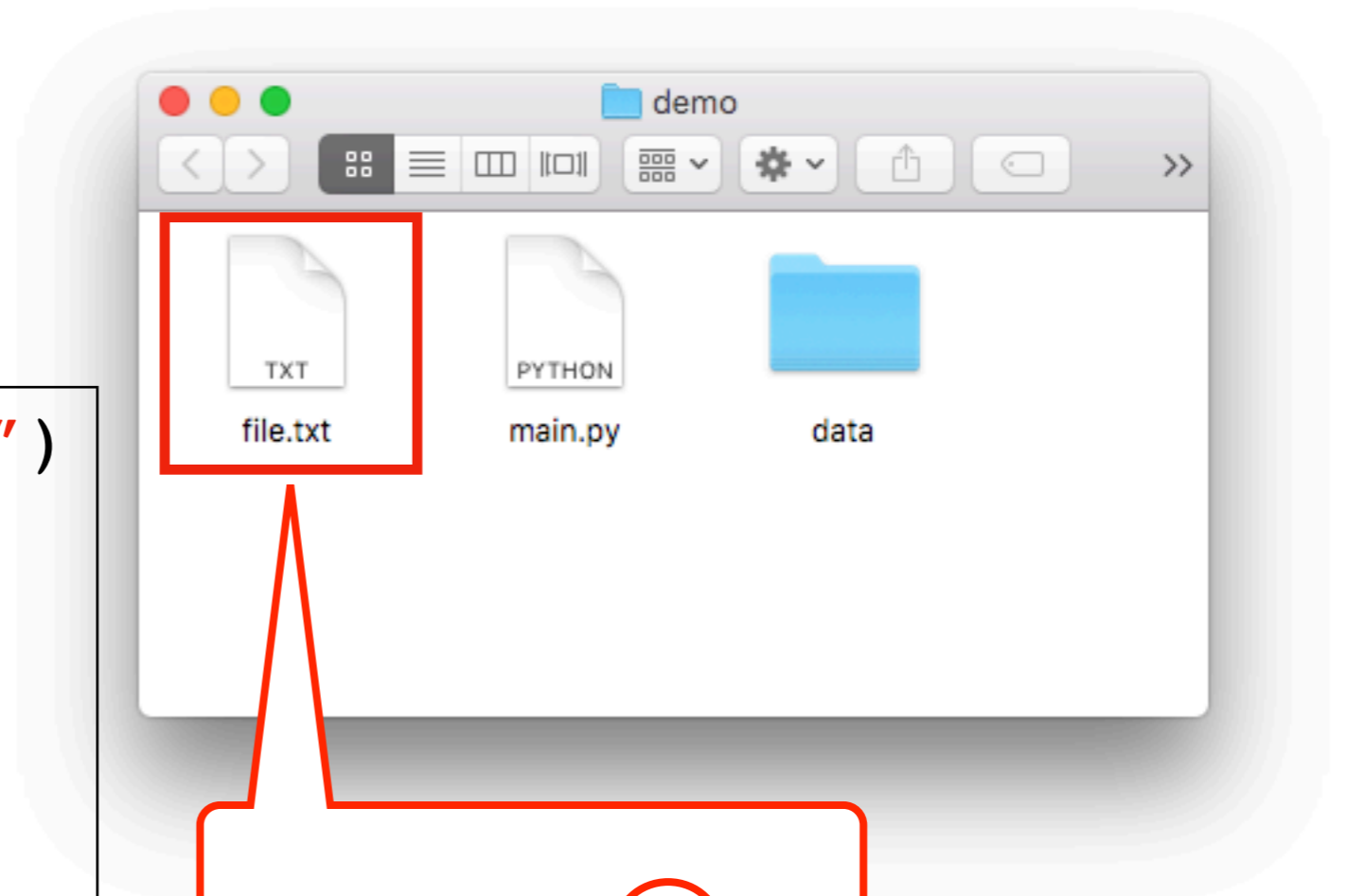
Write a file

“w” mode indicates we want to write to this file

```
f = open("file.txt", "w")
```

```
f.write("hello")  
f.write(" world\n")  
f.write("!!!!!\n")
```

→ `f.close()`



hello world
!!!!!!

be careful with newlines
(write doesn't add them like print does)

