[301] Files

Tyler Caraza-Harter

Relation to Recent Topics...

1   file objects are a type of iterator (lecture 24)

2   exceptions occur a lot with files (lecture 25)
not found, out of space, no permissions, etc

Learning Objectives Today

Basic file interactions
- opening/closing
- reading/writing

File formats
- JSON
- CSV

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

File exceptions

Encodings

Learning Objectives Today

Basic file interactions
- opening/closing
- reading/writing

File formats
- JSON
- CSV

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

File exceptions

Encodings

File objects

```
f = open(path)



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



f.close()
```

File objects

```
f = open(path)


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




f.close()
```

file object

File objects

built-in open function

```
f = open(path)


file object

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




f.close()
```

File objects

built-in open function

```
f = open(path)




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




f.close()
```

file object

file path

File objects

main.py:

```
f = open(path)

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



f.close()
```

built-in open function

file object

file path

we're running the code in this directory

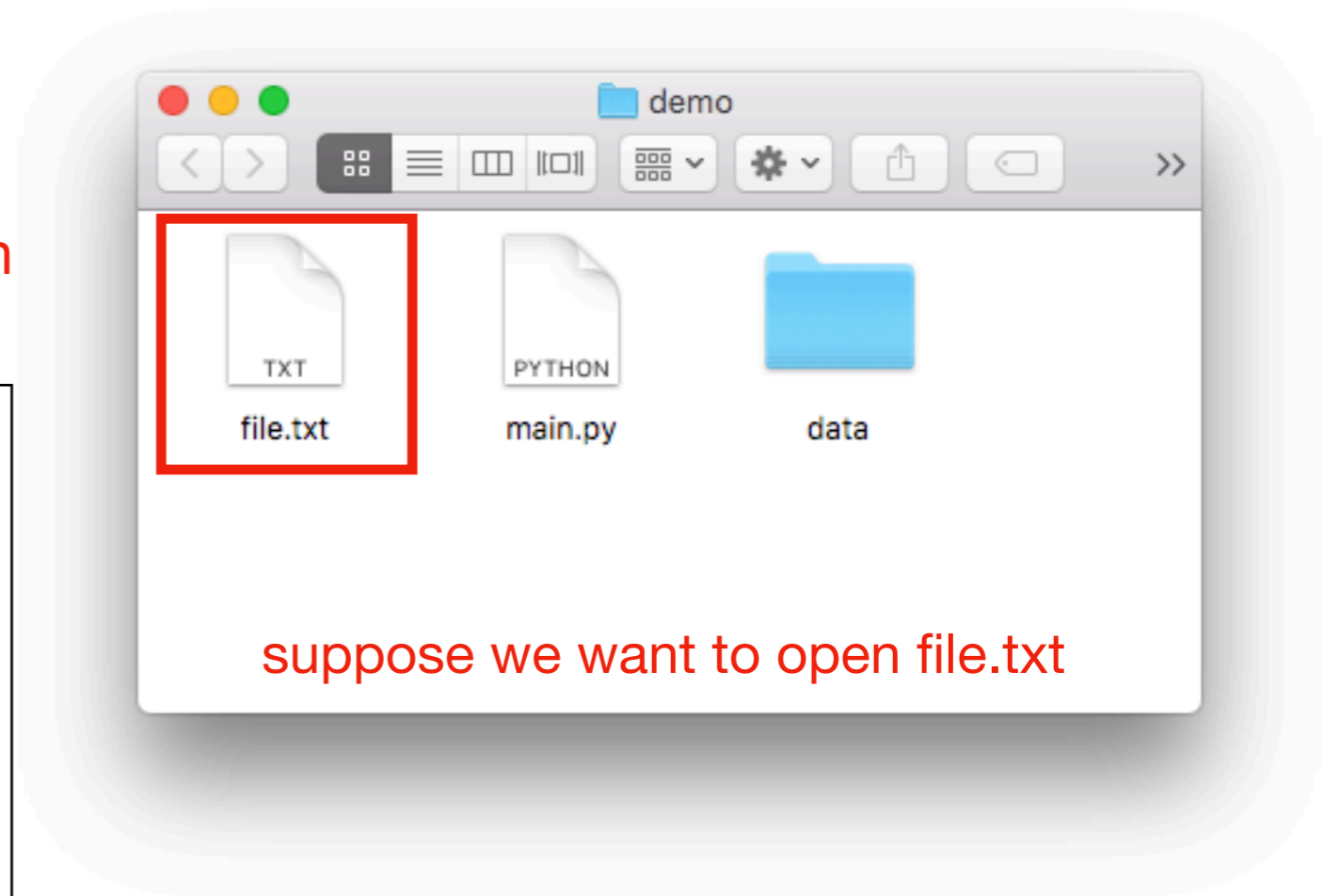(Finder window showing "demo" folder containing: file.txt, main.py, data)
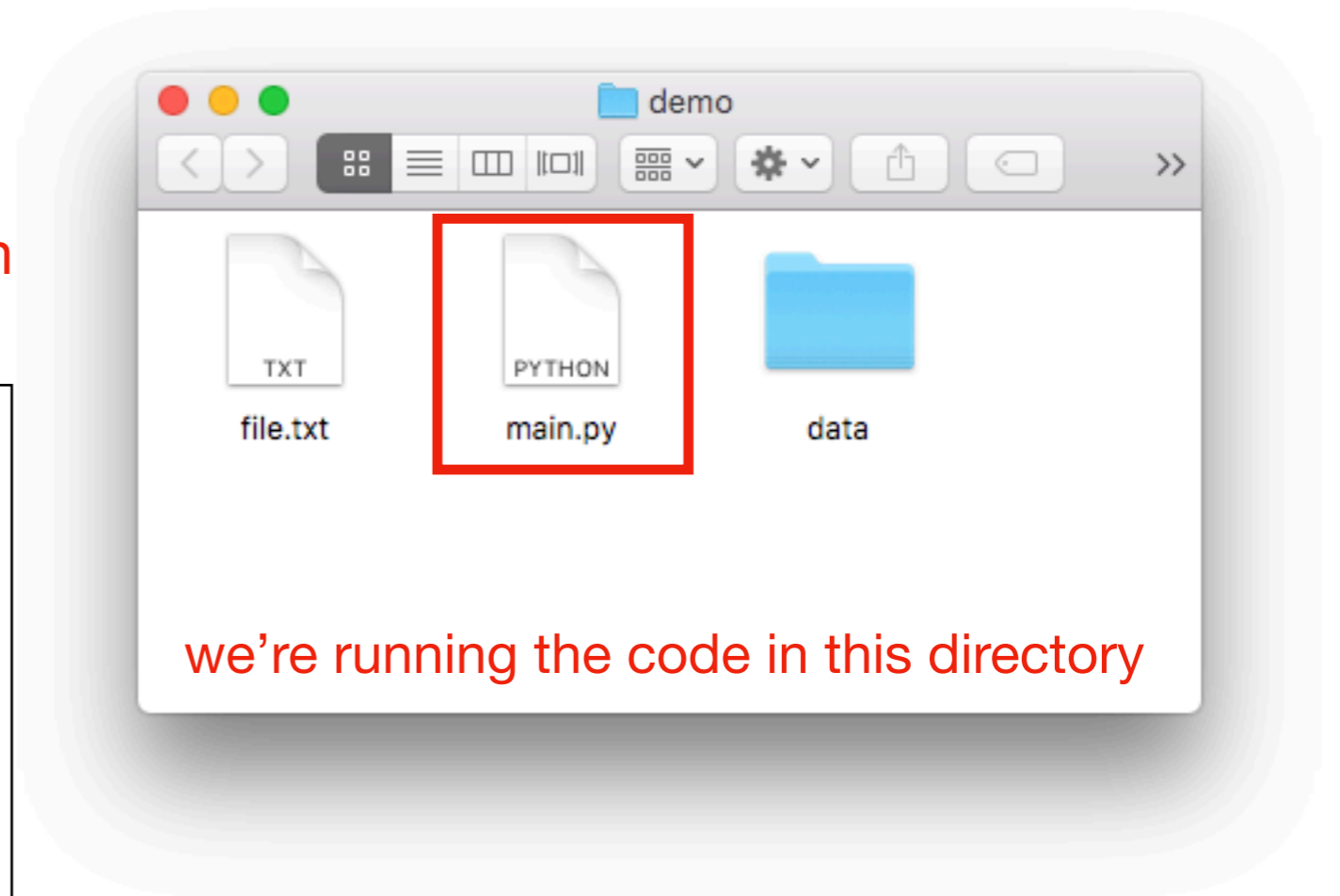
File objects

main.py:

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



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



f.close()
```
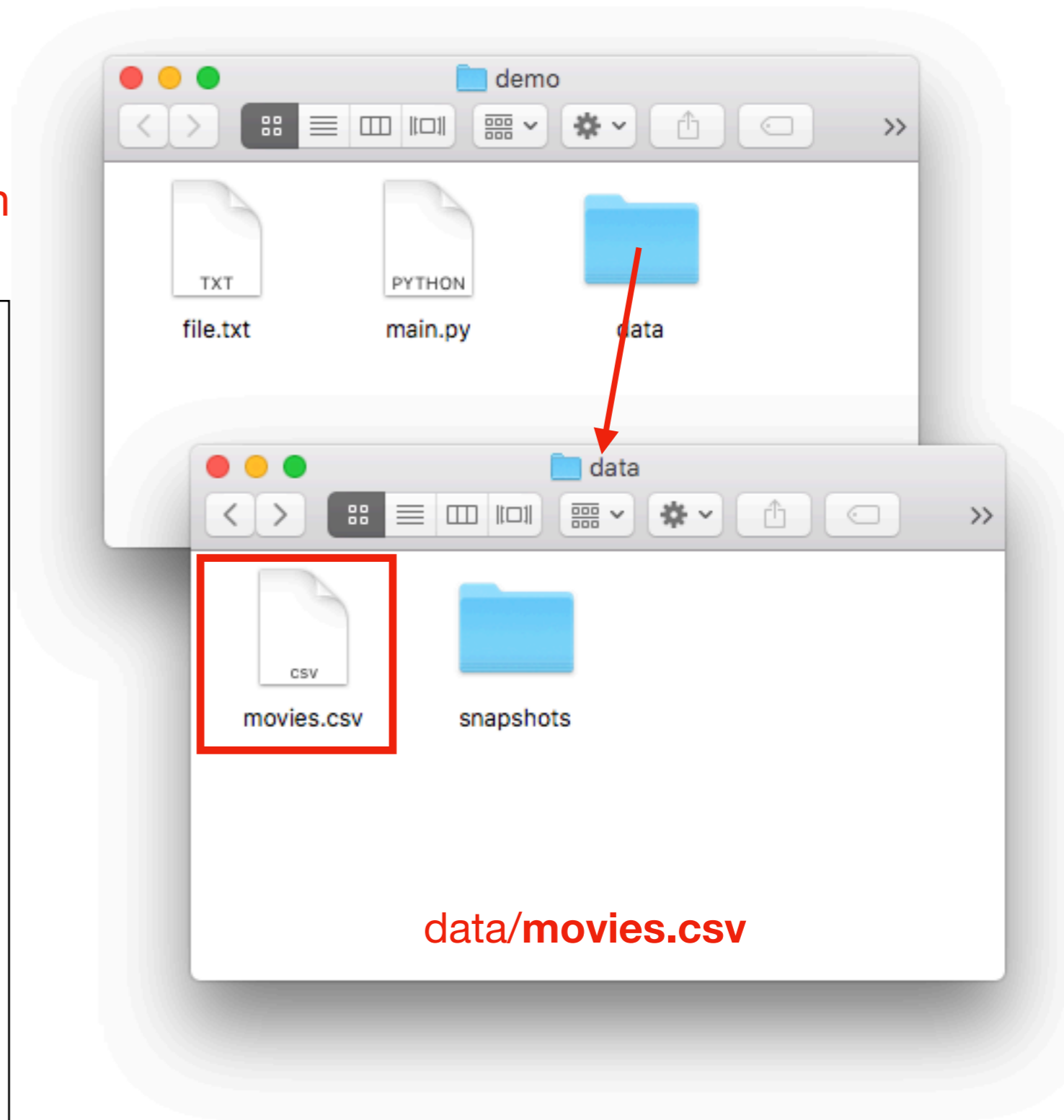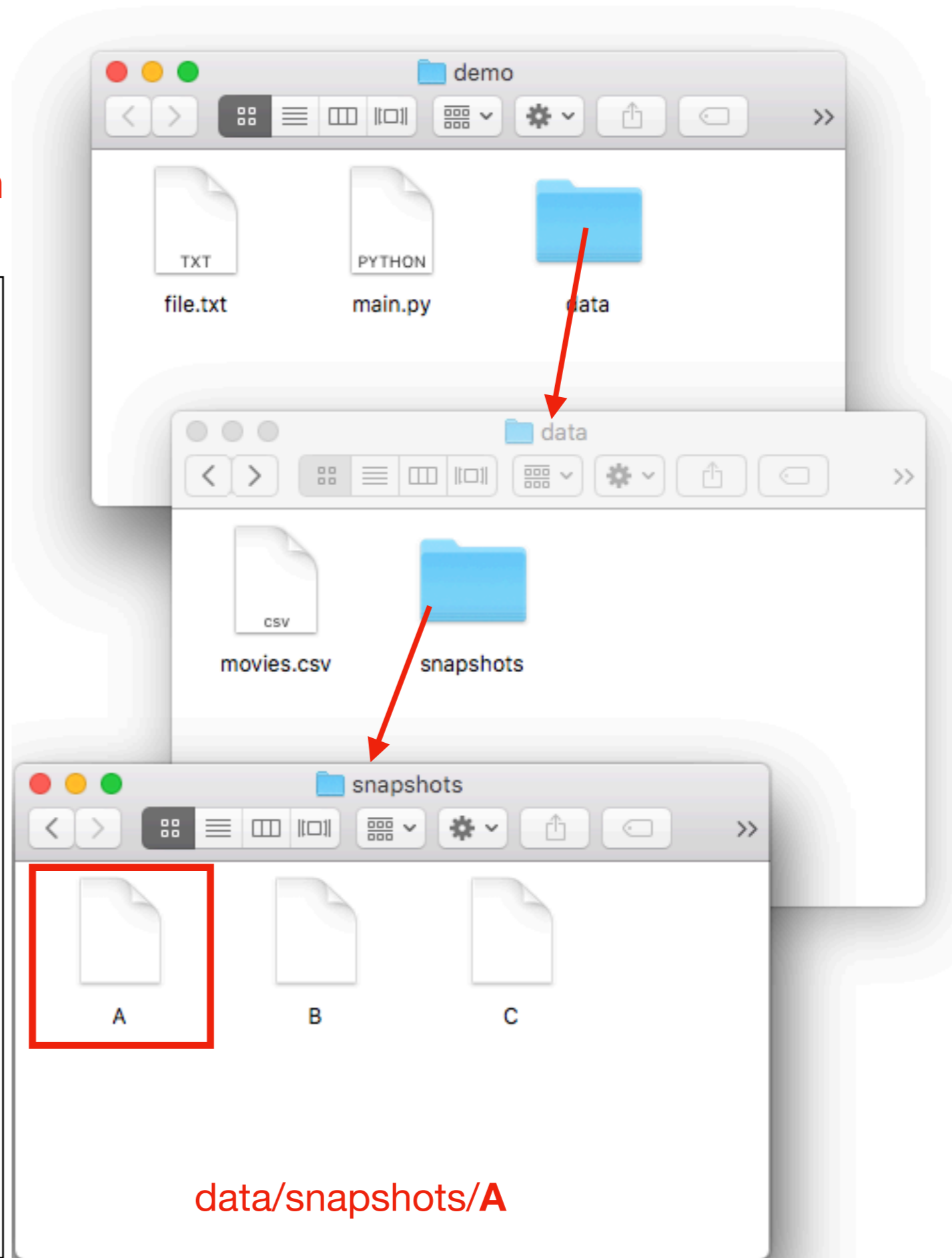
built-in open function

file object

file path

suppose we want to open file.txt

File objects

built-in open function

main.py:

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



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



f.close()
```

