

# [301] Files

Tyler Caraza-Harter

# 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)
```



file object

```
# read data from f
```

```
# OR
```

```
# write data to f
```

```
f.close()
```

# 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)
```

file object

file path

```
# read data from f
```

```
# OR
```

```
# write data to f
```

```
f.close()
```

# File objects

main.py: `f = open(path)`

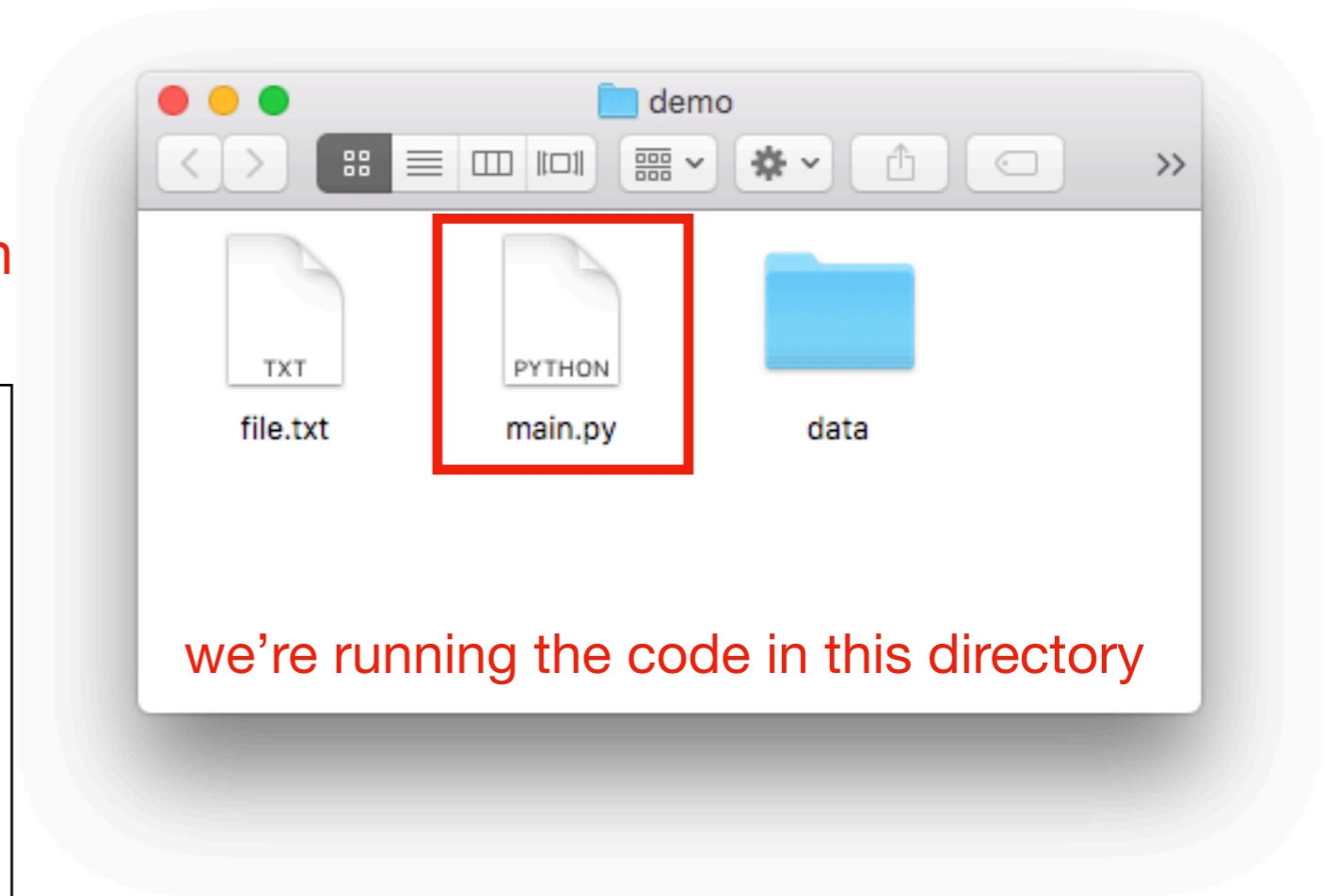
built-in open function

file object

file path

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

```
f.close()
```





# File objects

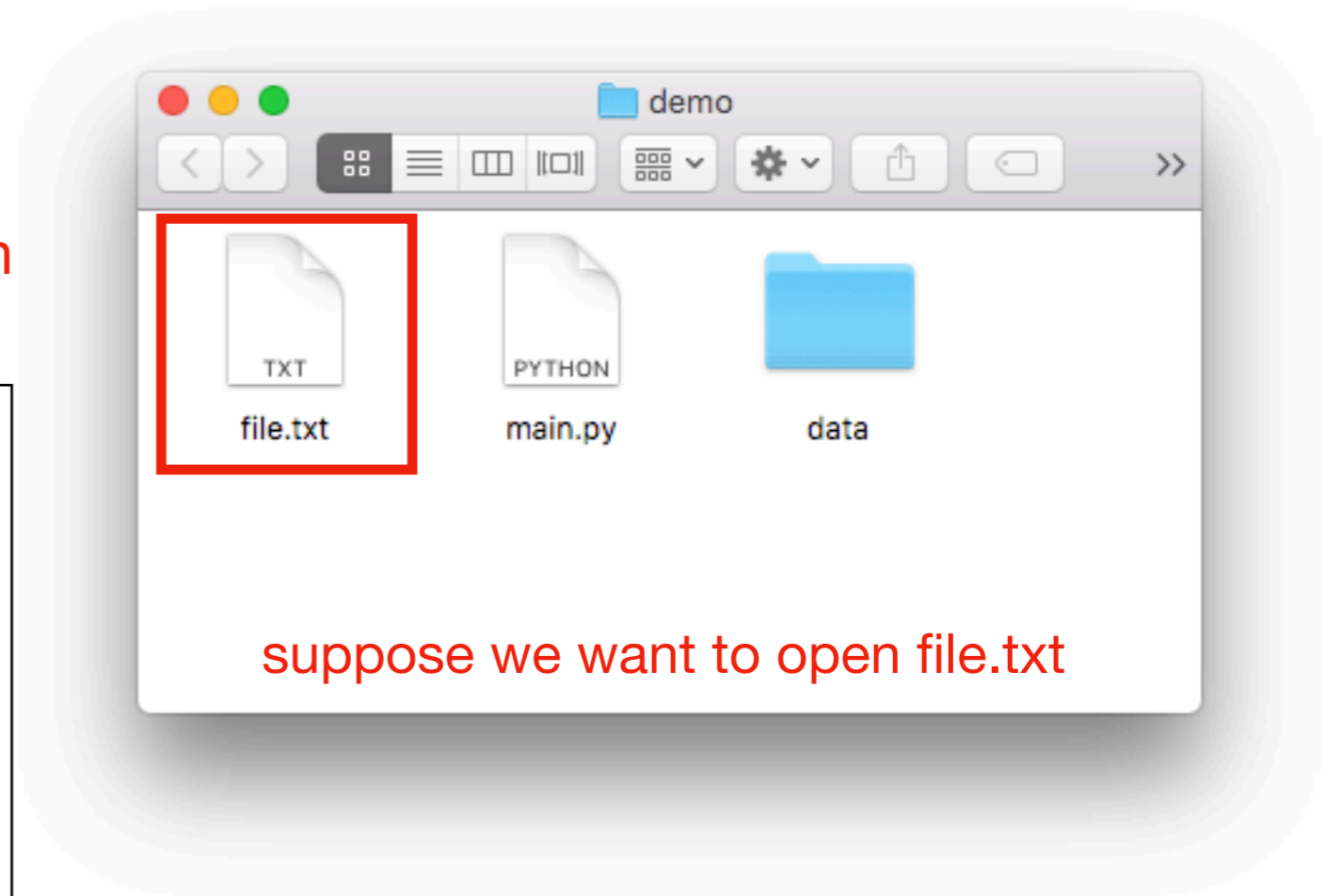
main.py:

```
f = open("file.txt")  
  
# read data from f  
# OR  
# write data to f  
  
f.close()
```