Learning Objectives Today

Basic file interactions

- opening/closing
- reading/writing

OS module

- `listdir`, `mkdir`, `exists`, `isdir`, `isfile`, `join`

File exceptions

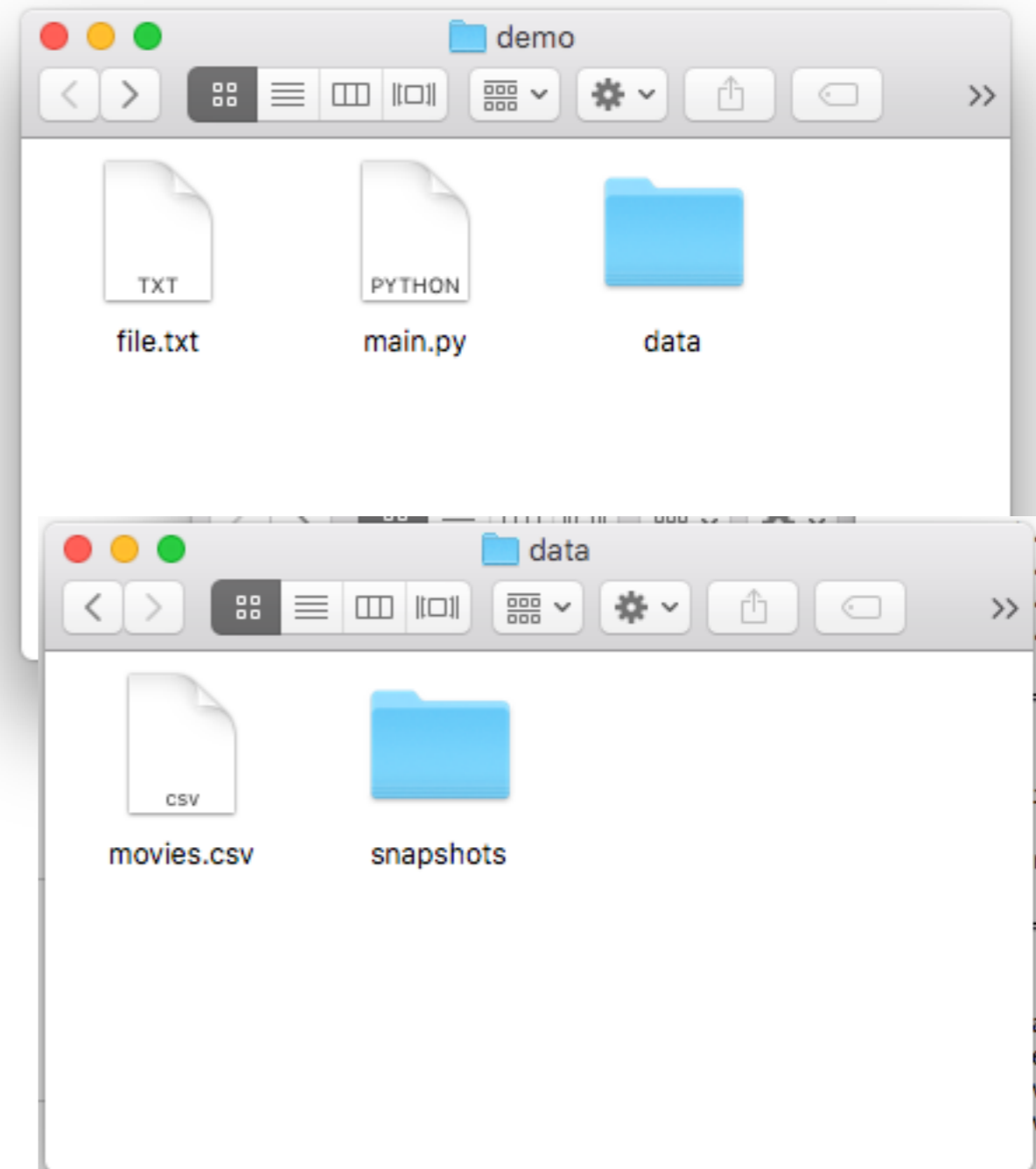
Encodings

OS Module (Operating System)

Many functions in `os` and `os.path` for working w/ files

- **`os.listdir`**
- `os.mkdir`
- `os.path.exists`
- `os.path.isfile`
- `os.path.isdir`
- `os.path.join`

```
>>> import os
>>> os.listdir(".")
["file.txt", "main.py", "data"]
>>> os.listdir("data")
["movies.csv", "snapshots"]
```