file object

file path

data/**movies.csv**

File objects

built-in open function

main.py:
```
f = open(
    "data/snapshots/A")



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



f.close()
```

file object

file path

data/snapshots/**A**

File objects

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



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



f.close()
```

File objects

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



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




f.close()
```

using file

File objects

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



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




f.close()
```

using file

cleanup

File objects

imagine a file object as a sandwich...

```
f = open(…)

use file

f.close()
```

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



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



f.close()
```

using file

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
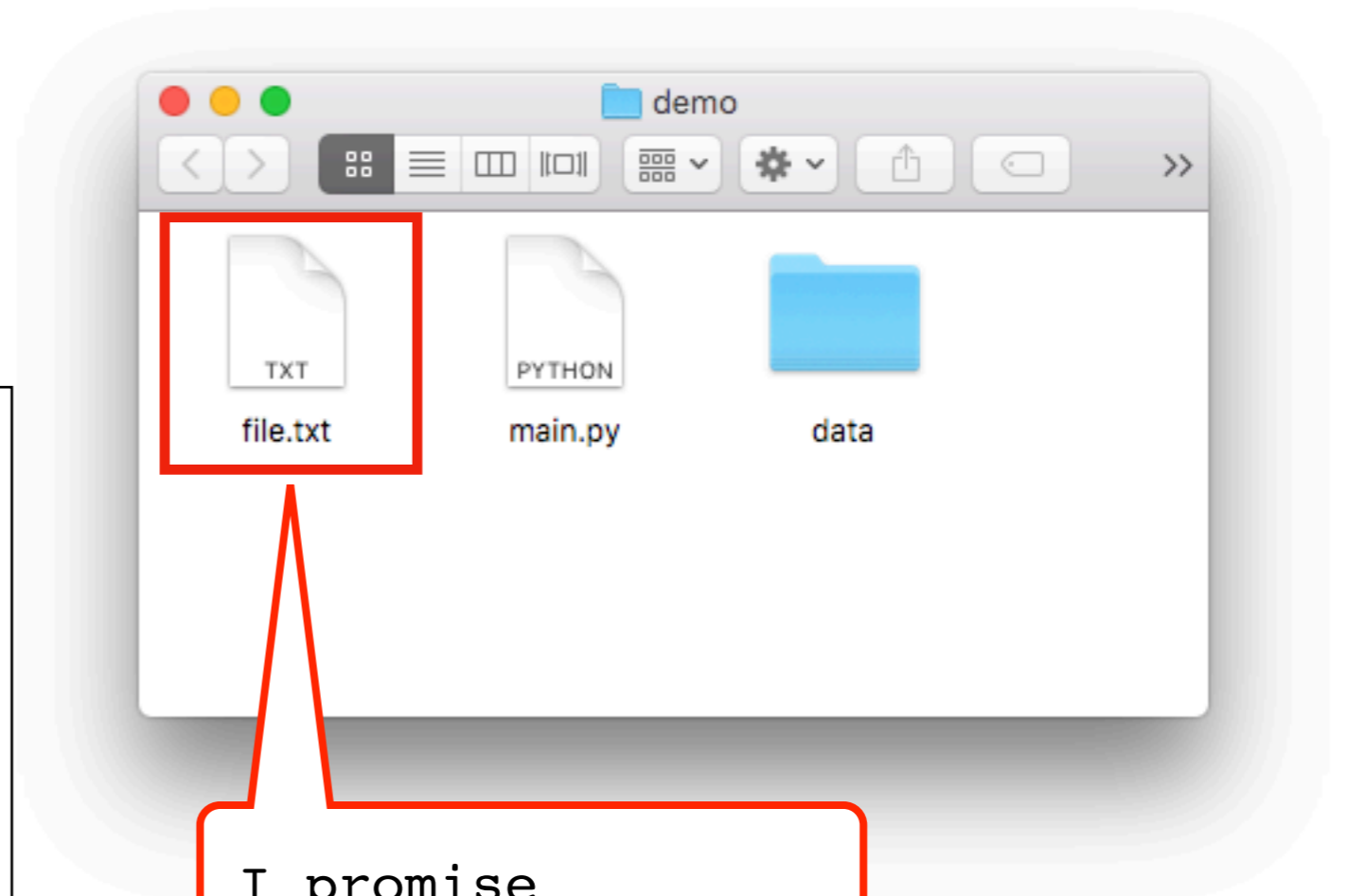
Encodings

Reading a file

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



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



f.close()
```
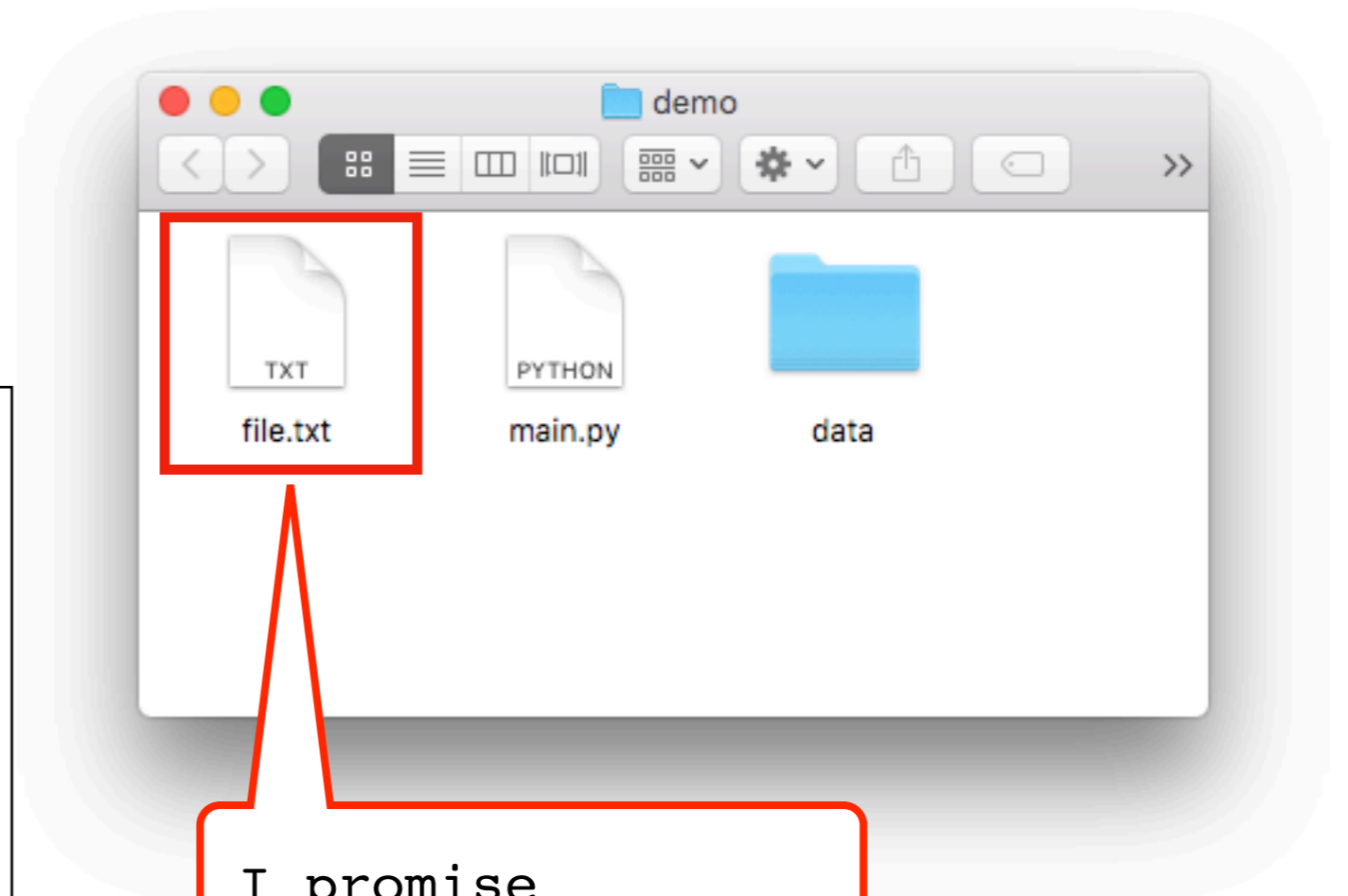
I promise
to always
close my files

Reading a file


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



data = f.read()   Option 1


print(data)



f.close()
```

I promise
to always
close my files

read() method
- fetch entire file contents
- return as a string
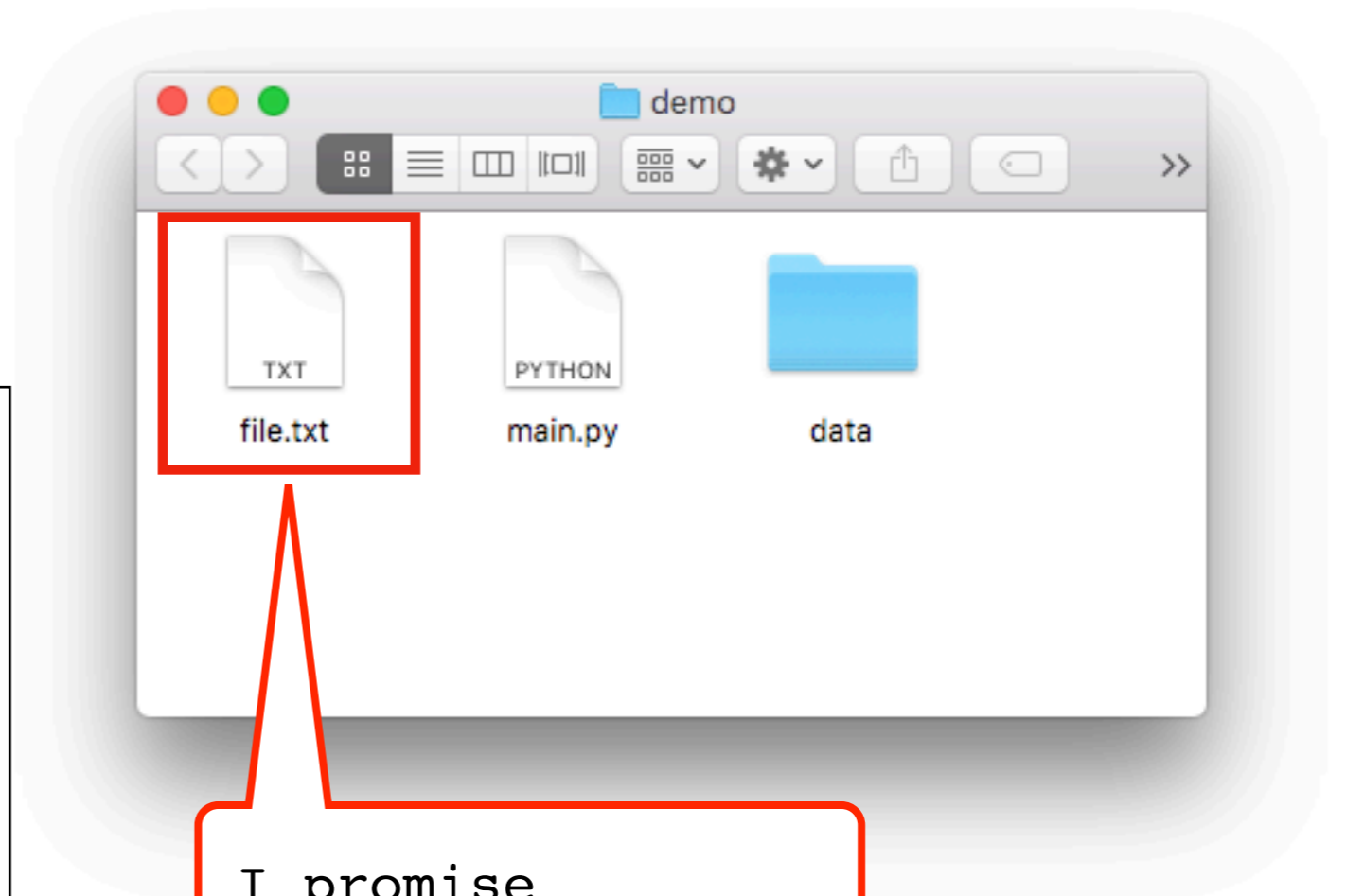
Reading a file