built-in open function

file object

file path



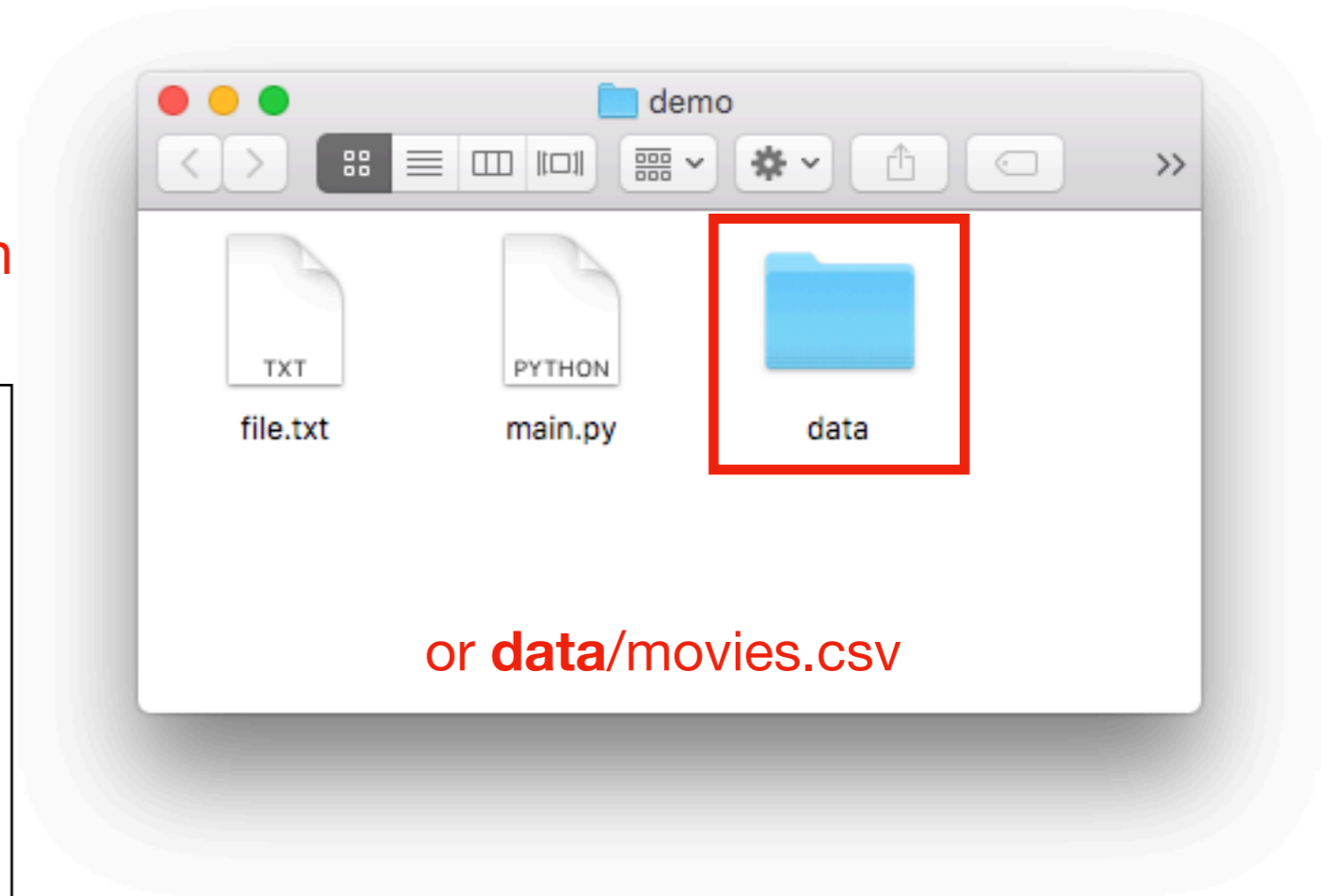
# File objects

main.py: `f = open(`  
          `"data/movies.csv")`  
          ↑                  ↑  
          file object      file path

built-in open function

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

f.close()
```



# File objects

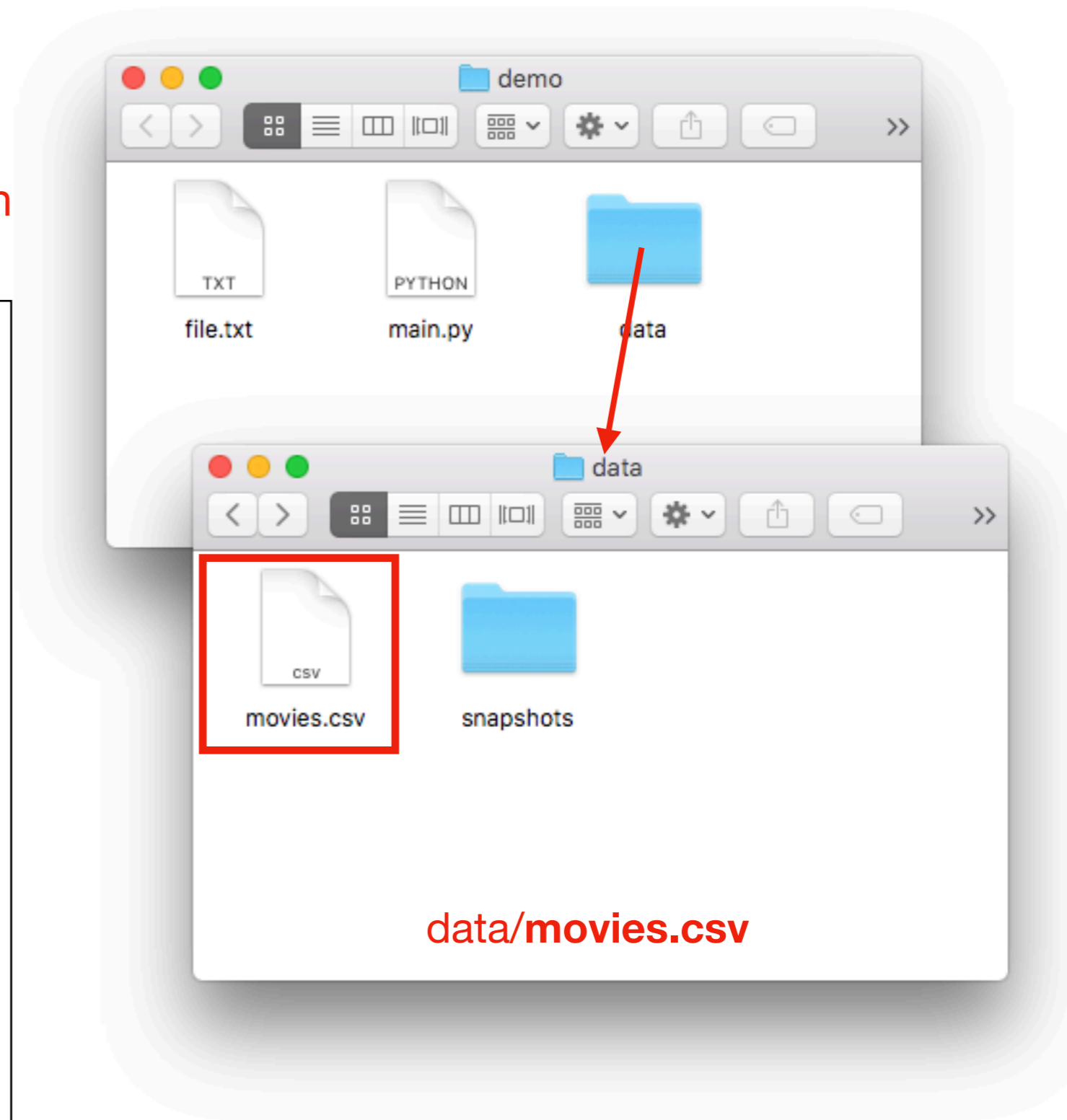
main.py:

```
f = open(  
    "data/movies.csv")  
  
# read data from f  
# OR  
# write data to f  
  
f.close()
```

built-in open function

file object

file path



# File objects

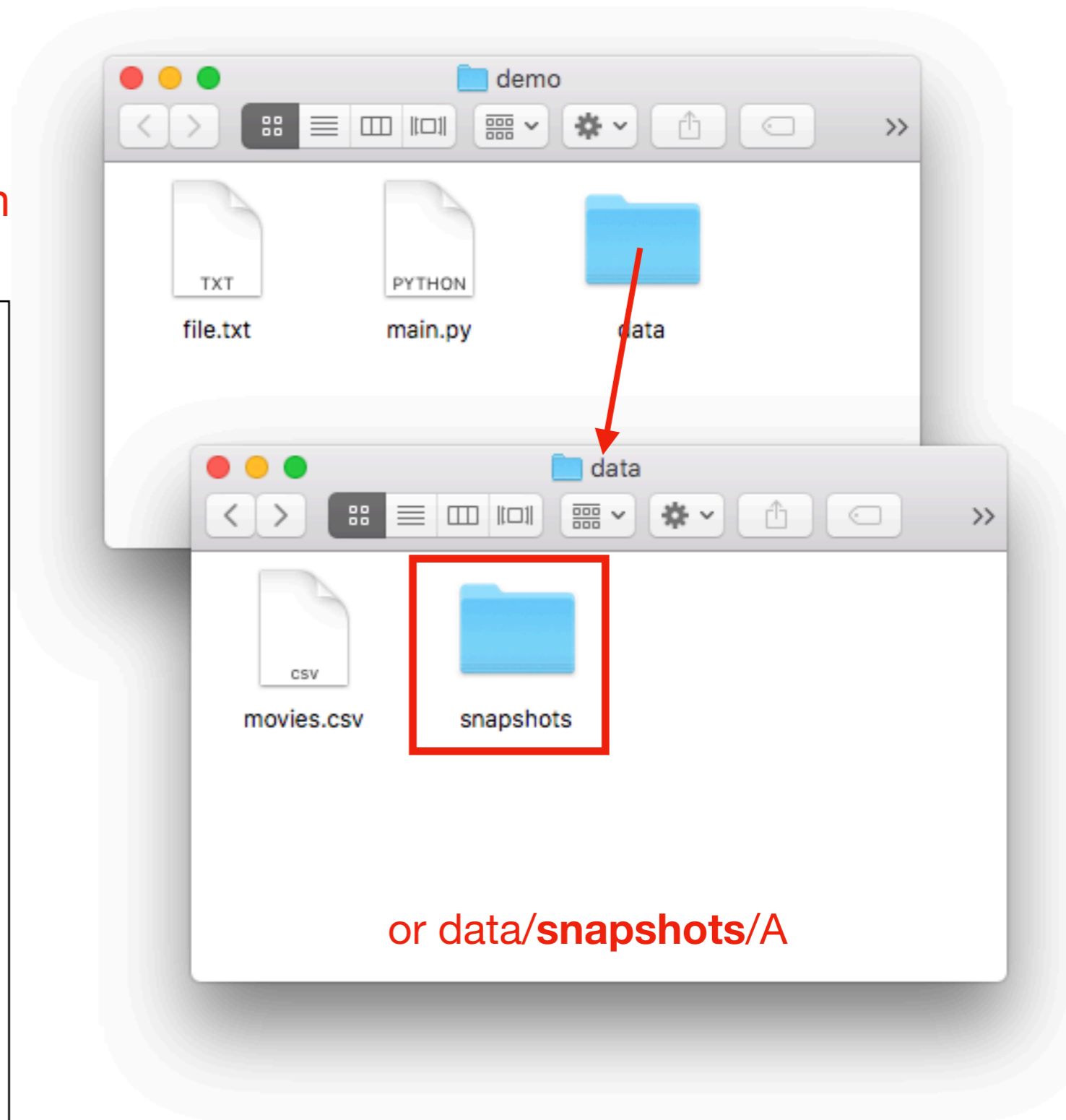
main.py:

```
f = open(  
    "data/snapshots/A"  
)  
  