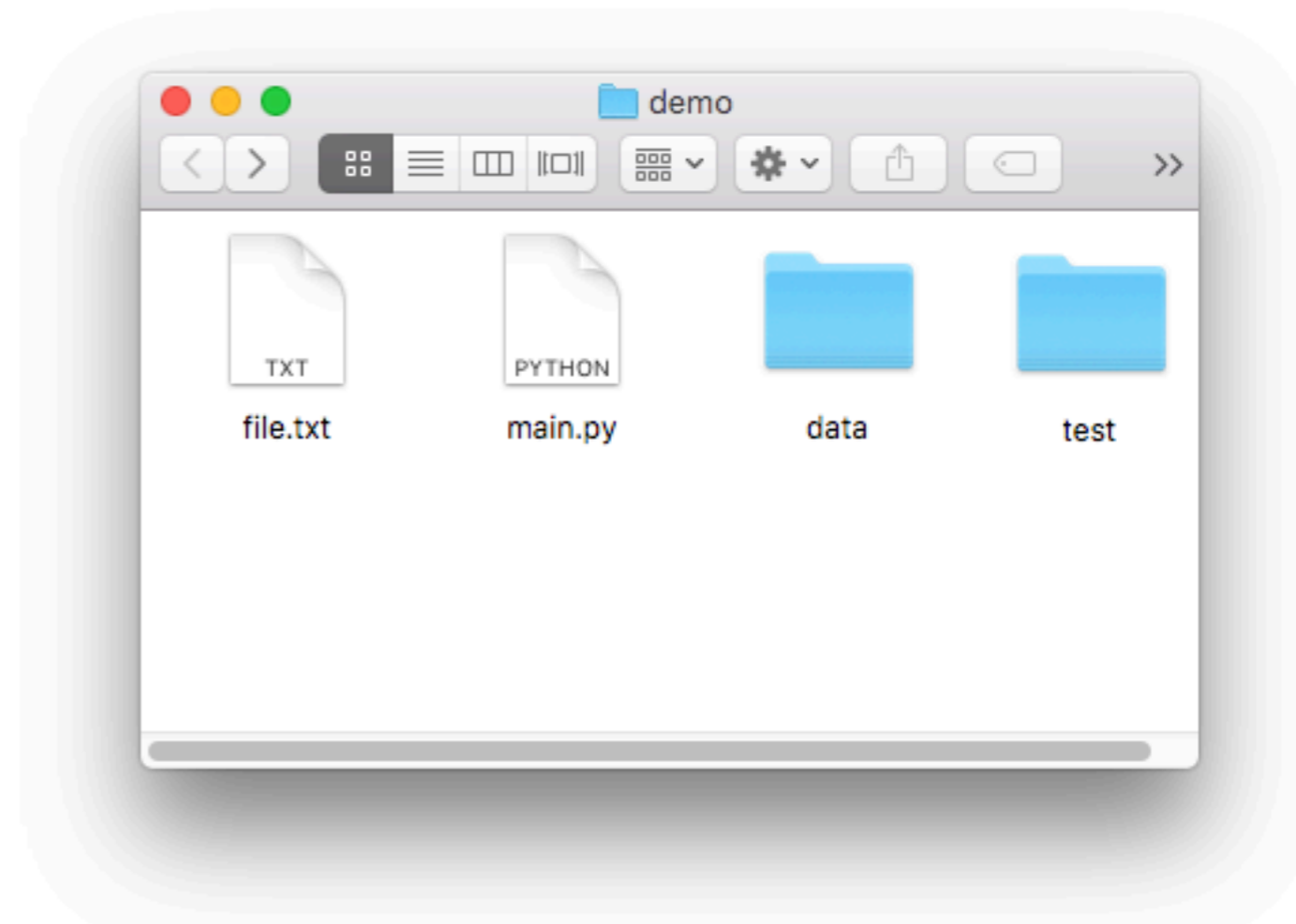


OS Module (Operating System)

Many functions in `os` and `os.path` for working w/ files

- `os.listdir`
- **`os.mkdir`**
- `os.path.exists`
- `os.path.isfile`
- `os.path.isdir`
- `os.path.join`

```
>>> import os  
>>> os.mkdir("test")
```

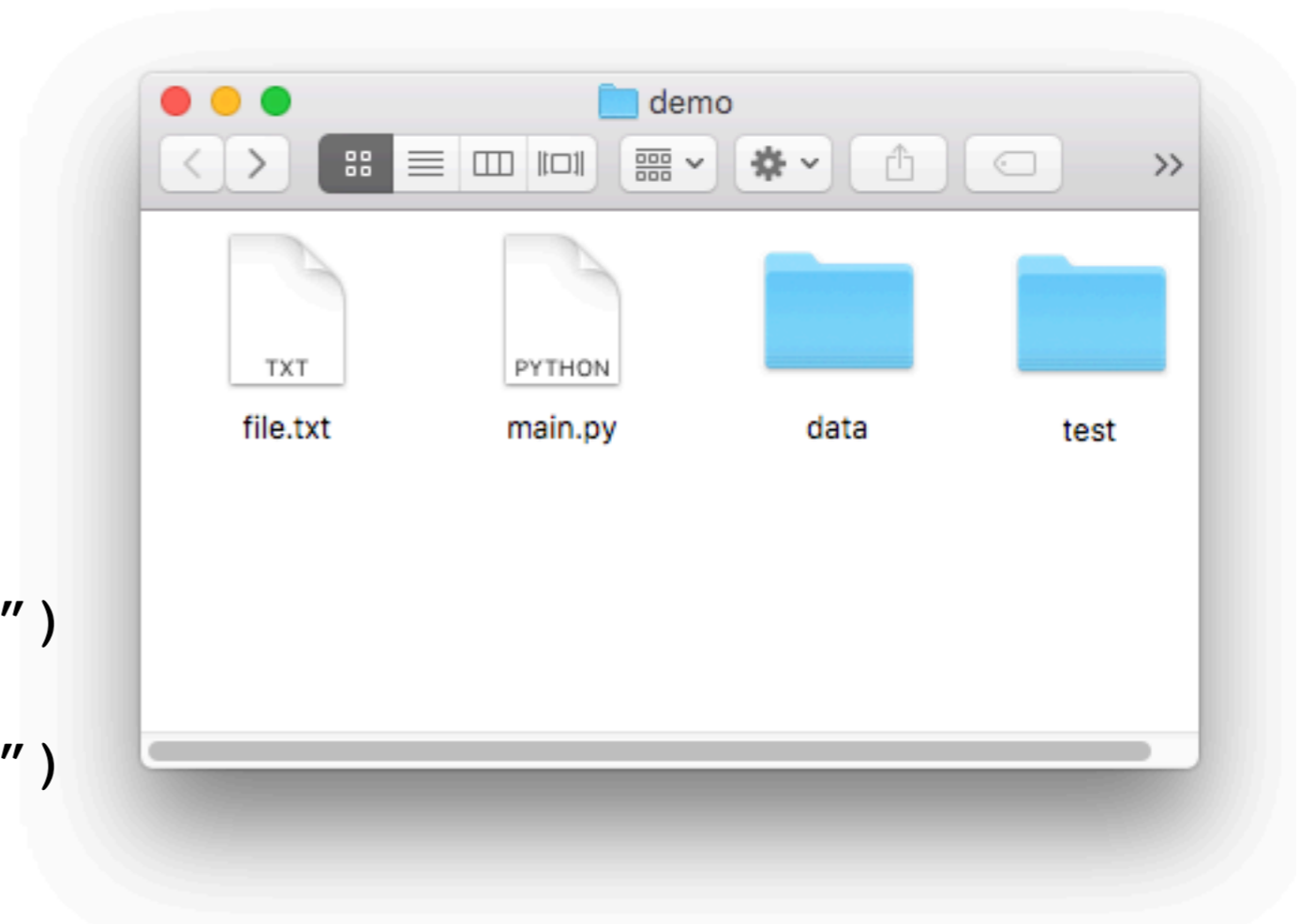


OS Module (Operating System)

Many functions in `os` and `os.path` for working w/ files

- `os.listdir`
- `os.mkdir`
- **`os.path.exists`**
- `os.path.isfile`
- `os.path.isdir`
- `os.path.join`

```
>>> import os
>>> os.path.exists("file.txt")
True
>>> os.path.exists("haha.txt")
False
```