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


data = f.read()    [Option 1]


print(data)



f.close()
```

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

I promise
to always
close my files

read() method
- fetch entire file contents
- return as a string

demo

TXT
file.txt

PYTHON
main.py

data

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
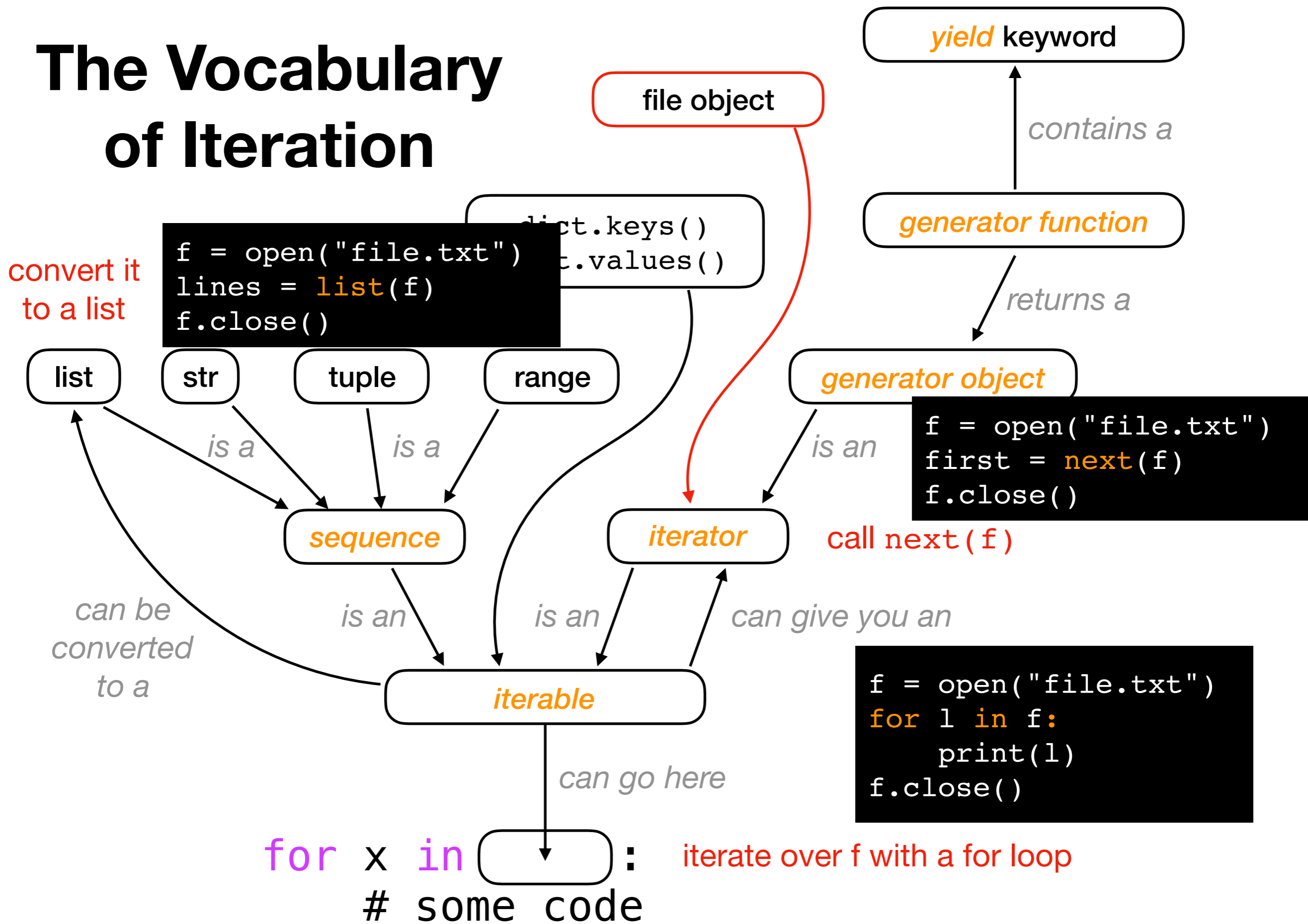
yield keyword

file object

dict.keys()
dict.values()

generator function

contains a

returns a

list   str   tuple   range   *generator object*

is a   *is a*

is an

sequence   *iterator*

can be converted to a

is an   *is an*   *can give you an*

iterable

can go here

```
for x in (    ):
    # some code
```

The Vocabulary of Iteration

yield keyword

file object

contains a

dict.keys()
dict.values()

generator function

convert it
to a list

returns a

list   str   tuple   range

generator object

is a   *is a*

is an

call next(f)

sequence

iterator

can be
converted
to a

is an   *is an*   *can give you an*

iterable

can go here

for x in ⬚:
    # some code

iterate over f with a for loop

The Vocabulary of Iteration

yield keyword

file object

contains a

generator function

```
f = open("file.txt")
lines = list(f)
f.close()
```

convert it
to a list

```
dict.keys()
  .values()
```

returns a

generator object

list   str   tuple   range

```
f = open("file.txt")
first = next(f)
f.close()
```

is a   *is a*

```
f = open("file.txt")
first = next(f)
f.close()
```

sequence   *iterator*   call `next(f)`

is an   *is an*   *can give you an*

*can be
converted
to a*

iterable

```
f = open("file.txt")
for l in f:
    print(l)
f.close()
```

can go here

`for x in`   `:`   iterate over f with a for loop
`  # some code`

Write a file

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



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



f.close()
```

I promise
to always
close my files

Write a file

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



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



f.close()
```


I promise
to always
close my files

Write a file

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



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



f.close()
```
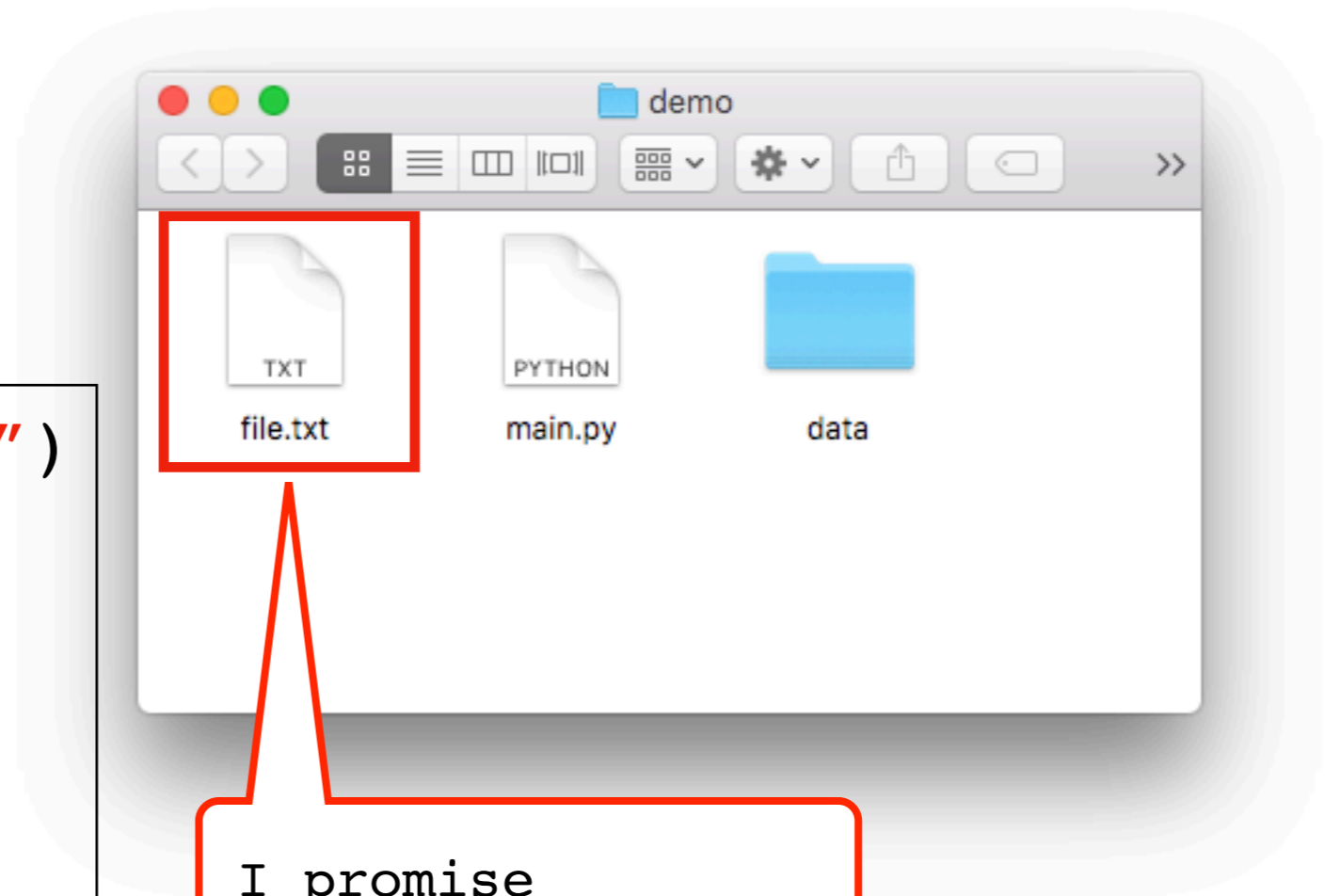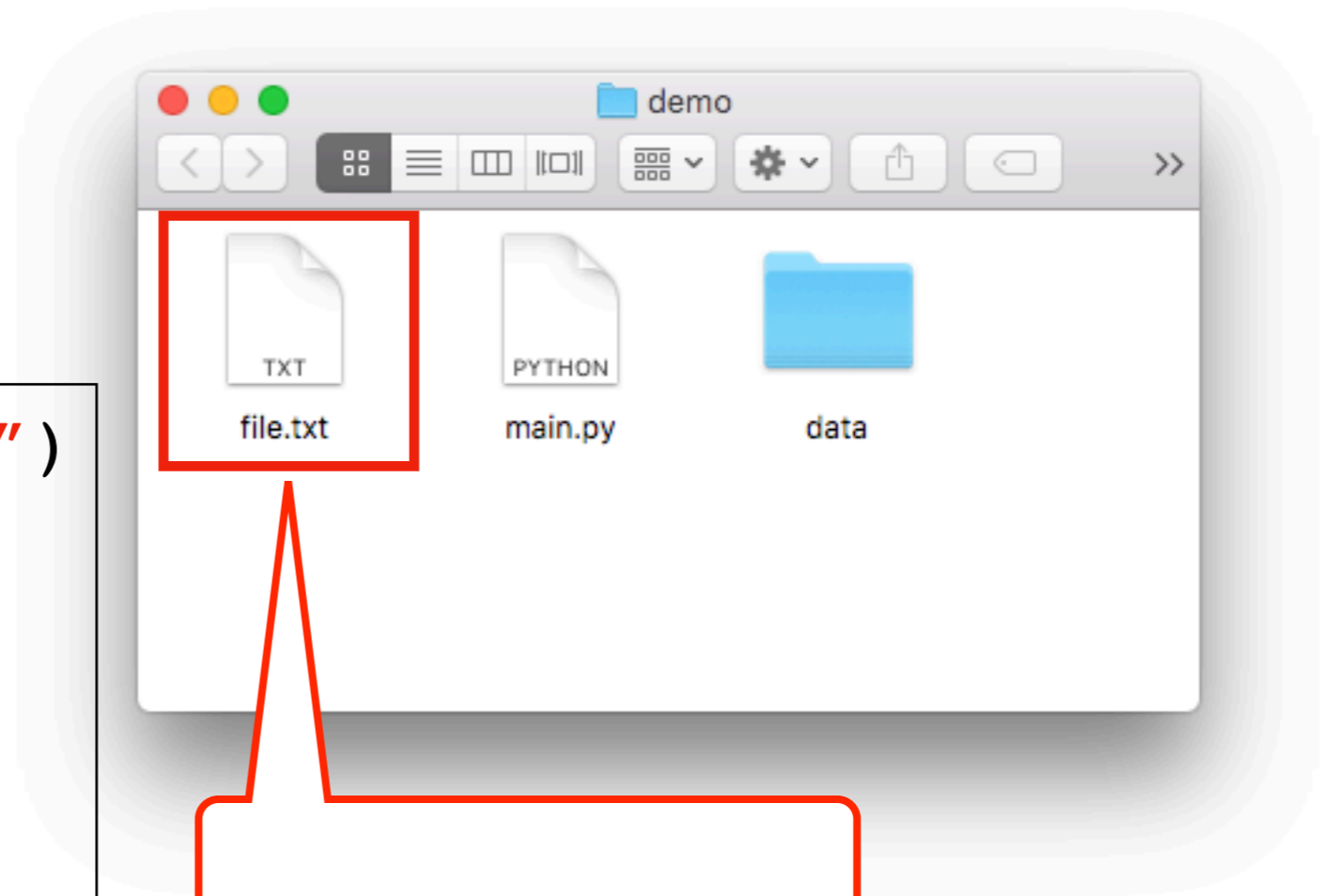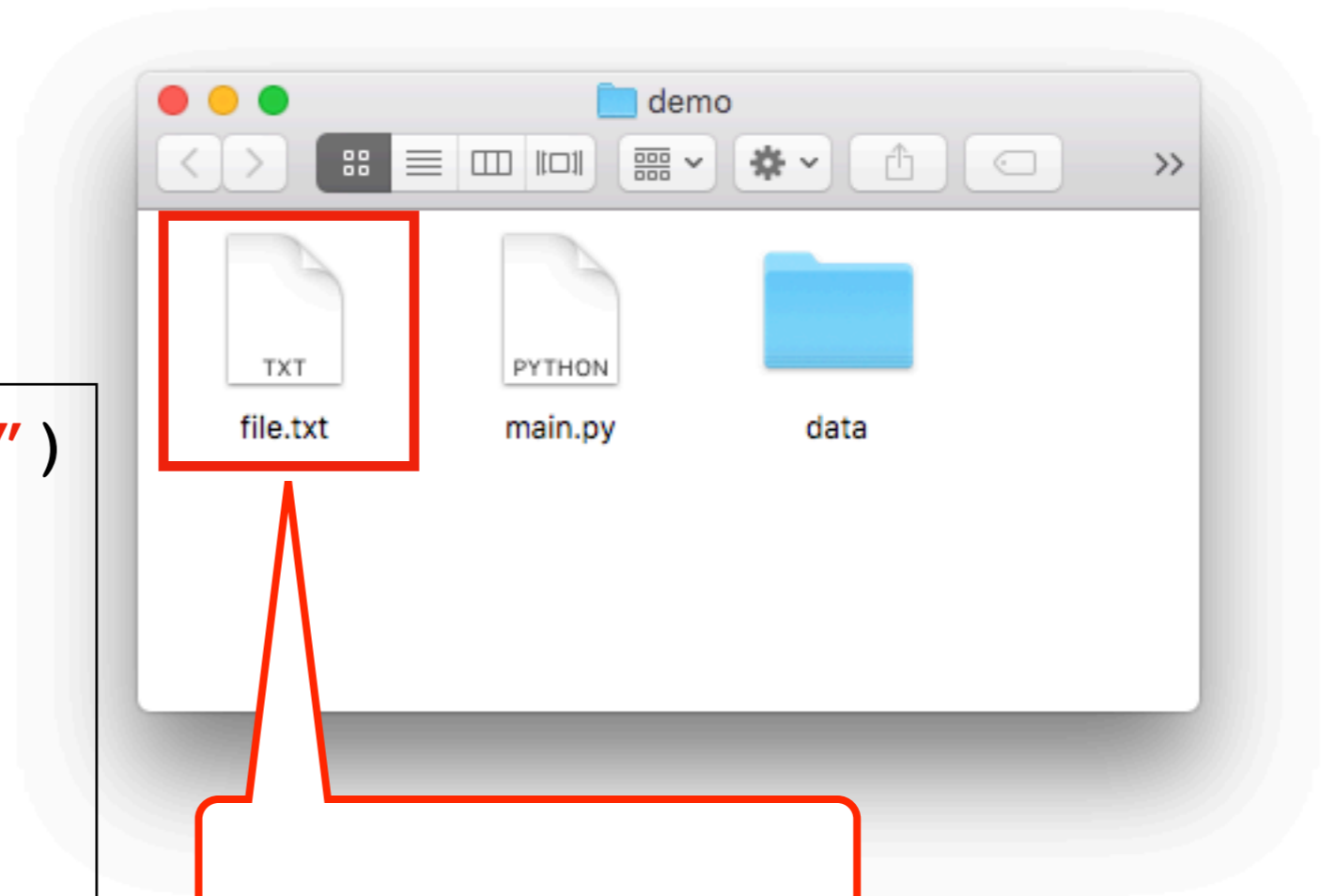
I promise
to always
close my files