# read data from f  
# OR  
# write data to f  
  
f.close()
```

built-in open function

file object

file path



# File objects

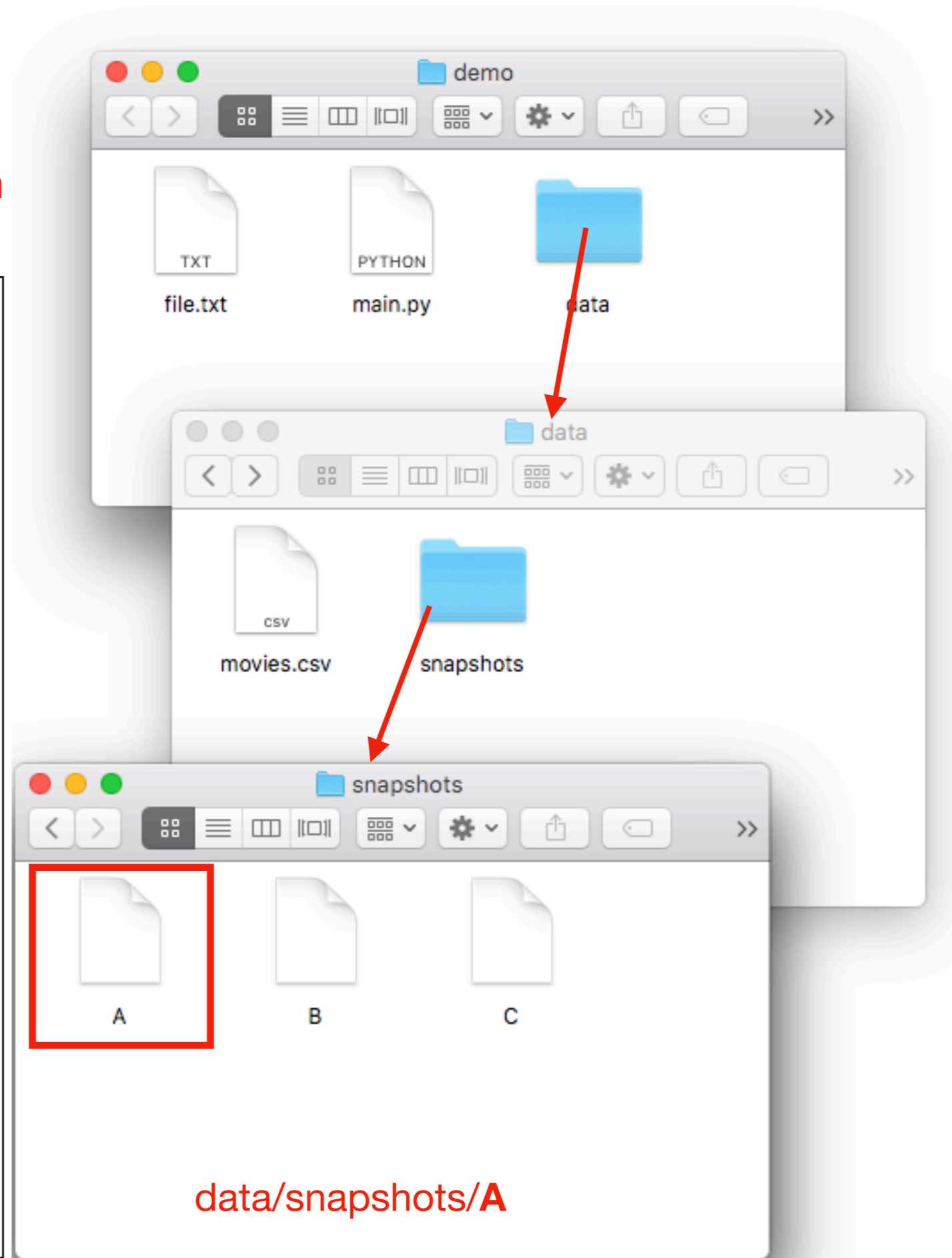
main.py:

```
f = open(  
    "data/snapshots/A")  
  
# read data from f  
# OR  
# write data to f  
  
f.close()
```

built-in open function

file object

file path



# File objects

```
main.py: f = open("file.txt")  
  
# read data from f  
# OR  
# write data to f  
  
f.close()
```

# File objects

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

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

using file

```
f.close()
```

# File objects

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

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

using file

```
f.close()
```

cleanup



# File objects

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

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

using file

```
f.close()
```

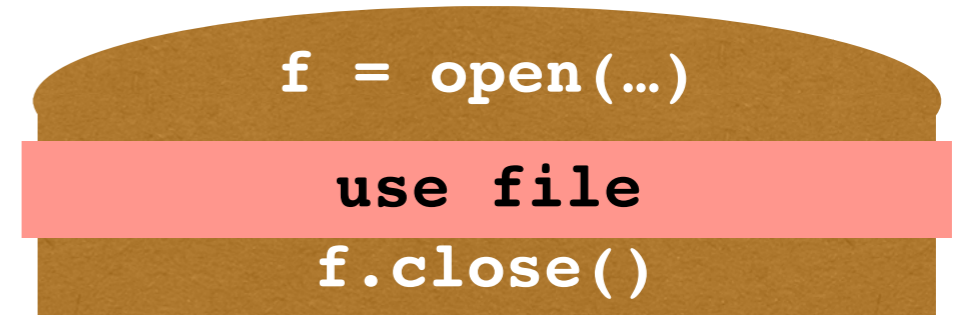
cleanup

## Reasons for closing

- avoid data loss
- limited number of open files

# 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

## File formats

- JSON
- CSV

## OS module

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

## File exceptions

## Encodings

# Reading a file

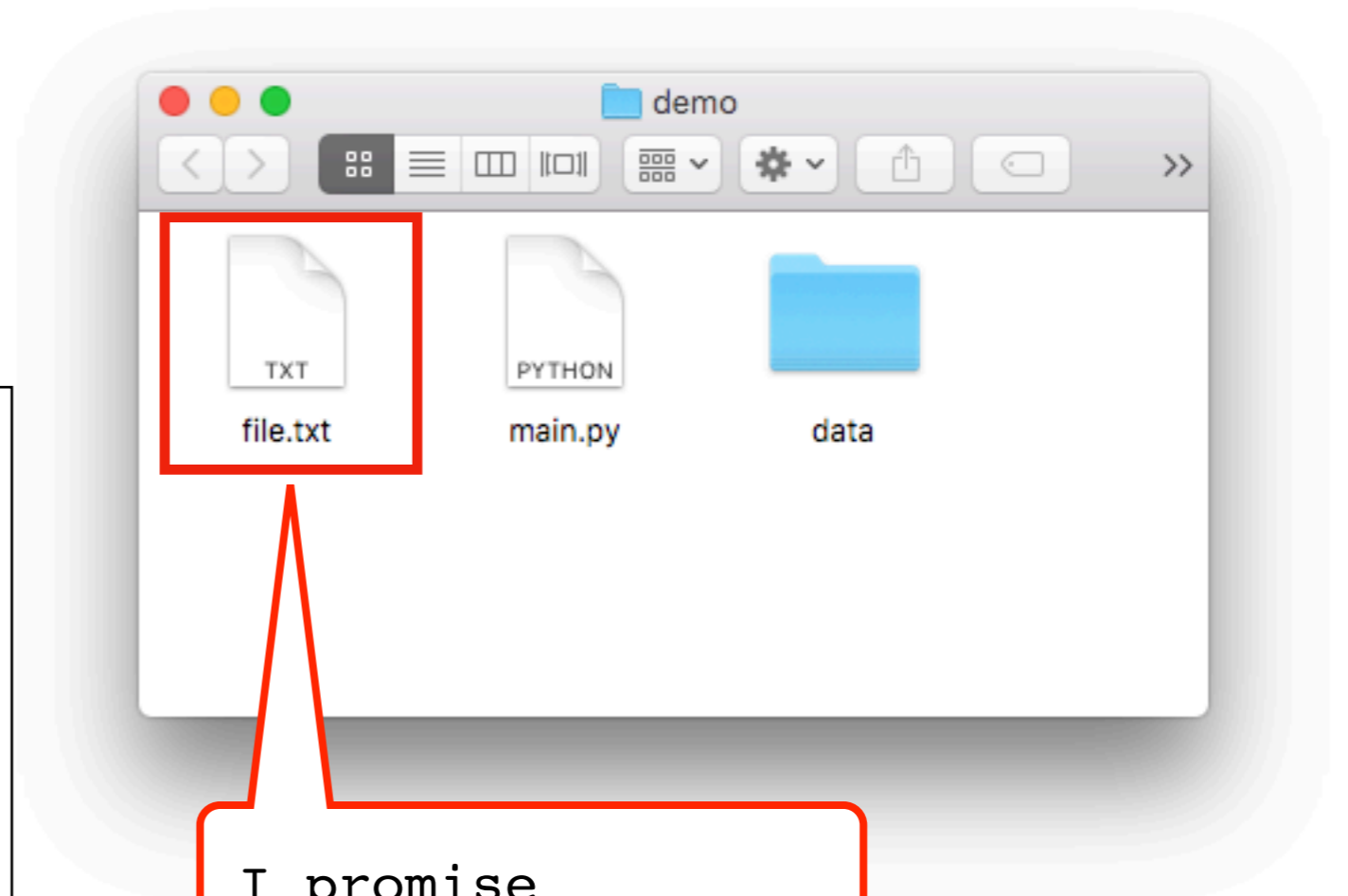
```
f = open("file.txt")
```