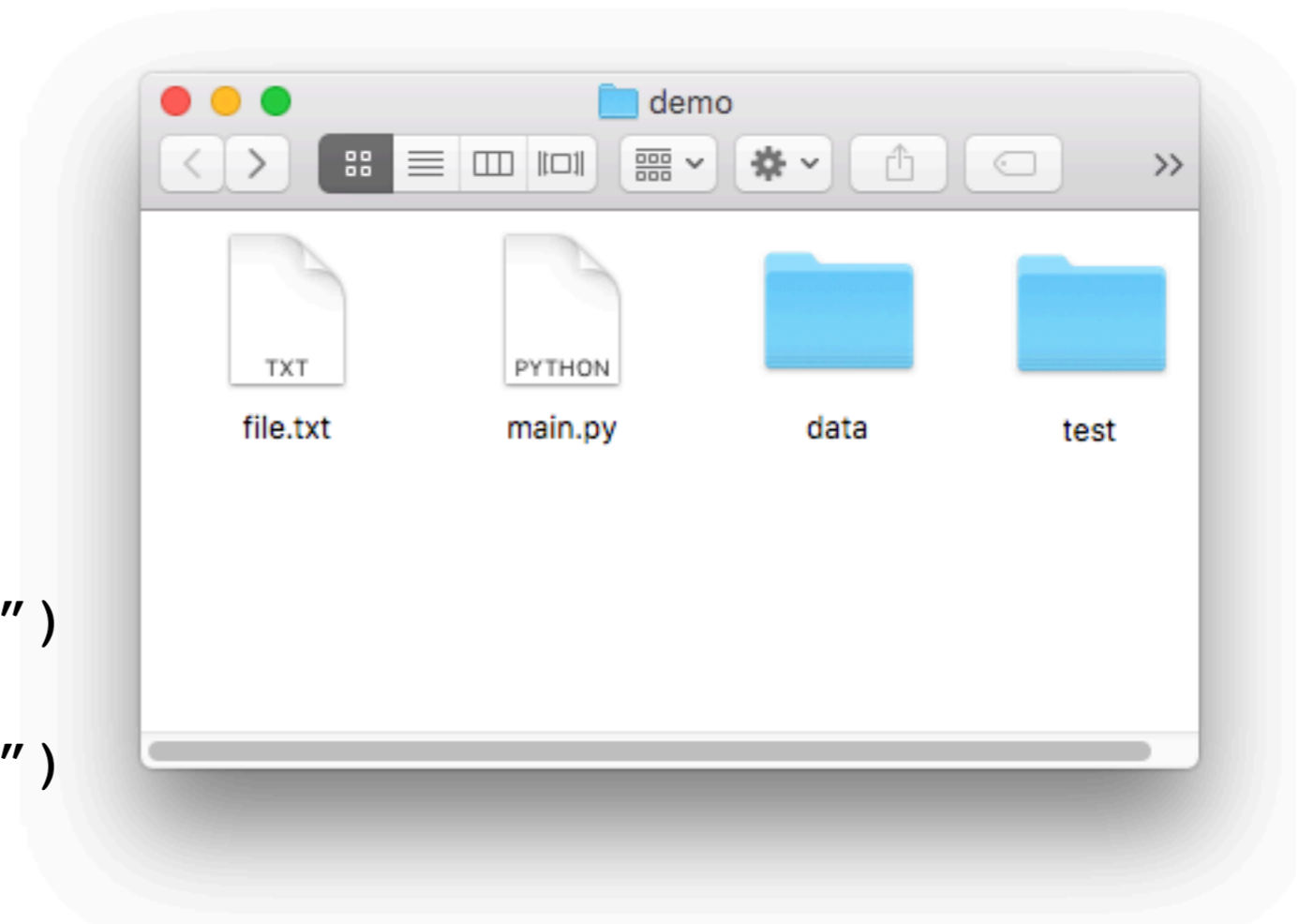


OS Module (Operating System)

Many functions in `os` and `os.path` for working w/ files

- `os.listdir`
- `os.mkdir`
- `os.path.exists`
- **`os.path.isfile`**
- `os.path.isdir`
- `os.path.join`

```
>>> import os
>>> os.path.isfile("haha.txt")
False
>>> os.path.isfile("file.txt")
True
```