Write a 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

```
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.

Write a 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

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



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



f.close()
```
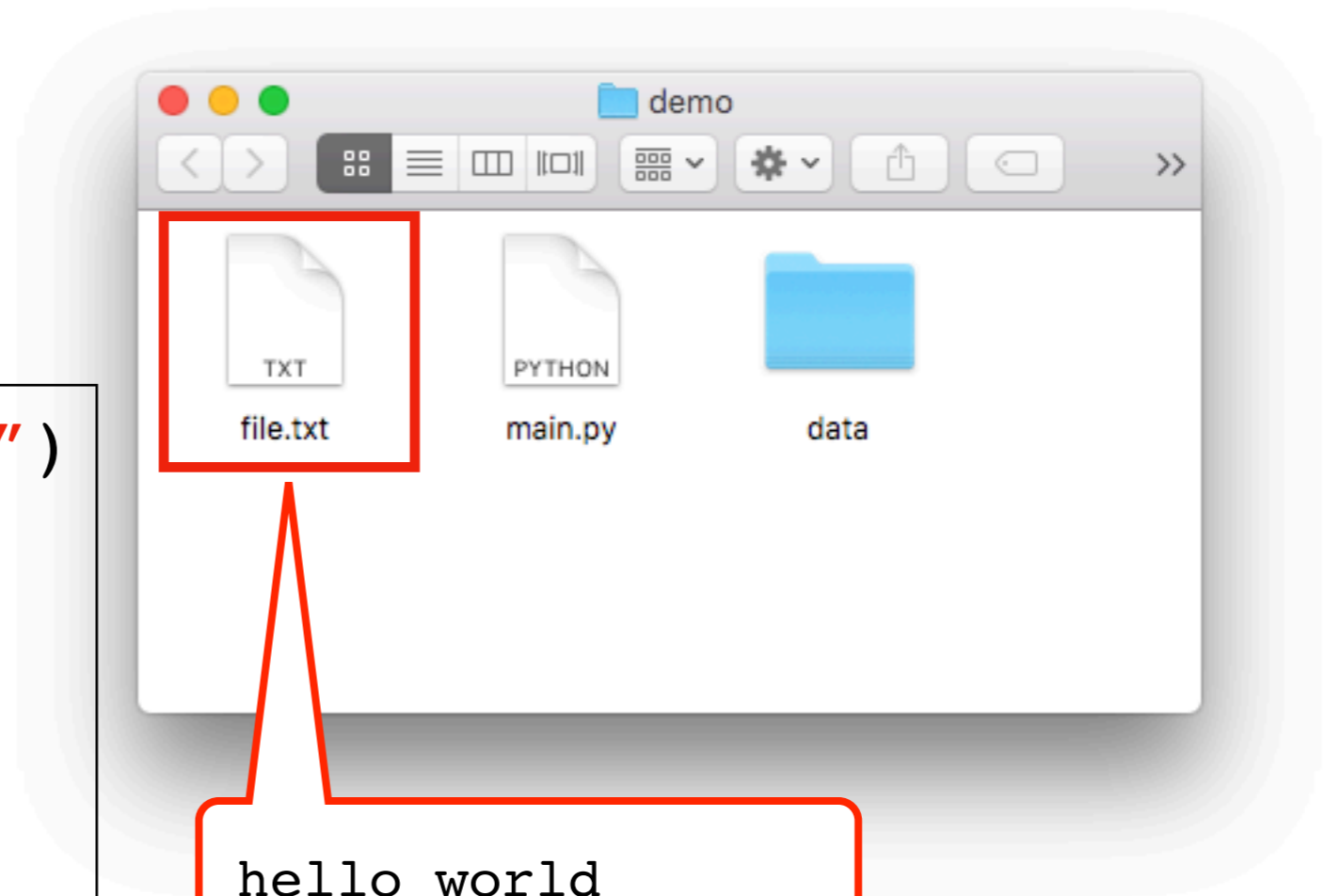
demo

TXT
file.txt

PYTHON
main.py

data

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()
```

demo

TXT
file.txt

PYTHON
main.py

data

hello world

Write a file

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



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



f.close()
```

demo

TXT
file.txt

PYTHON
main.py

data

```
hello world
!!!!!!
```

Write a file

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



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



f.close()
```

demo

file.txt   main.py   data
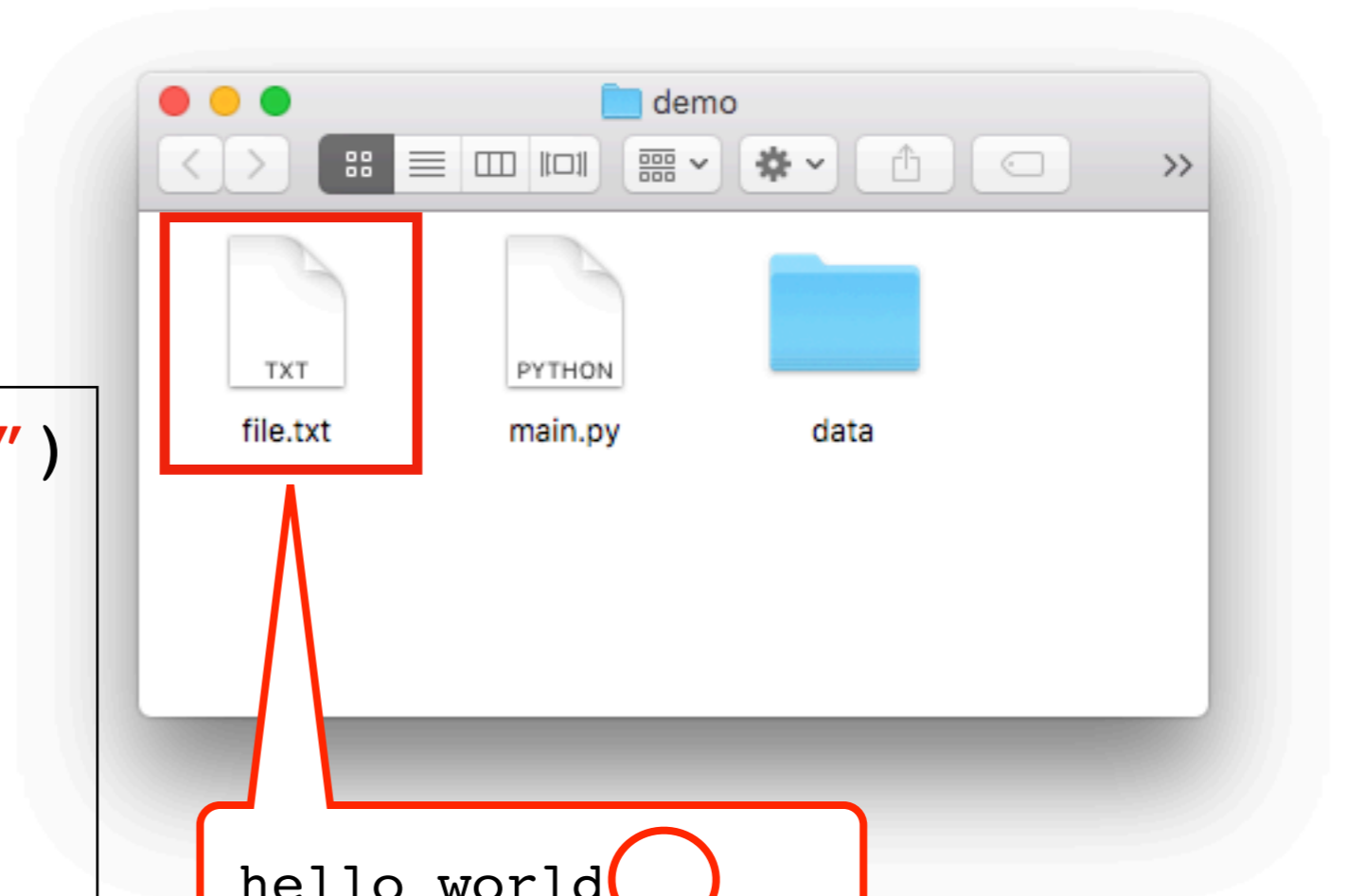
hello world
!!!!!!

be careful with newlines

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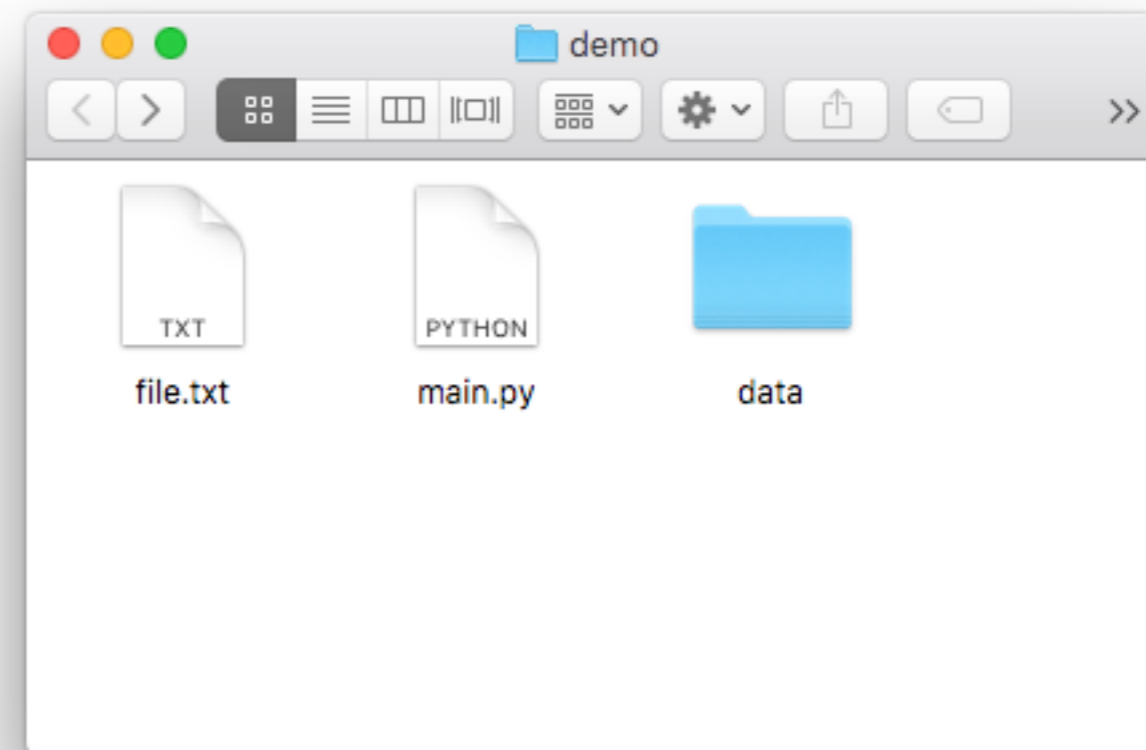
Many functions in os and os.path for working w/ files

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

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"]
```
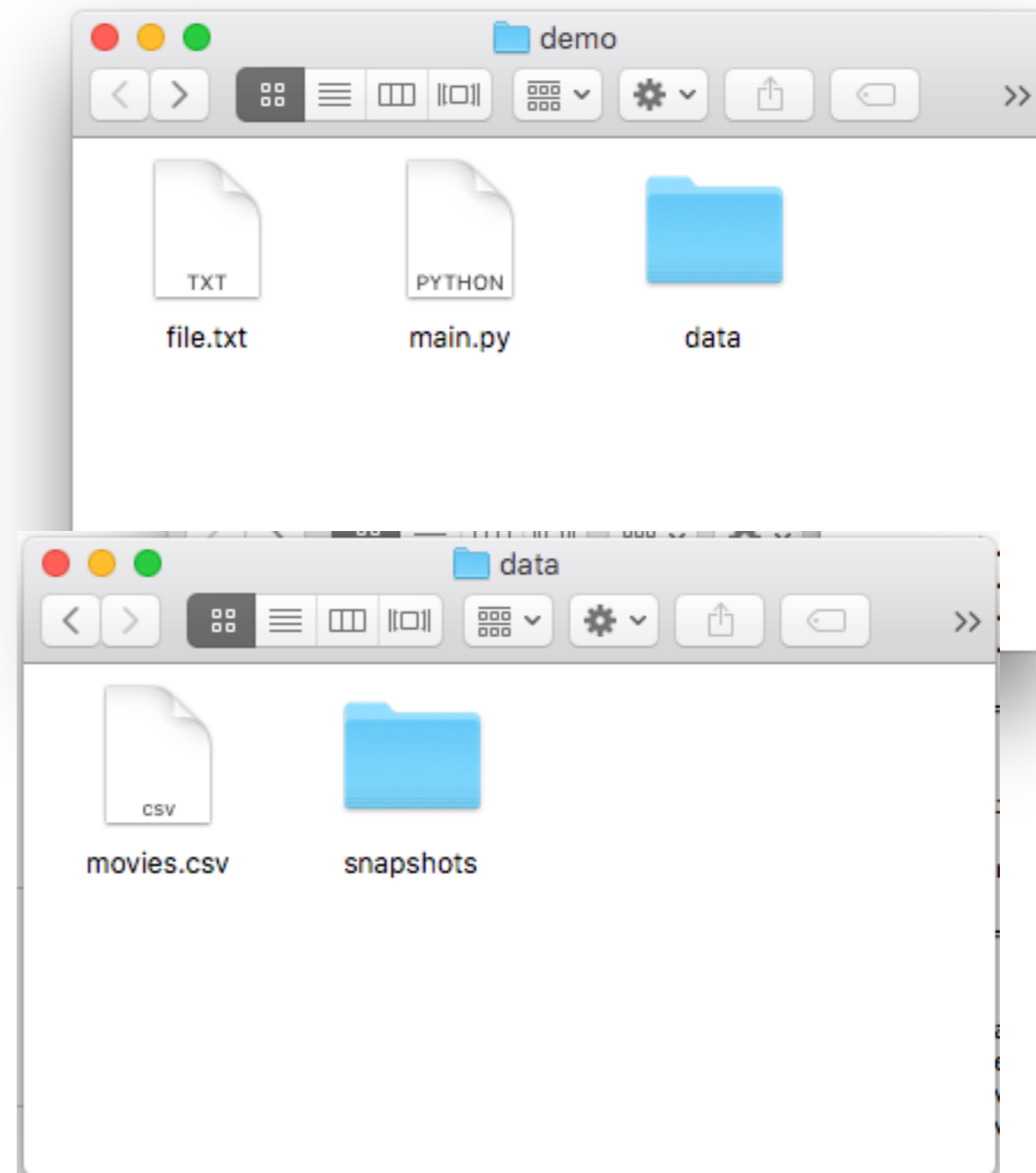
OS Module (Operating System)