```
# read data from f
```

```
# OR
```

```
# write data to f
```

```
f.close()
```



I promise  
to always  
close my files

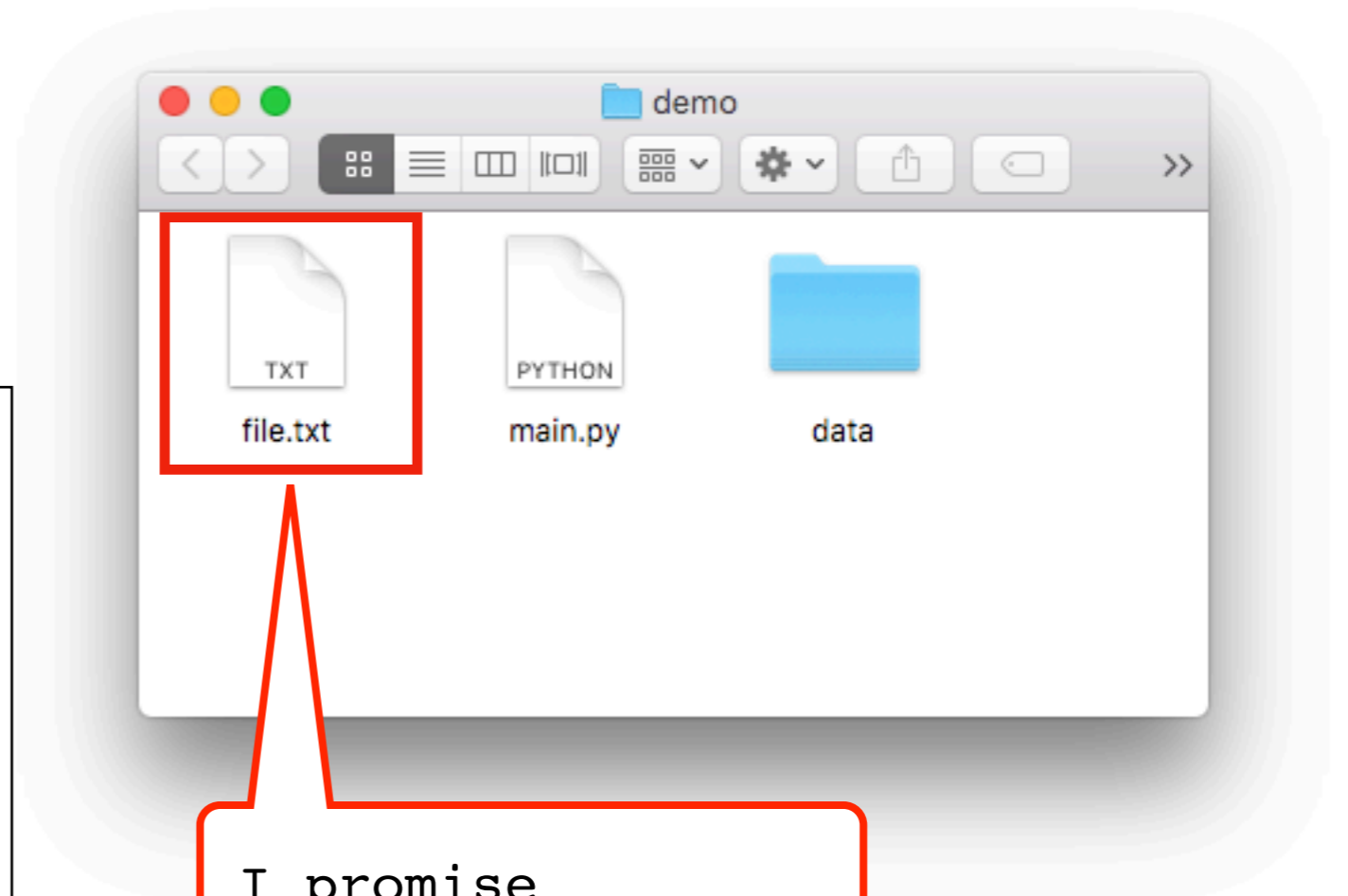
# Reading a file

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

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

```
print(data)
```

```
f.close()
```



I promise  
to always  
close my files

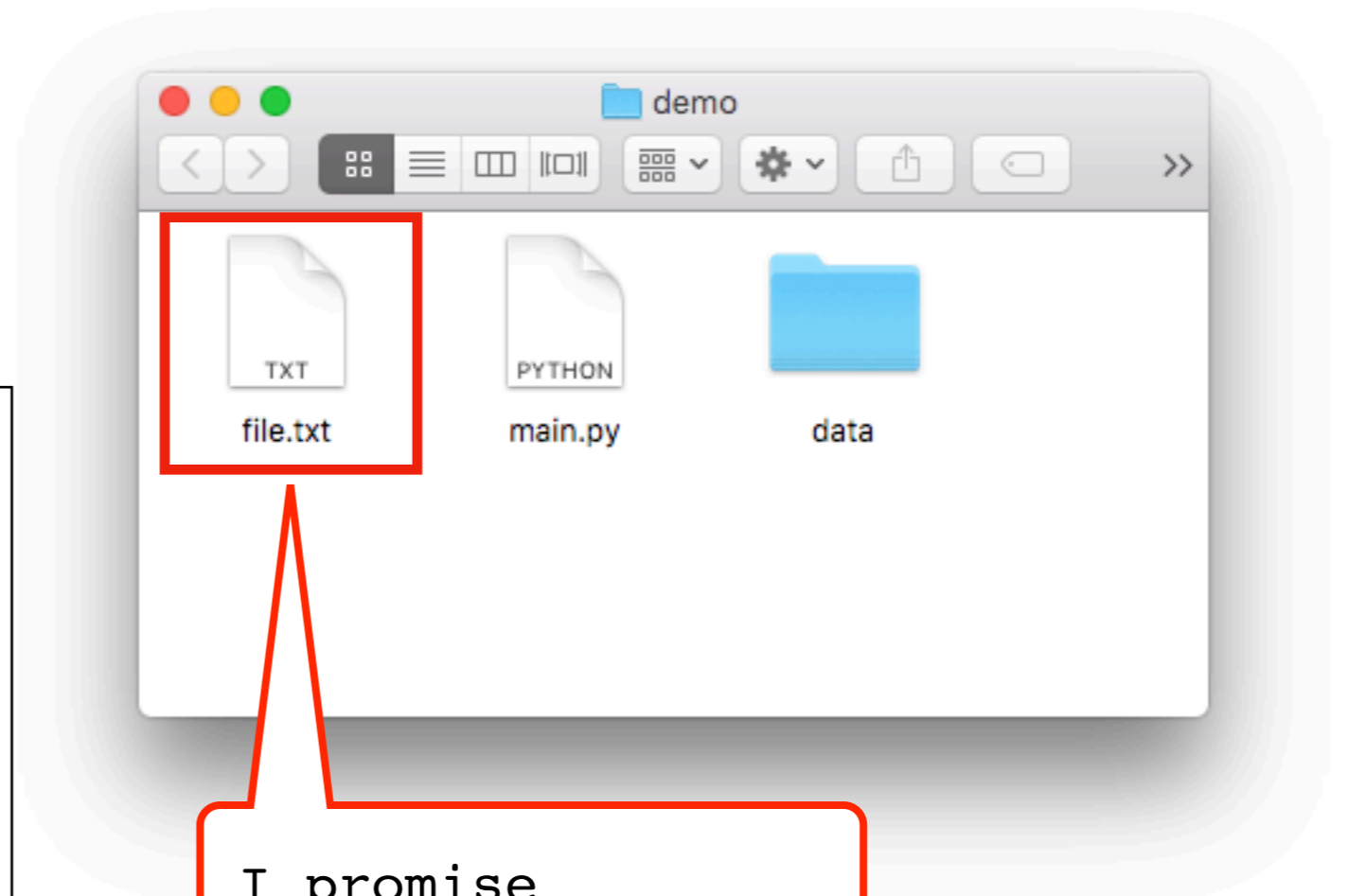
# Reading a file

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

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