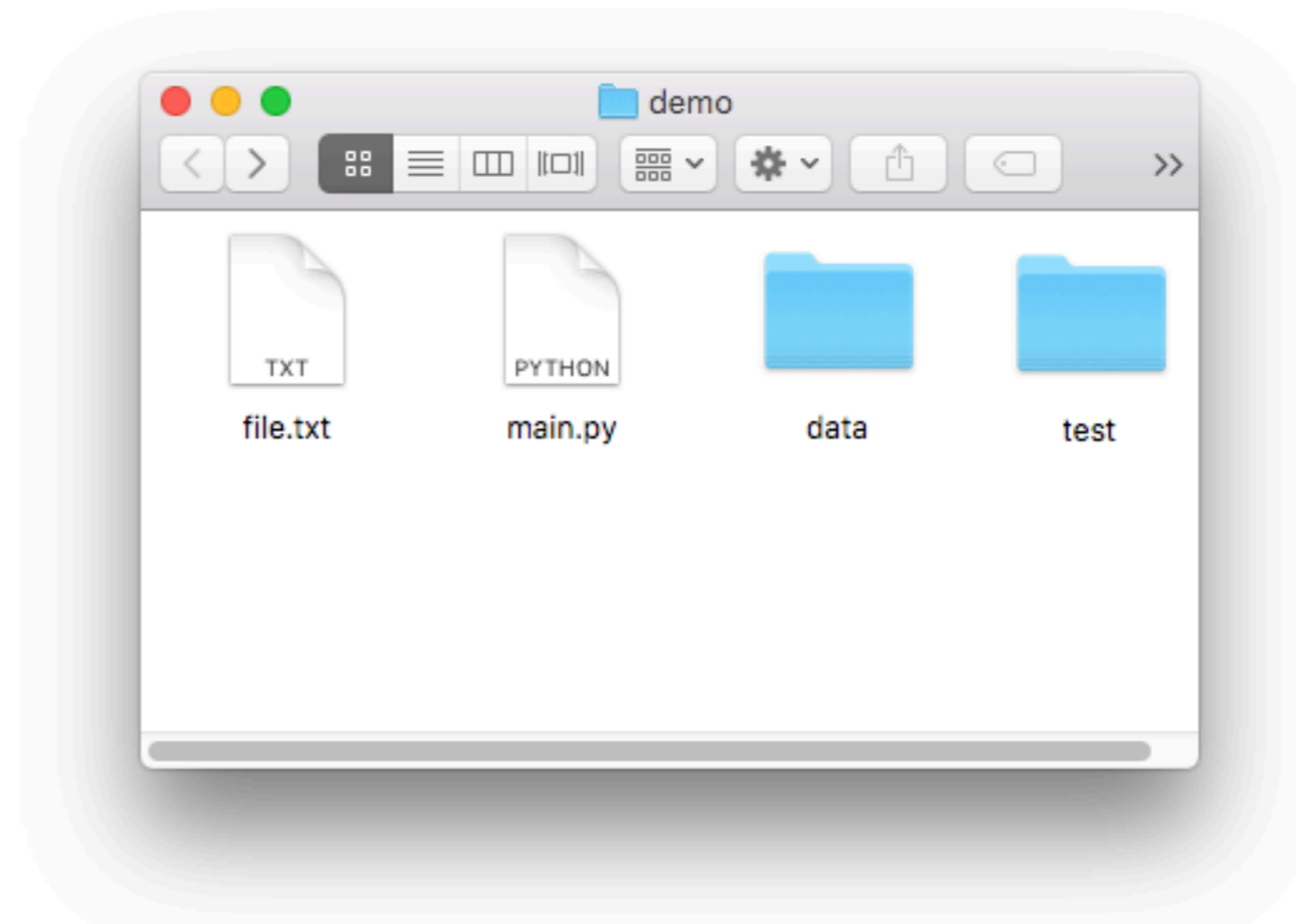


OS Module (Operating System)

Many functions in `os` and `os.path` for working w/ files

- `os.listdir`
- `os.mkdir`
- `os.path.exists`
- `os.path.isfile`
- **`os.path.isdir`**
- `os.path.join`

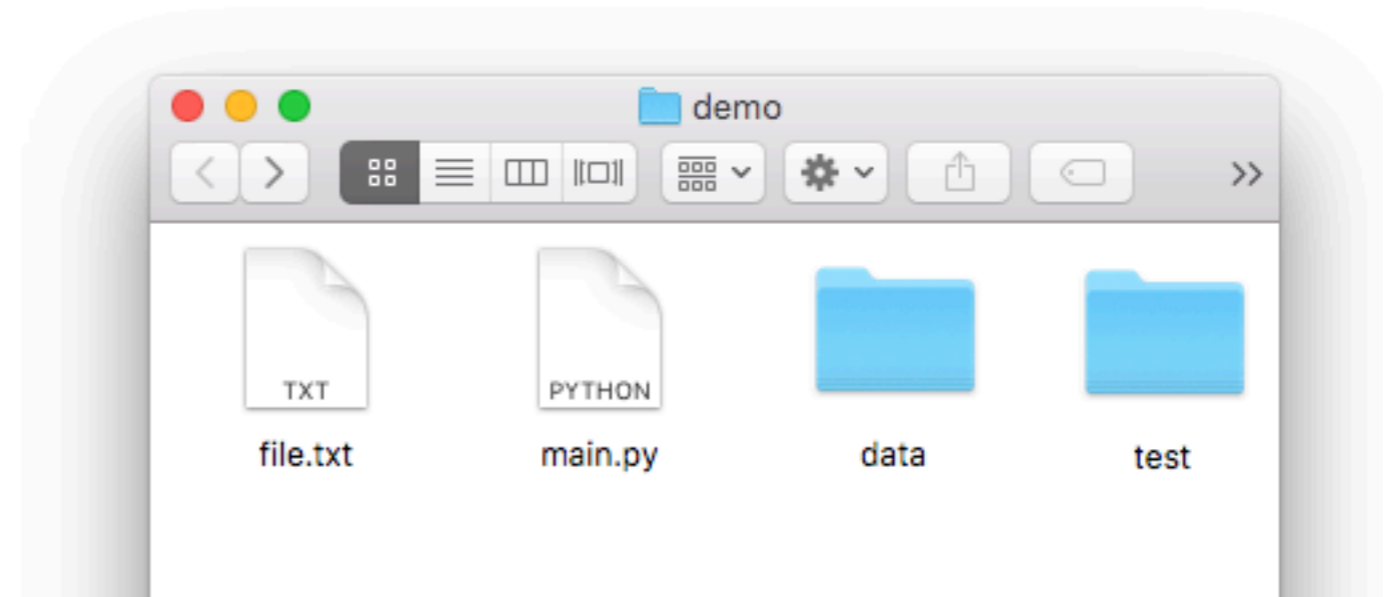
```
>>> import os
>>> os.path.isfile("data")
False
>>> os.path.isdir("data")
True
```



OS Module (Operating System)

Many functions in `os` and `os.path` for working w/ files

- `os.listdir`
- `os.mkdir`
- `os.path.exists`
- `os.path.isfile`
- `os.path.isdir`
- **`os.path.join`**



```
>>> import os
>>> os.path.join("data", "movies.csv")
data/movies.csv
```