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("data")
["movies.csv", "snapshots"]
```
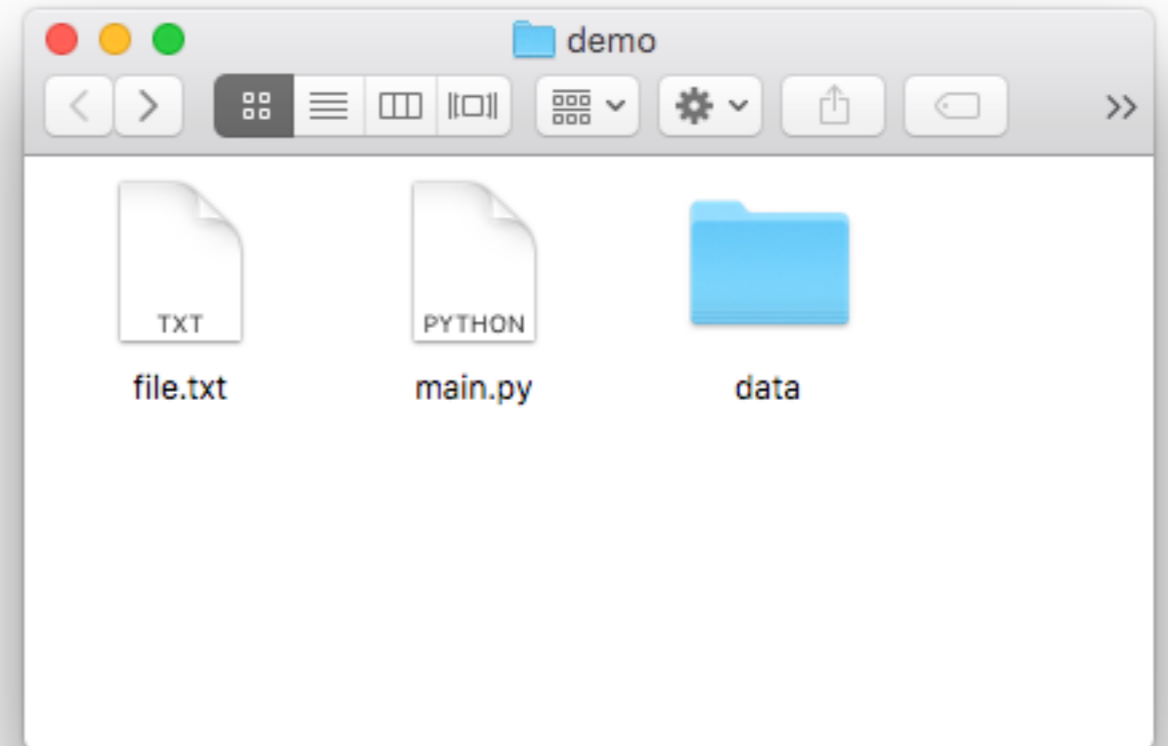
OS Module (Operating System)

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

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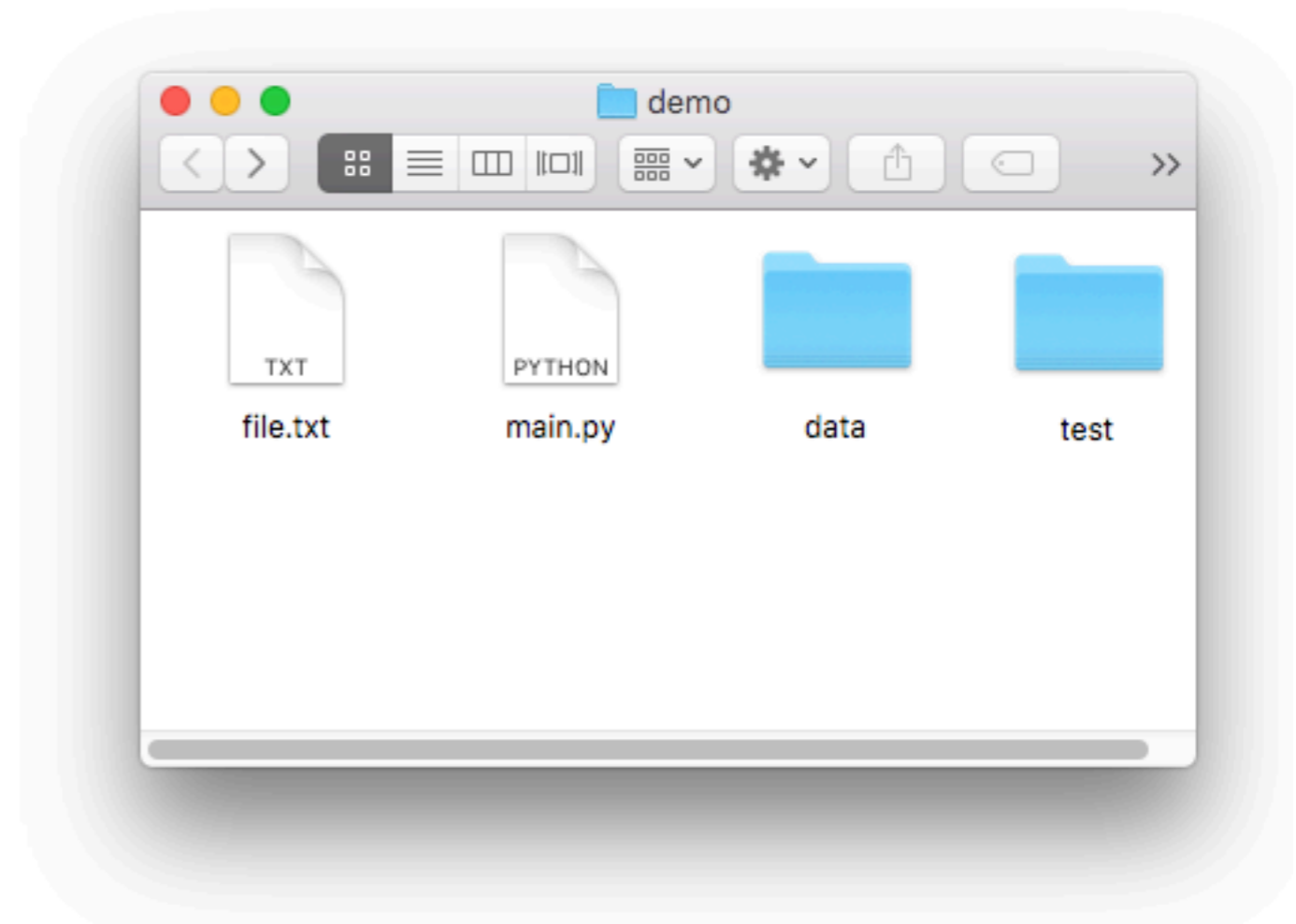- os.path.join

```
>>> 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
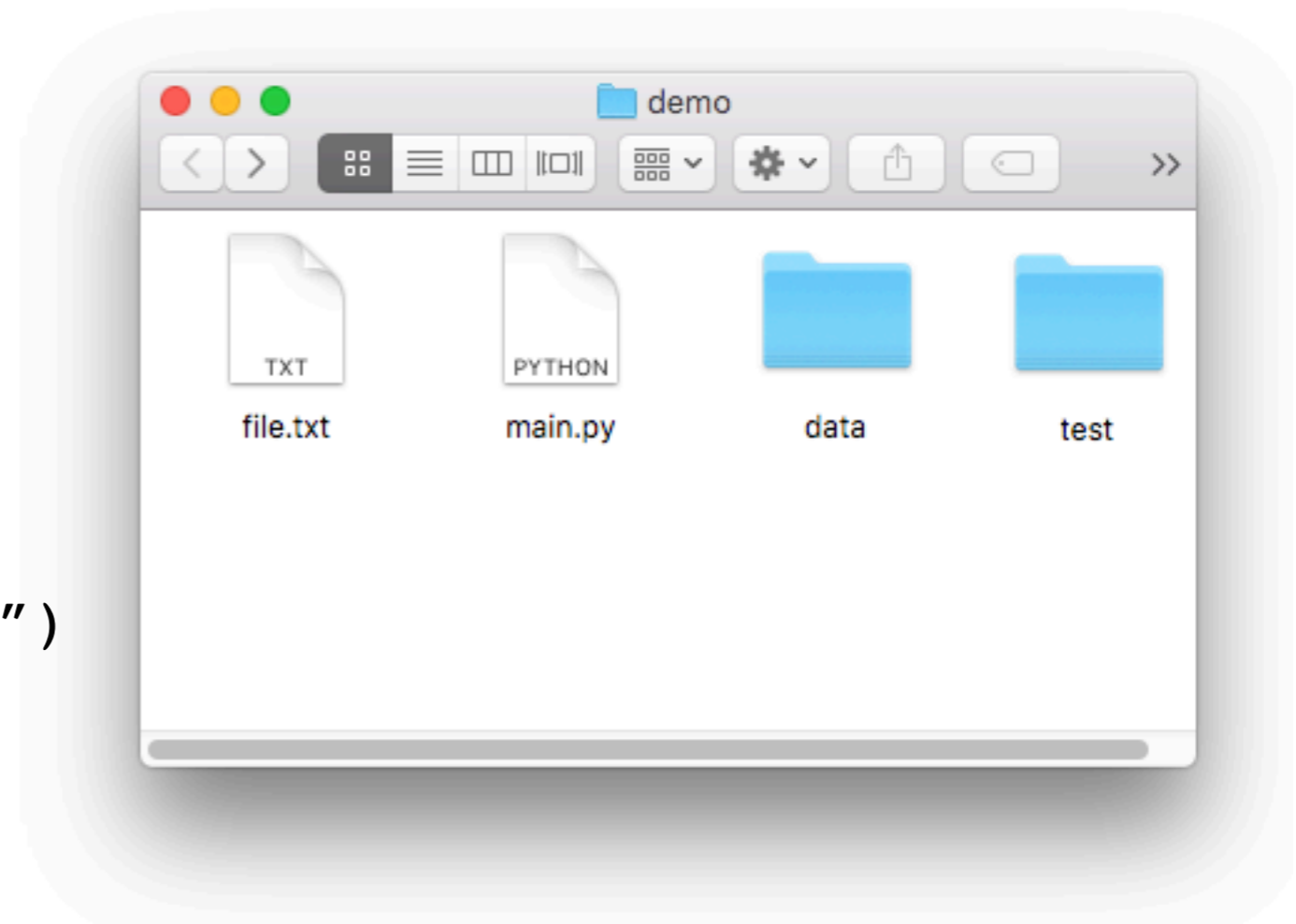- os.path.join