```
print(data)
```

```
f.close()
```



I promise  
to always  
close my files

`read()` method

- fetch entire file contents
- return as a string

# Reading a file

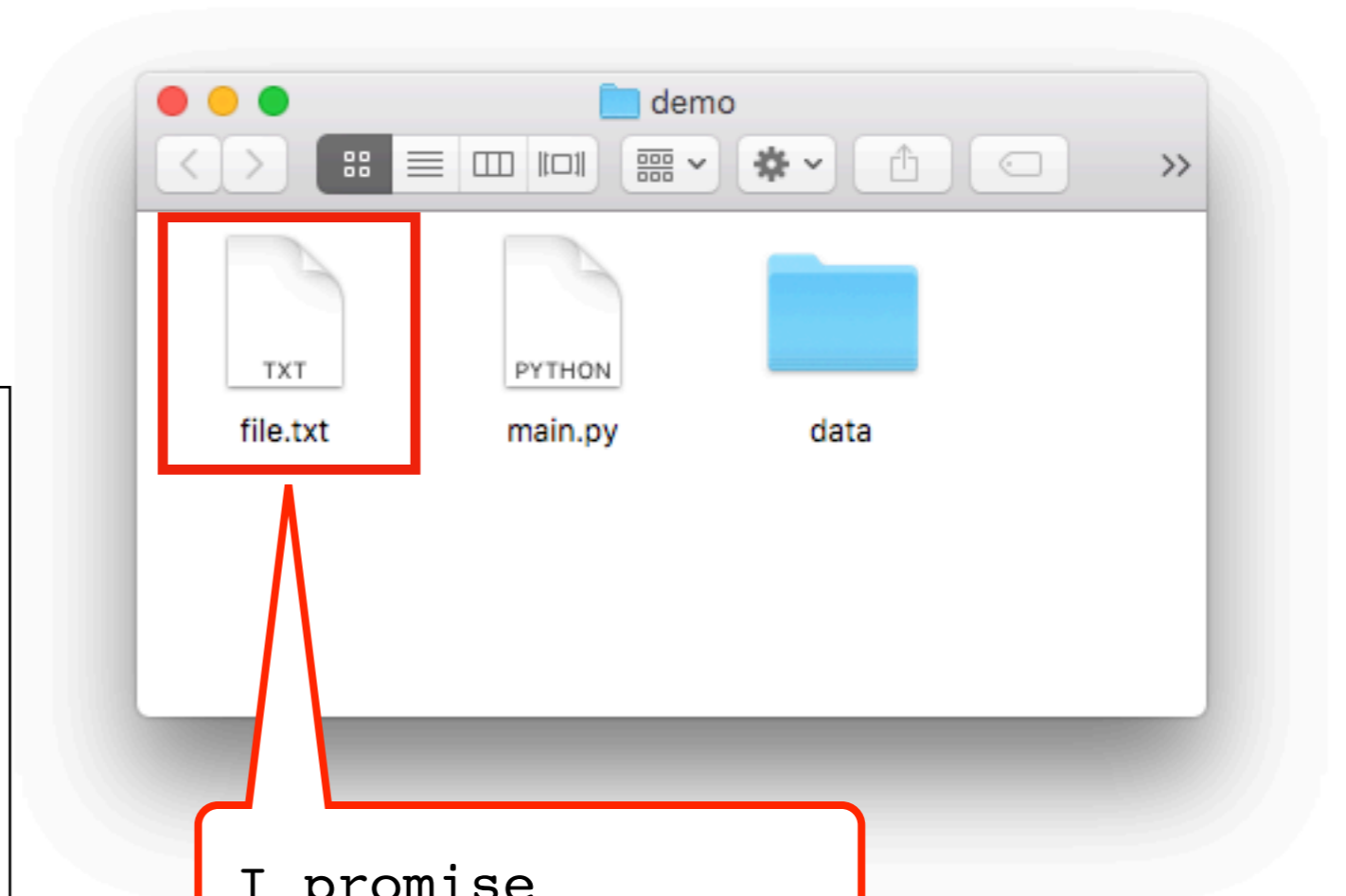
```
f = open("file.txt")
```

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