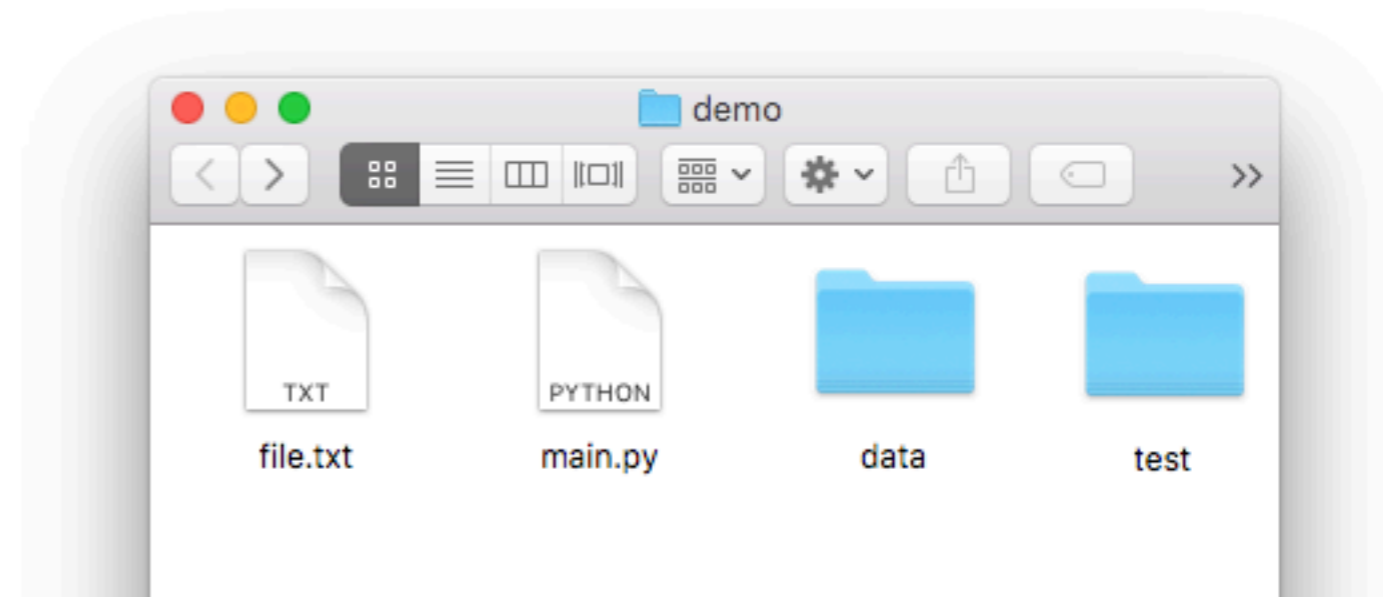


on Mac/Linux

OS Module (Operating System)

Many functions in `os` and `os.path` for working w/ files

- `os.listdir`
- `os.mkdir`
- `os.path.exists`
- `os.path.isfile`
- `os.path.isdir`
- **`os.path.join`**



```
>>> import os
>>> os.path.join("data", "movies.csv")
data\\movies.csv
```

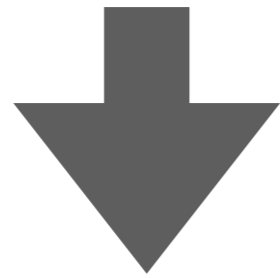


on Windows

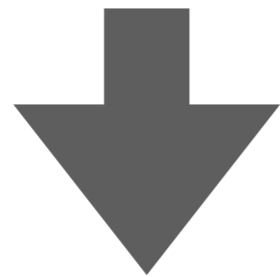
Windows

Your project:

```
path = "\".join("data", "movies.csv")  
f = open(path)  
...
```



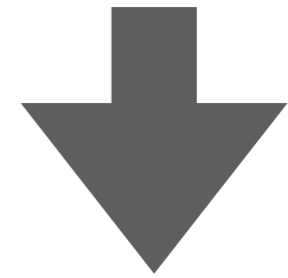
you run test.py



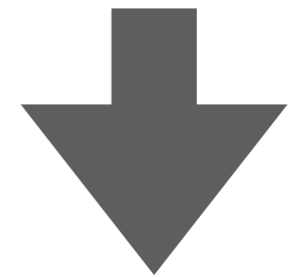
Linux

submit

...