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

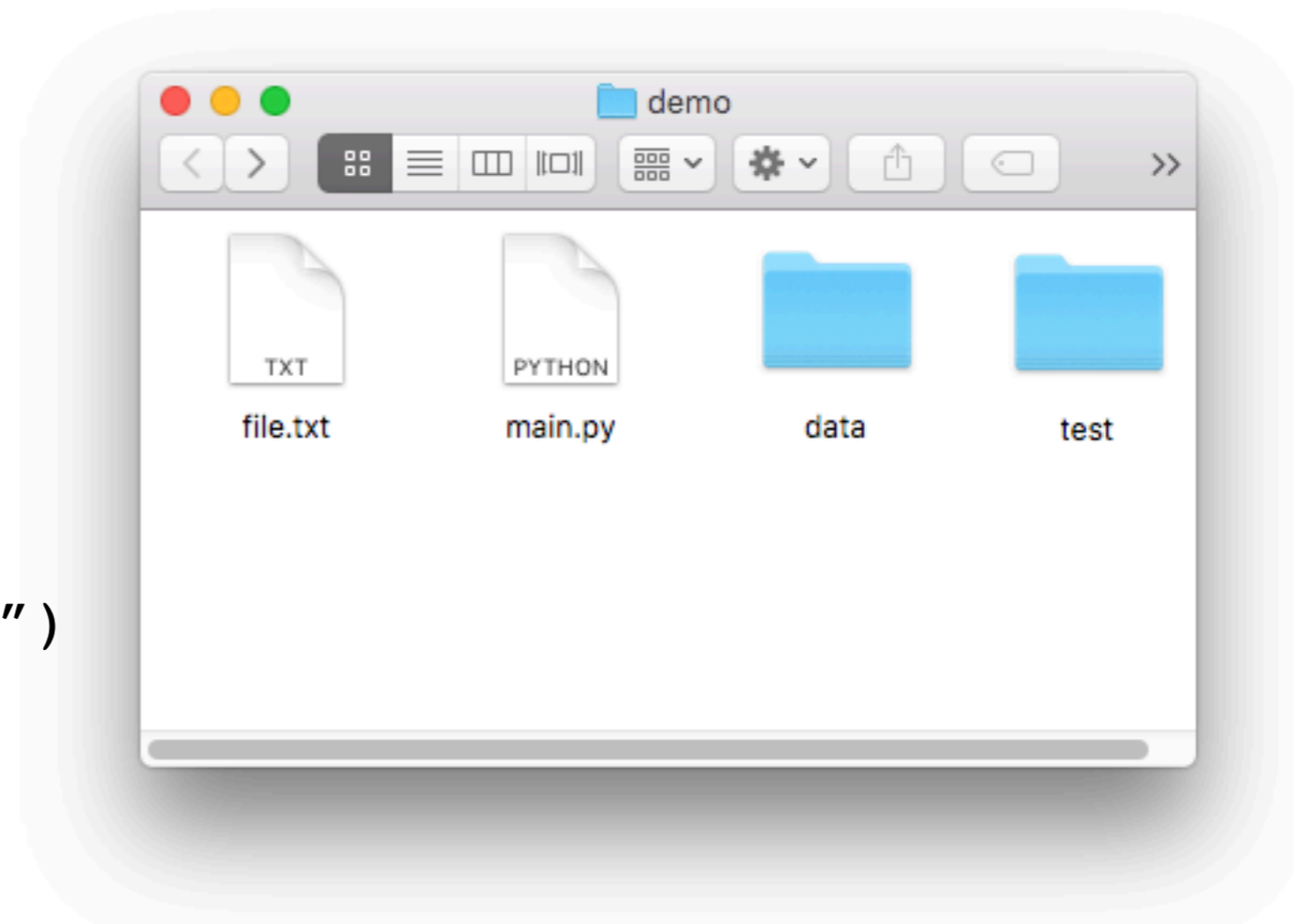# OS Module (Operating System)

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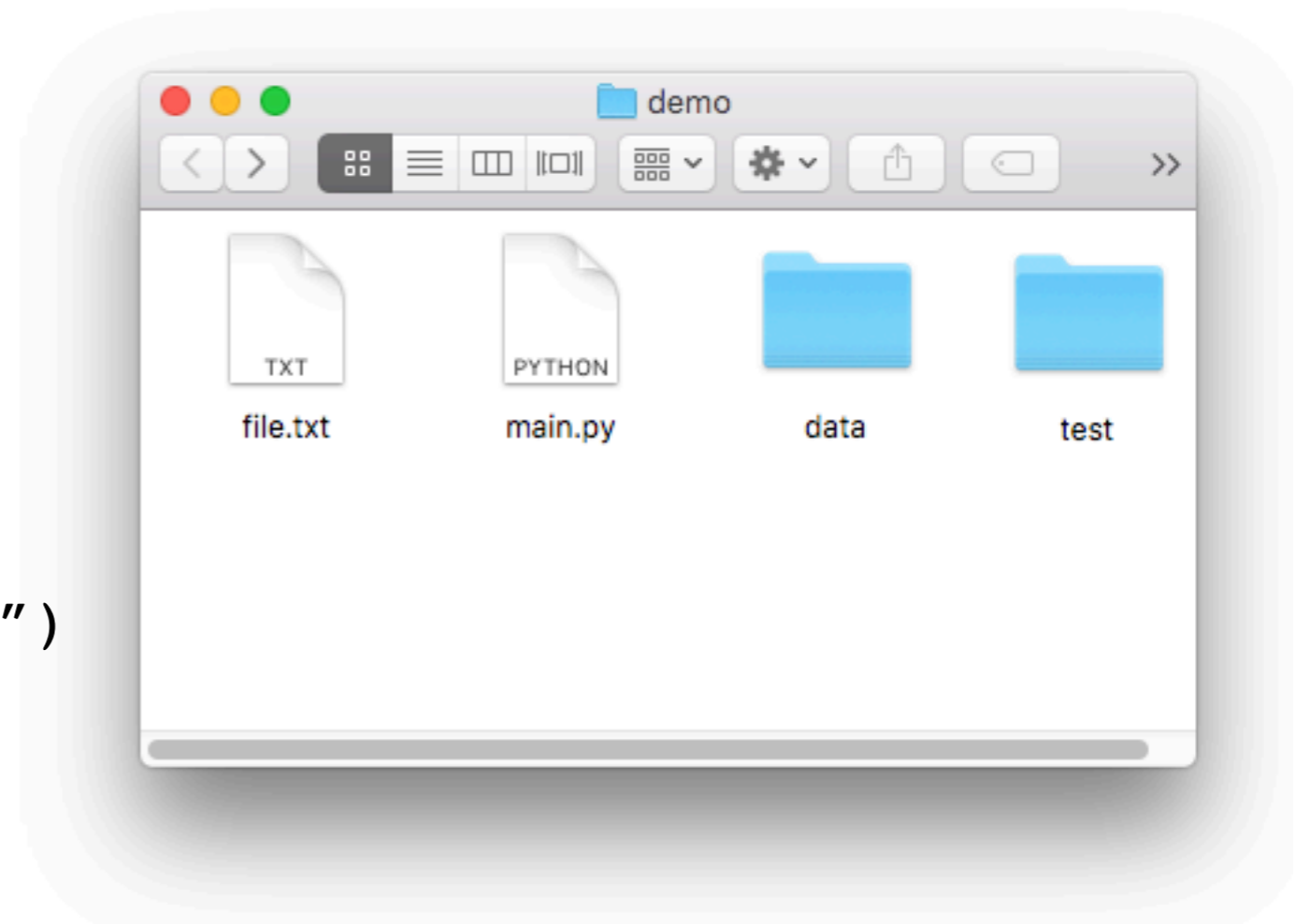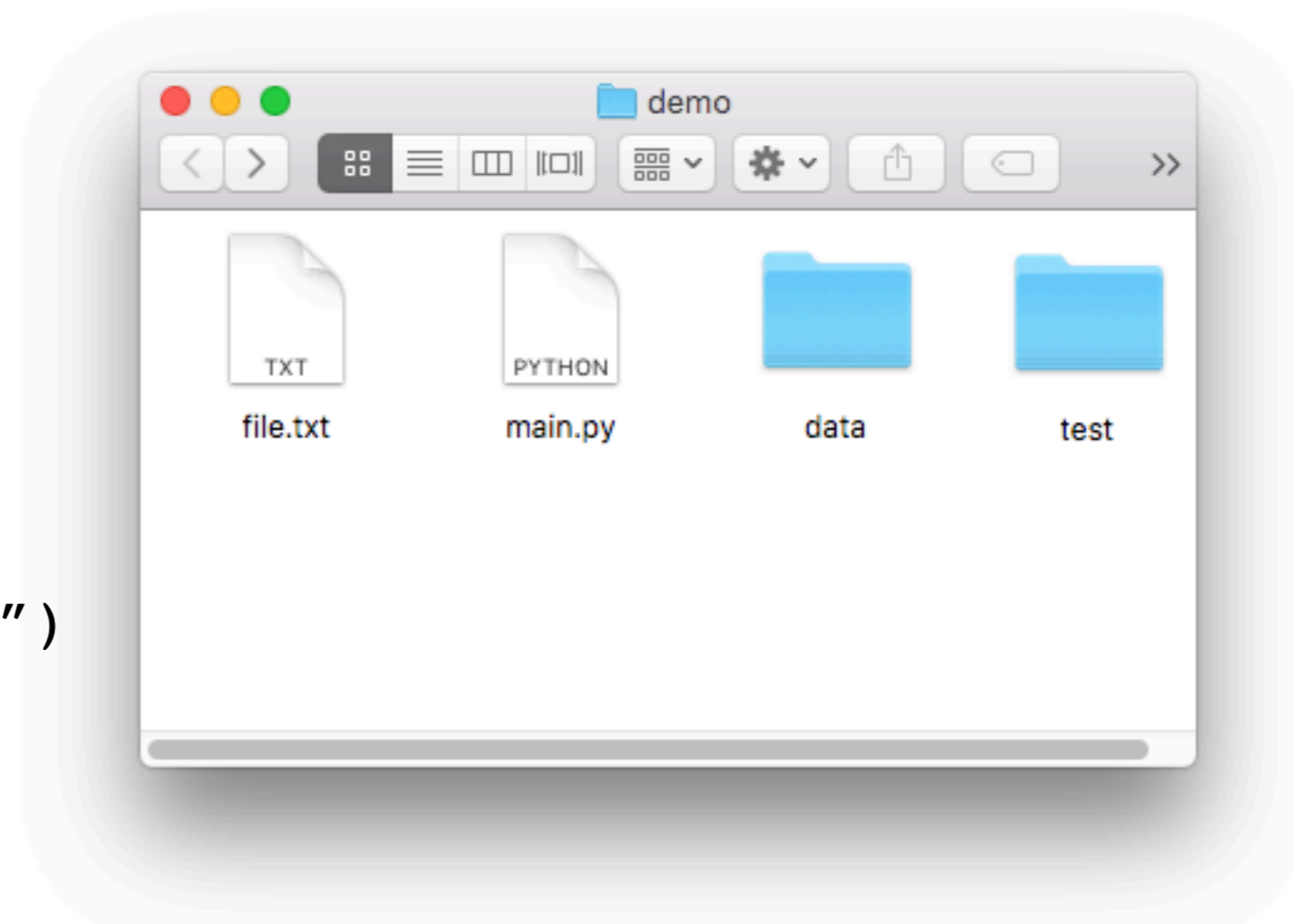
```
>>> import os
>>> os.path.exists("haha.txt")
False
```