```
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

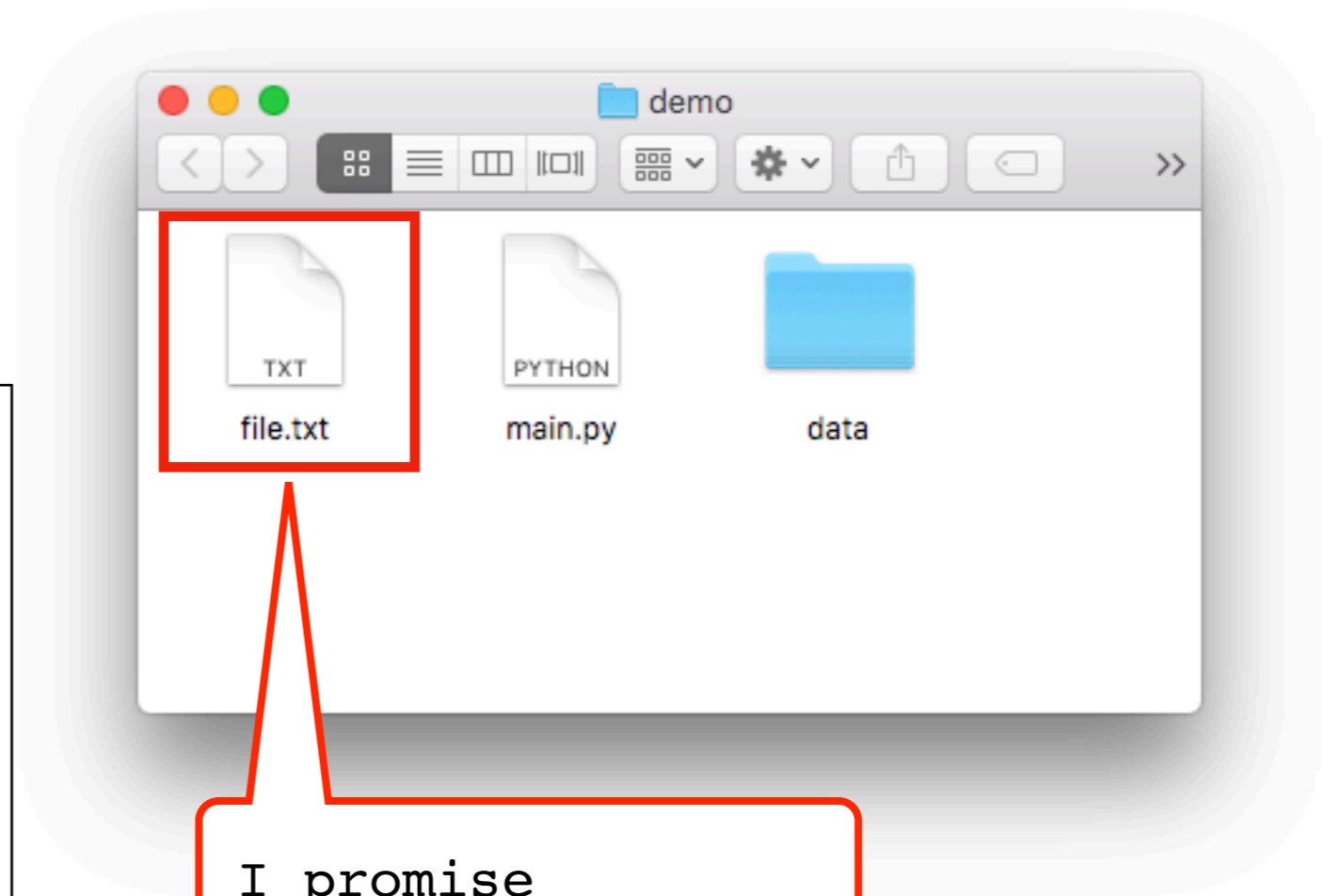
```
f = open("file.txt")

data = f.read()
data = data.split("\n")

print(data)

data is: ["I promise", "to always", "close my files"]

f.close()
```



I promise  
to always  
close my files

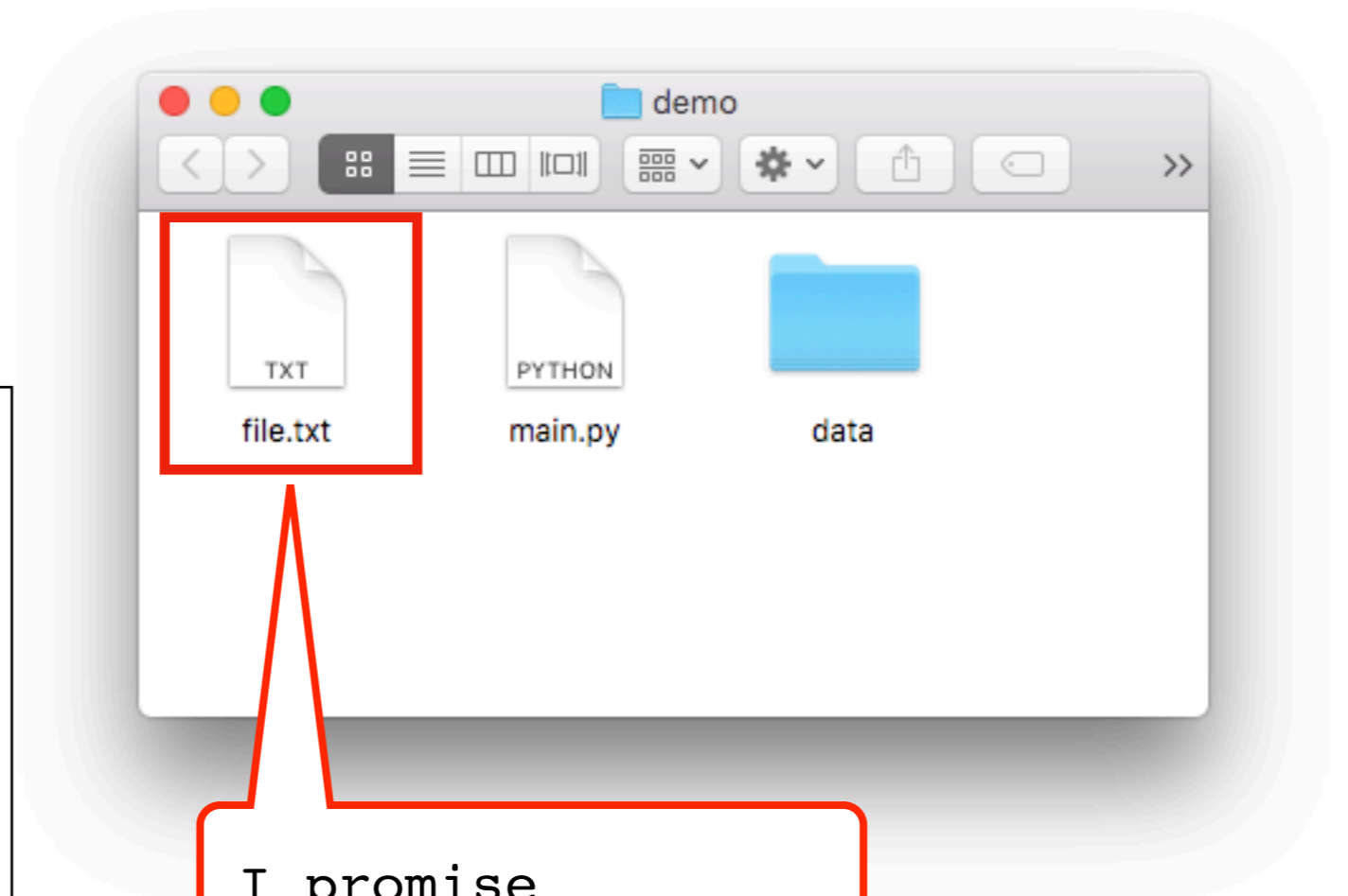


# Reading a file

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

for line in f:
    print(line.rstrip())