we run test.py



Learning Objectives Today

Basic file interactions

- opening/closing
- reading/writing

OS module

- listdir, mkdir, exists, isdir, isfile, join

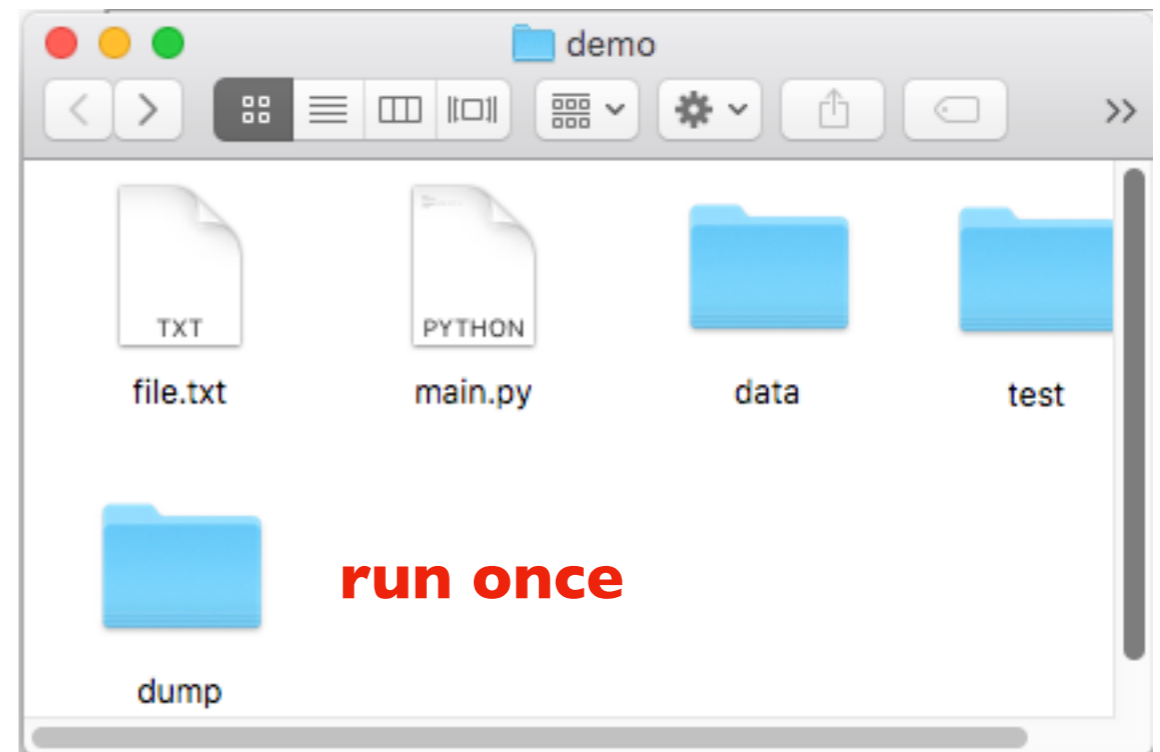
File exceptions

Encodings

Exceptions

Working with files leads to many exceptions

- missing files
- lacking permissions
- not enough space
- mixing up directories and files
- corrupt formats
- etc, etc



```
import os
```

```
os.mkdir('dump')
```

```
f = open(os.path.join('dump', 'out.txt'), 'w')
```

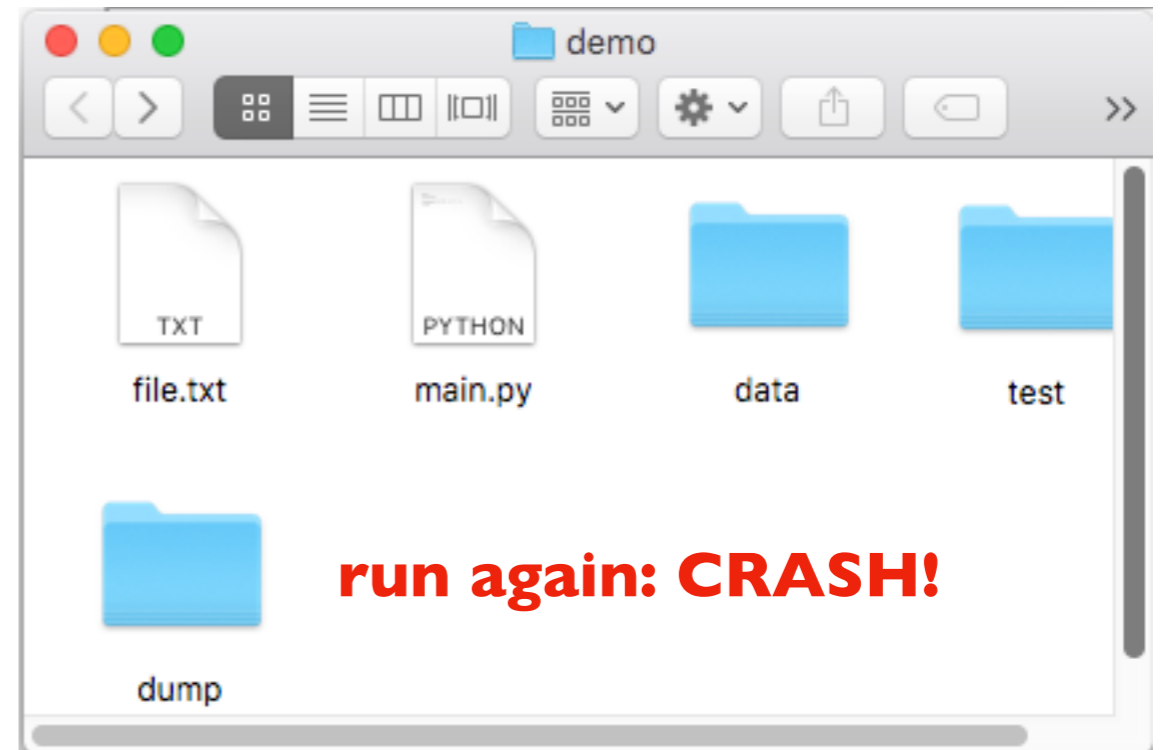
```
f.write('hi')
```

```
f.close()
```

Exceptions

Working with files leads to many exceptions

- missing files
- lacking permissions
- not enough space
- mixing up directories and files
- corrupt formats
- etc, etc



```
import os
```

```
os.mkdir('dump')
```

```
f = open(os.path.join('dump', 'out.txt'), 'w')
```

```
f.write('hi')
```

```
f.close()
```