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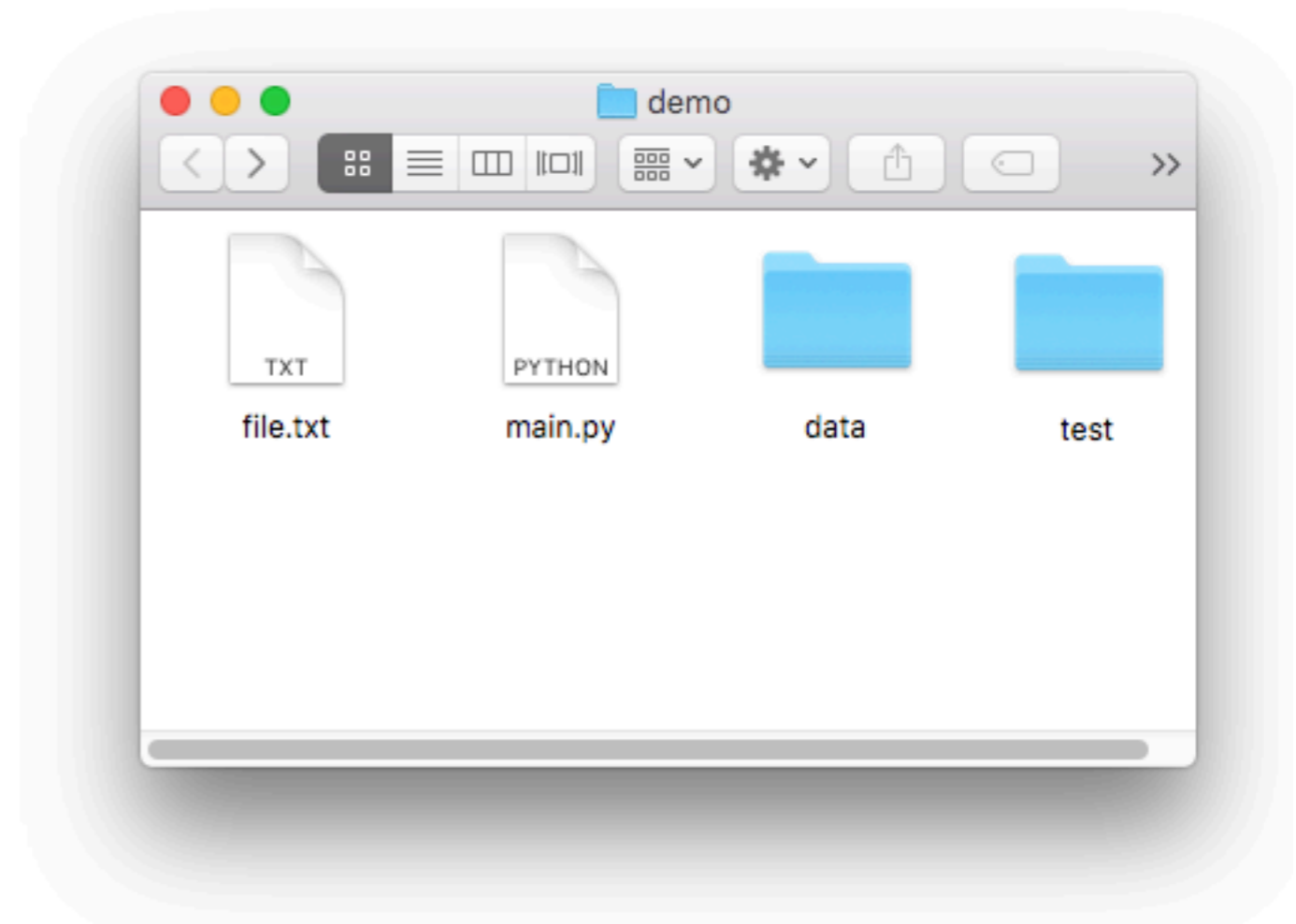
```
>>> import os
>>> os.path.isfile("file.txt")
True
```

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
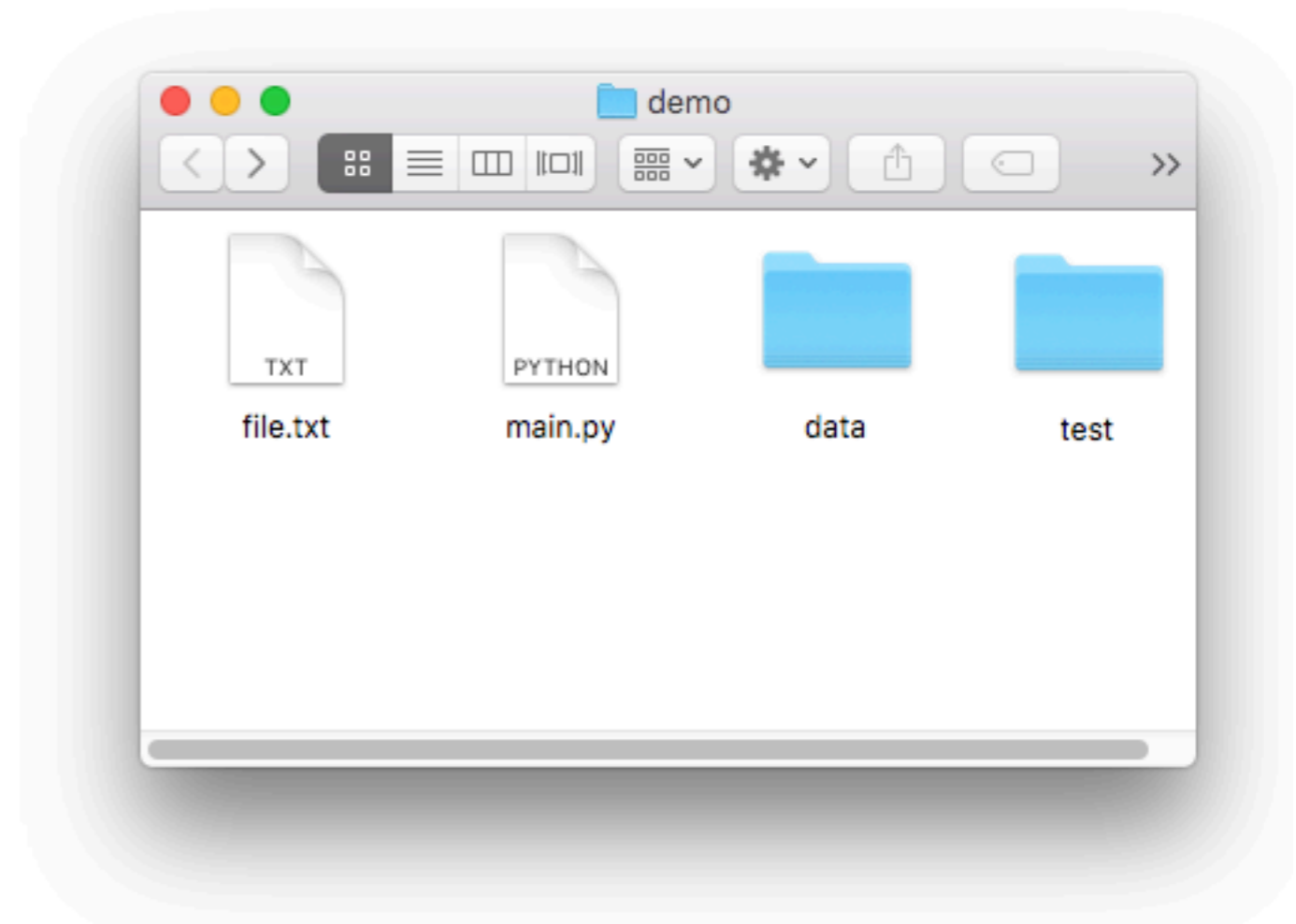- os.path.join

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

OS Module (Operating System)

Many functions in os and os.path for working w/ files
- os.listdir
- os.mkdir
- os.path.exists
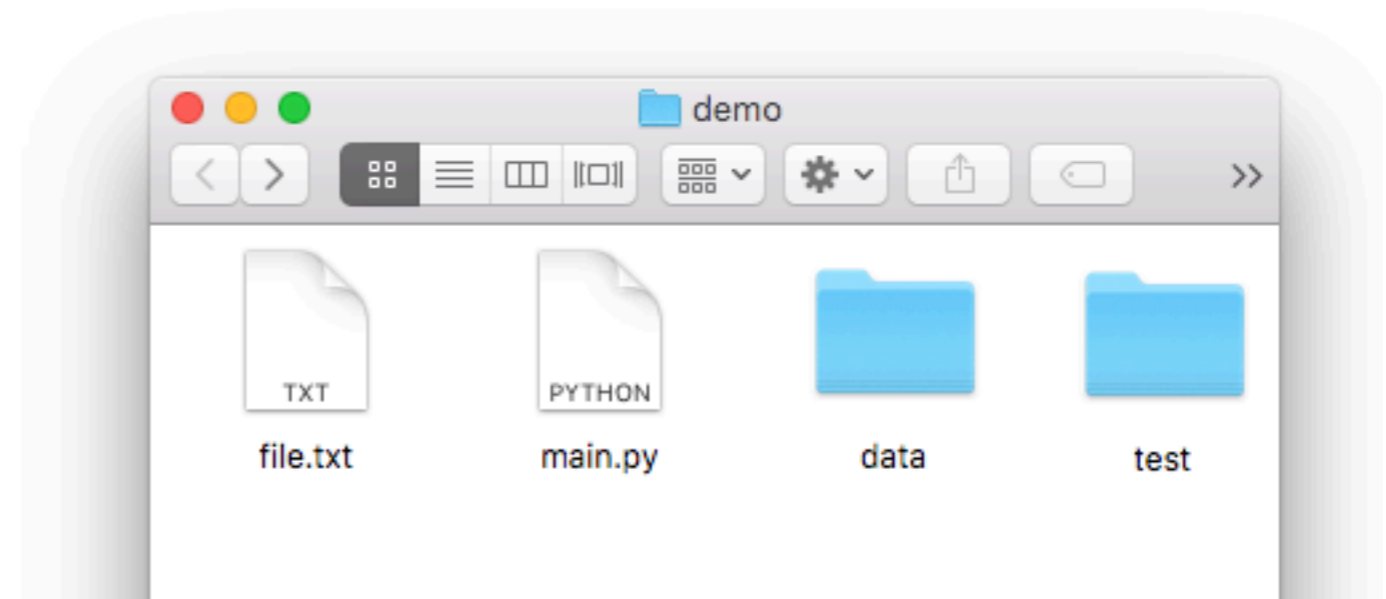- os.path.isfile
- **os.path.isdir**
- os.path.join

```
>>> import os
>>> os.path.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
```
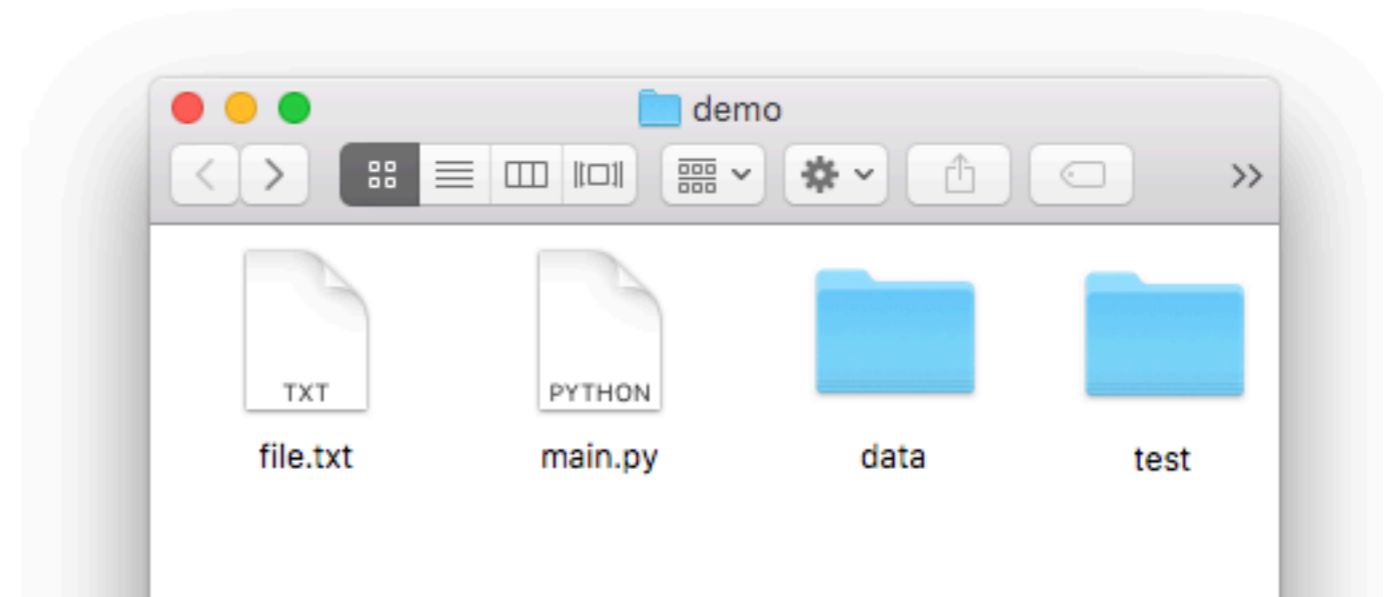
↑

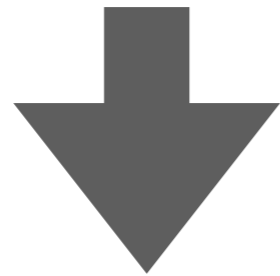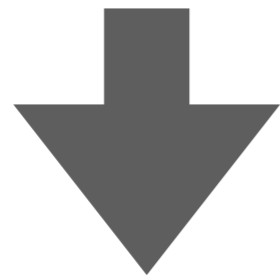on Windows

Windows

Your project:

```
path = "/".join("data", "movies.csv")
f = open(path)
...
```
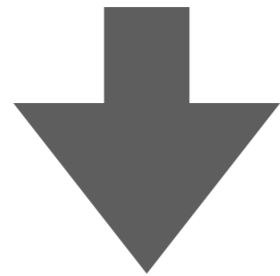
you run test.py

Windows

Your project:

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

you run test.py

✅

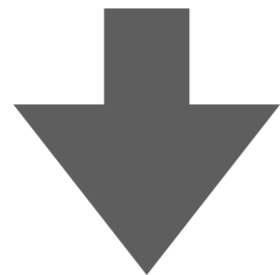submit

Linux

```
...
```

we run test.py

?