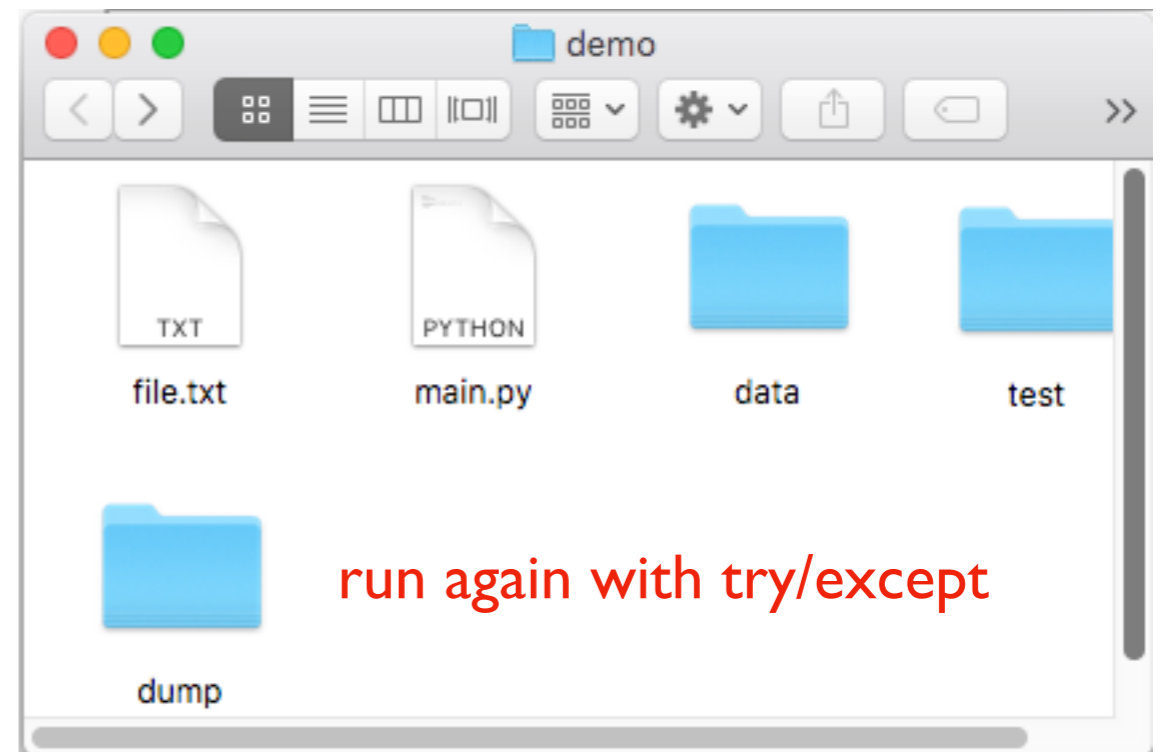
```
Traceback (most recent call last):  
  File "test2.py", line 3, in <module>  
    os.mkdir('dump')
```

```
FileExistsError: [Errno 17] File exists: 'dump'
```

Exceptions

Working with files leads to many exceptions

- missing files
- lacking permissions
- not enough space
- mixing up directories and files
- corrupt formats
- etc, etc



```
import os

try:
    os.mkdir('dump')
except FileExistsError:
    pass # ignore it if dump already existed

f = open(os.path.join('dump', 'out.txt'), 'w')
f.write('hi')
f.close()
```

Learning Objectives Today

Basic file interactions

- opening/closing
- reading/writing

OS module

- listdir, mkdir, exists, isdir, isfile, join

File exceptions

Encodings

Exercise: person 1 encodes a word with encoding 1,
person 2 decodes with encoding 2

Word: ukulele

A	00001	N	01110
B	00010	O	01111
C	00011	P	10000
D	00100	Q	10001
E	00000	R	10010
F	00110	S	10011
G	00111	T	10100
H	01000	U	10101
I	01001	V	10110
J	01010	W	10111
K	01011	X	11000
L	11111	Y	11001
M	01101	Z	11010

encoding 1

A	00001	N	01110
B	00010	O	01011
C	00011	P	10000
D	00100	Q	10001
E	00000	R	10010
F	00110	S	10011
G	00111	T	10100
H	01000	U	01100
I	01001	V	10110
J	01010	W	10111
K	01111	X	11000
L	10101	Y	11001
M	01101	Z	11010

encoding 2

Exercise: person 1 encodes a word with encoding 1,
person 2 decodes with encoding 2

Word: ~~ukulele~~
lol?e?e

A	00001	N	01110
B	00010	O	01111
C	00011	P	10000
D	00100	Q	10001
E	00000	R	10010
F	00110	S	10011
G	00111	T	10100
H	01000	U	10101
I	01001	V	10110
J	01010	W	10111
K	01011	X	11000
L	11111	Y	11001
M	01101	Z	11010

encoding 1

A	00001	N	01110
B	00010	O	01011
C	00011	P	10000
D	00100	Q	10001
E	00000	R	10010
F	00110	S	10011
G	00111	T	10100
H	01000	U	01100
I	01001	V	10110
J	01010	W	10111
K	01111	X	11000
L	10101	Y	11001
M	01101	Z	11010

encoding 2

Encoding Defaults Done Wrong

Mac

```
f = open('example.txt', 'w',  
        encoding='utf-8')  
f.write('baño')  
f.close()
```

example.txt

Windows

```
f = open('example.txt', 'r',  
        encoding='cp1252')  
print(f.read())  
f.close()
```

example.txt

Windows computer prints **“baÑ±o”** instead of **“baño”**

Encoding Defaults Done Wrong

Mac

```
f = open('example.txt', 'w',  
         encoding='utf-8')  
f.write('baño')  
f.close()
```

example.txt

Windows

```
f = open('example.txt', 'r',  
         encoding='cp1252')  
print(f.read())  
f.close()
```

example.txt

Takeaway: if you see weird characters printed by your program, it's a good time to learn more about encodings

Coding Demos

Demo 1: add numbers in a file

Goal: read all lines from a file as integers and add them

Input:

- file containing **50 million numbers** between 0 and 100

Output:

- The sum of the numbers

Example:

```
prompt> python sum.py  
2499463617
```

Two ways:

- Put all lines in a list first
- Directly use iterable file

Bonus: create generator function that does the str => int conversion

Challenge - Demo 1: Score Tracker

Goal: tally up points, and print who is winning

Input:

- Person who just scored

Output:

- Everybody's score

Example:

```
prompt> python point.py alice  
alice: 1
```

```
prompt> python point.py bob  
alice: 1  
bob: 1
```

```
prompt> python point.py alice  
alice: 2  
bob: 1
```

Challenge - Demo 2: File Finder

Goal: search directories (recursively) for a given file name, then print that file

Input:

- The filename to search for

Output:

- The contents of that file

Challenge - Demo 3: sorting files by line length

Goal: output file contents, with shortest line first

Input:

- a text file

Output:

- print lines sorted