Windows

Your project:

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

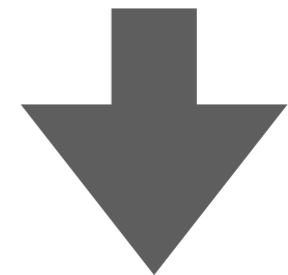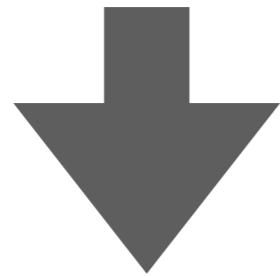**you run test.py**

✓
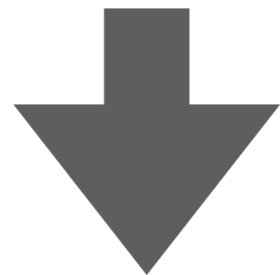
Linux

submit

```
...
```
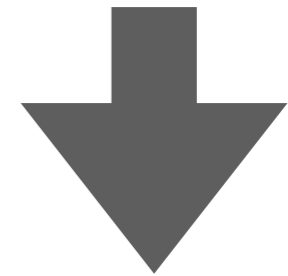
we run test.py
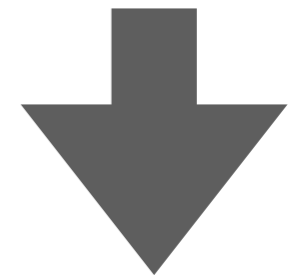
✗

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

Exceptions

Working with files leads to many exceptions
- missing files
- lacking permissions
- not enough space
- mixing up directories and files
- corrupt formats
- etc, etc

```python
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


```python
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

run again: CRASH!

```python
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'
```
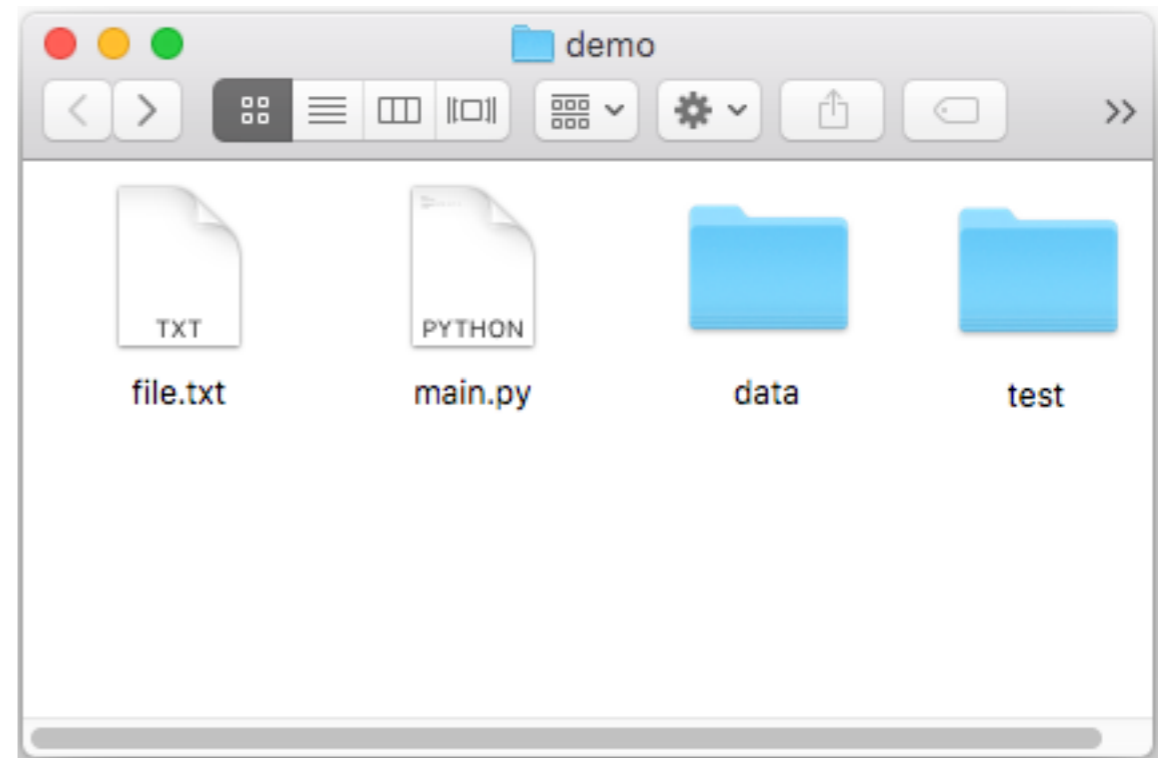
Exceptions

Working with files leads to many exceptions
- missing files
- lacking permissions
- not enough space
- mixing up directories and files
- corrupt formats
- etc, etc

run again with try/except
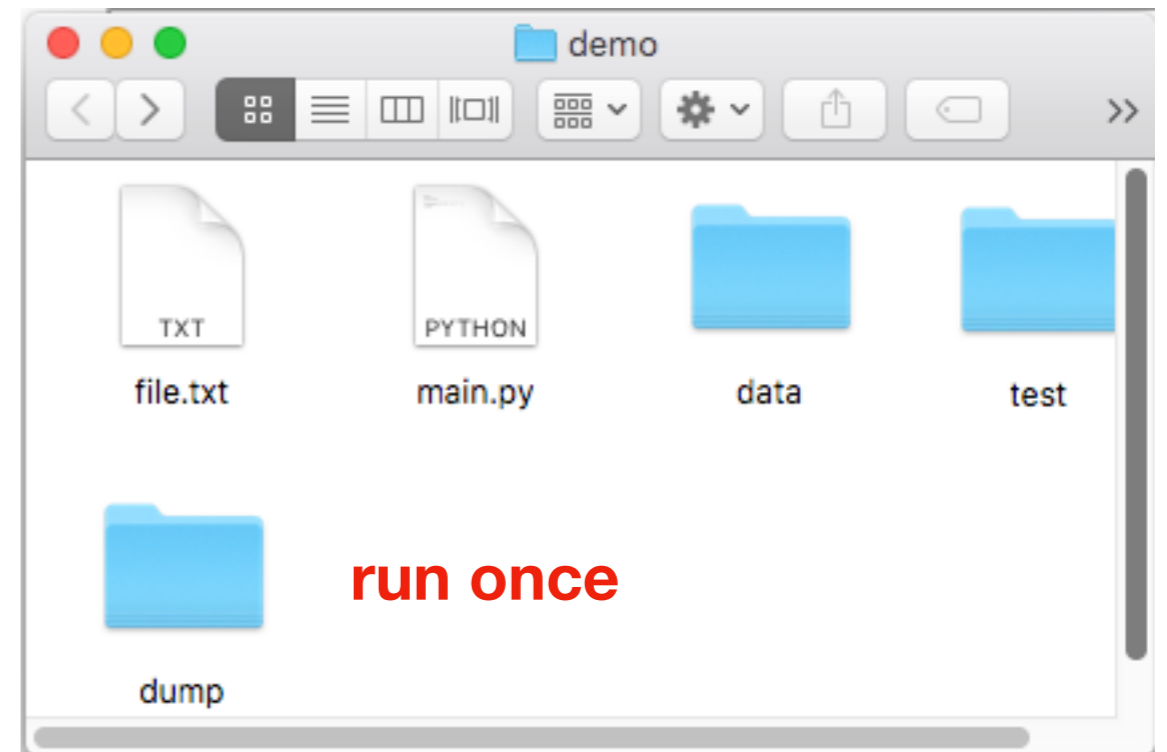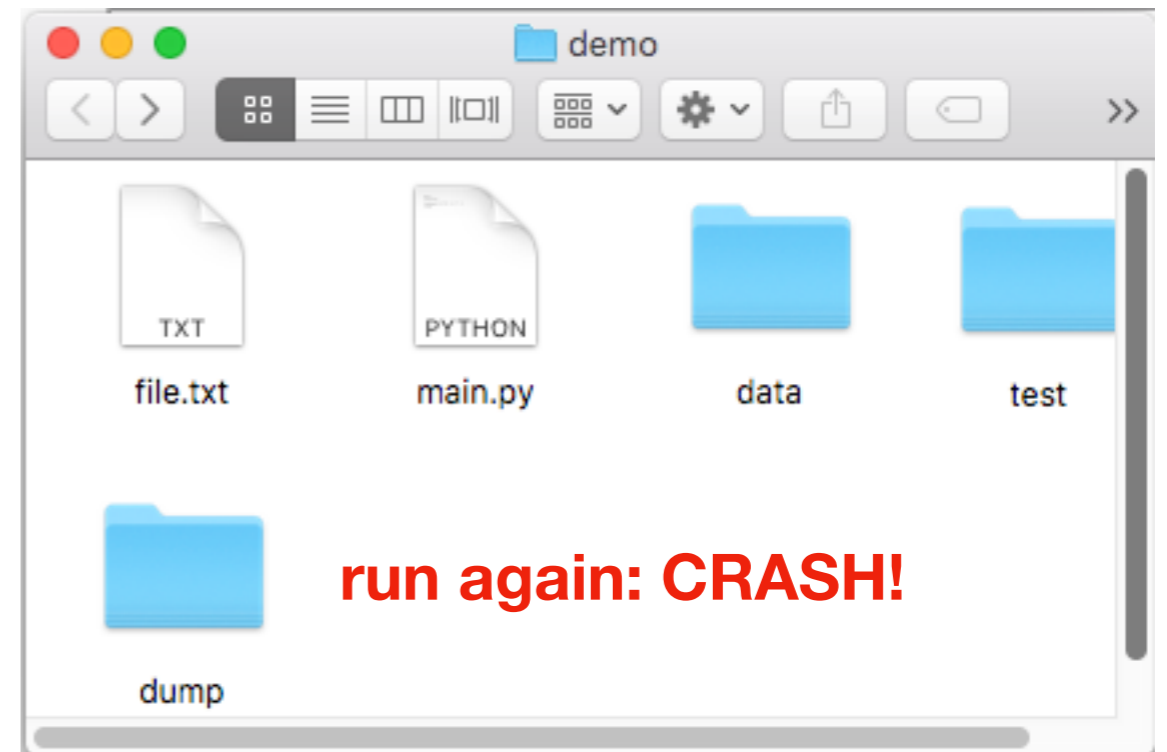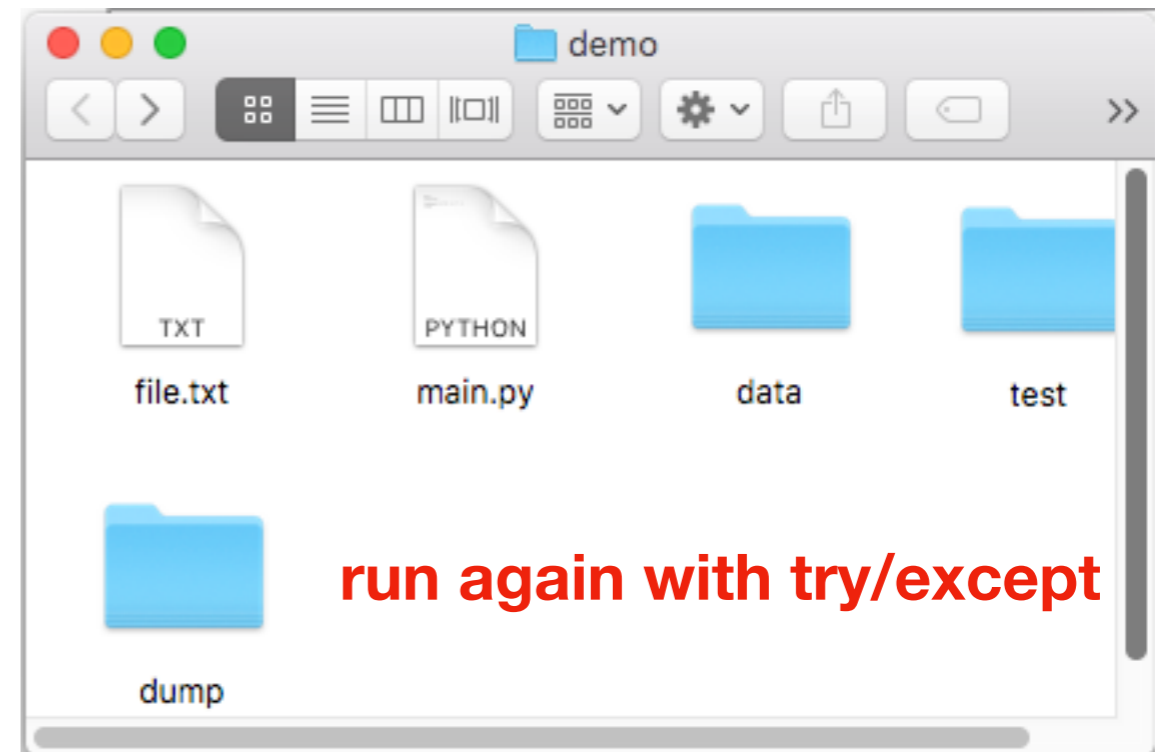
```python
import os

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

encoding 1

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 2

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 Defaults Done Wrong

Mac

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

example.txt

Windows

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

example.txt

Windows computer prints **"baÃ±o"** instead of **"baño"**

Encoding Defaults Done Wrong

Mac

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

Windows

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

example.txt

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: 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
```

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