f.close()
```



I promise  
to always  
close my files

recall a **file object** is an **iterator**

- can loop over
- can convert to list

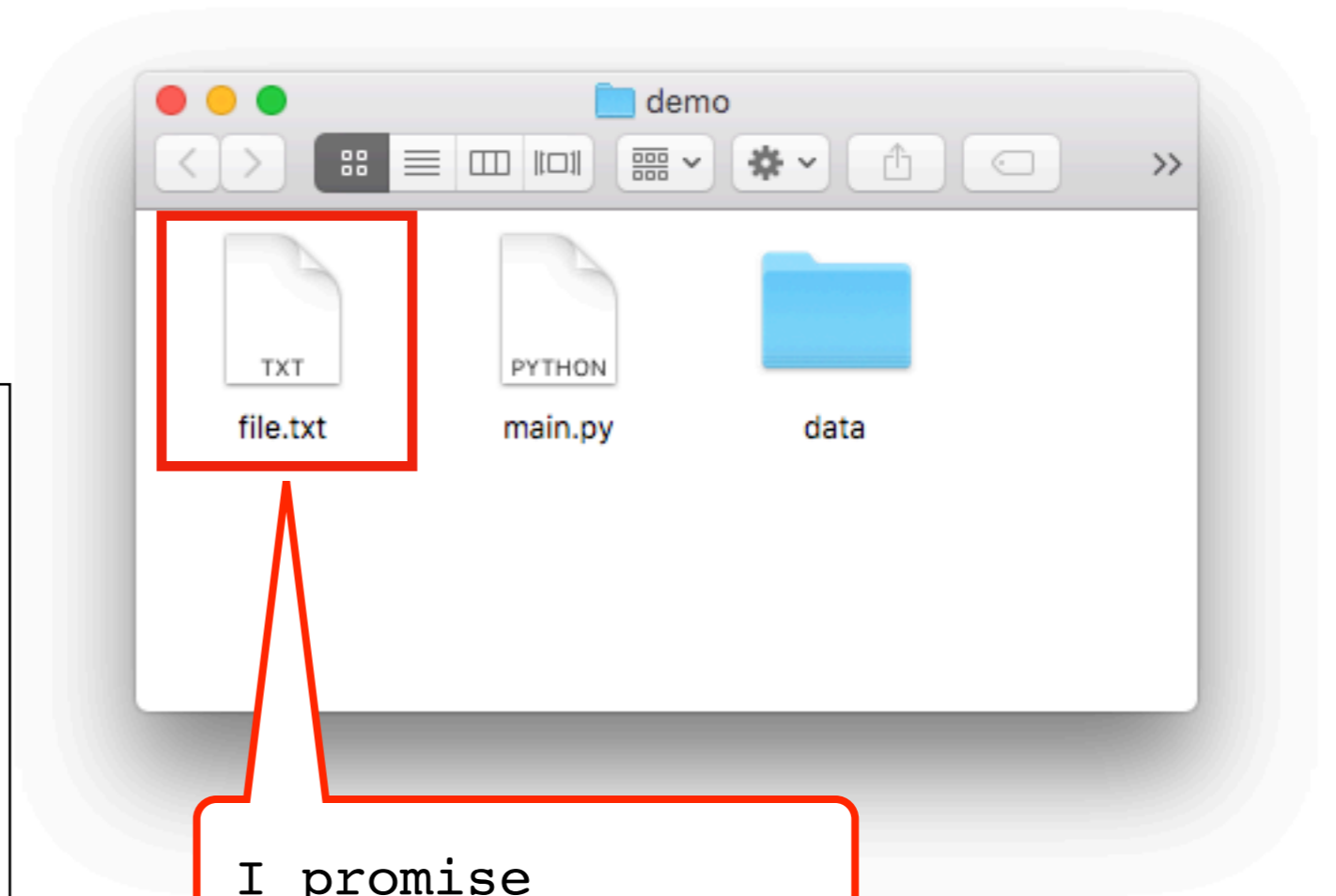
# Reading a file

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

```
lines = list(f)
```

```
f.close()
```

lines is: ["I promise\n", "to always\n", "close my files\n"]



I promise  
to always  
close my files

recall a **file object** is an **iterator**

- can loop over
- **can convert to list**

# Write a file

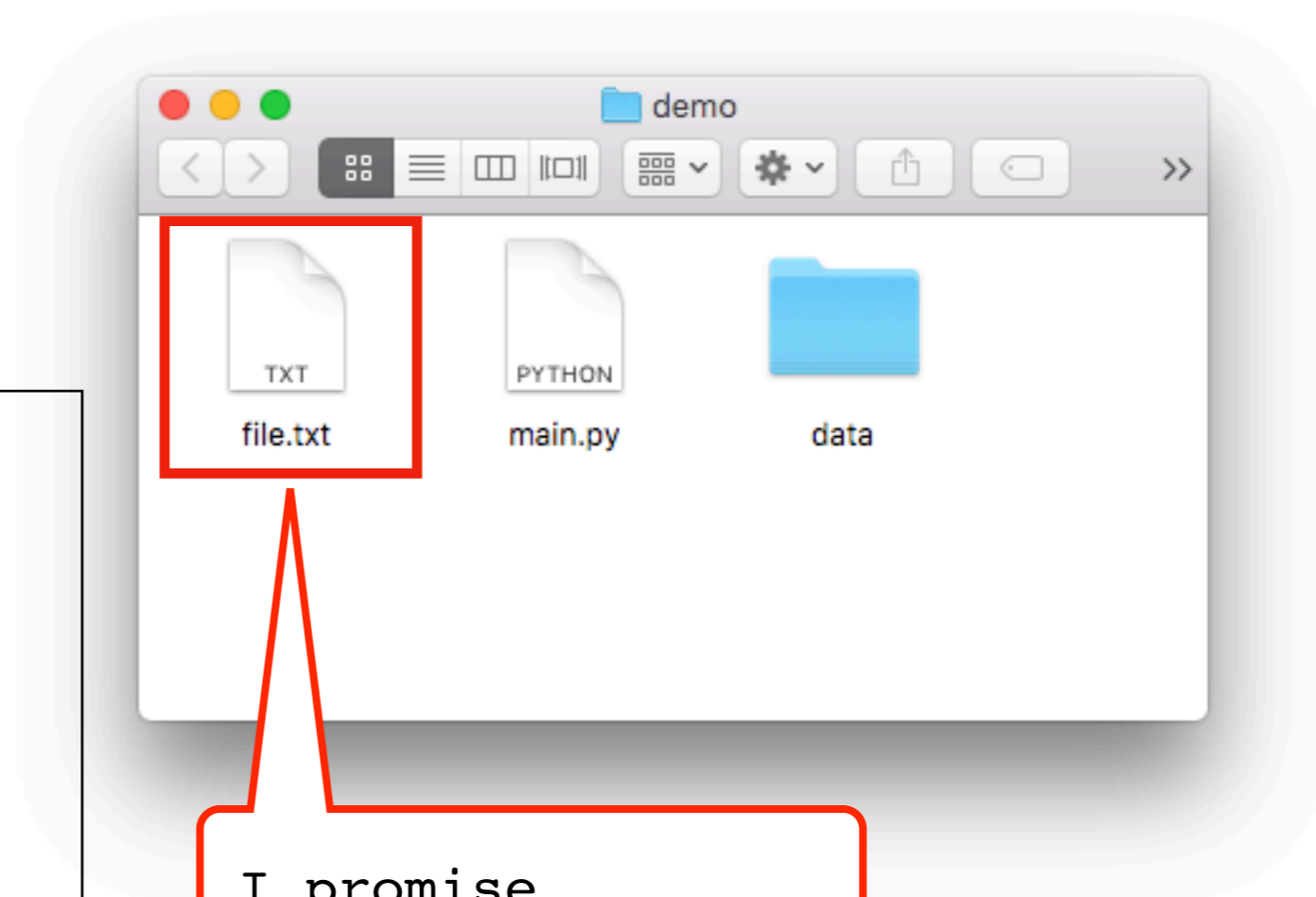
```
f = open("file.txt")
```

```
# read data from f
```

```
# OR
```

```
# write data to f
```

```
f.close()
```



I promise  
to always  
close my files

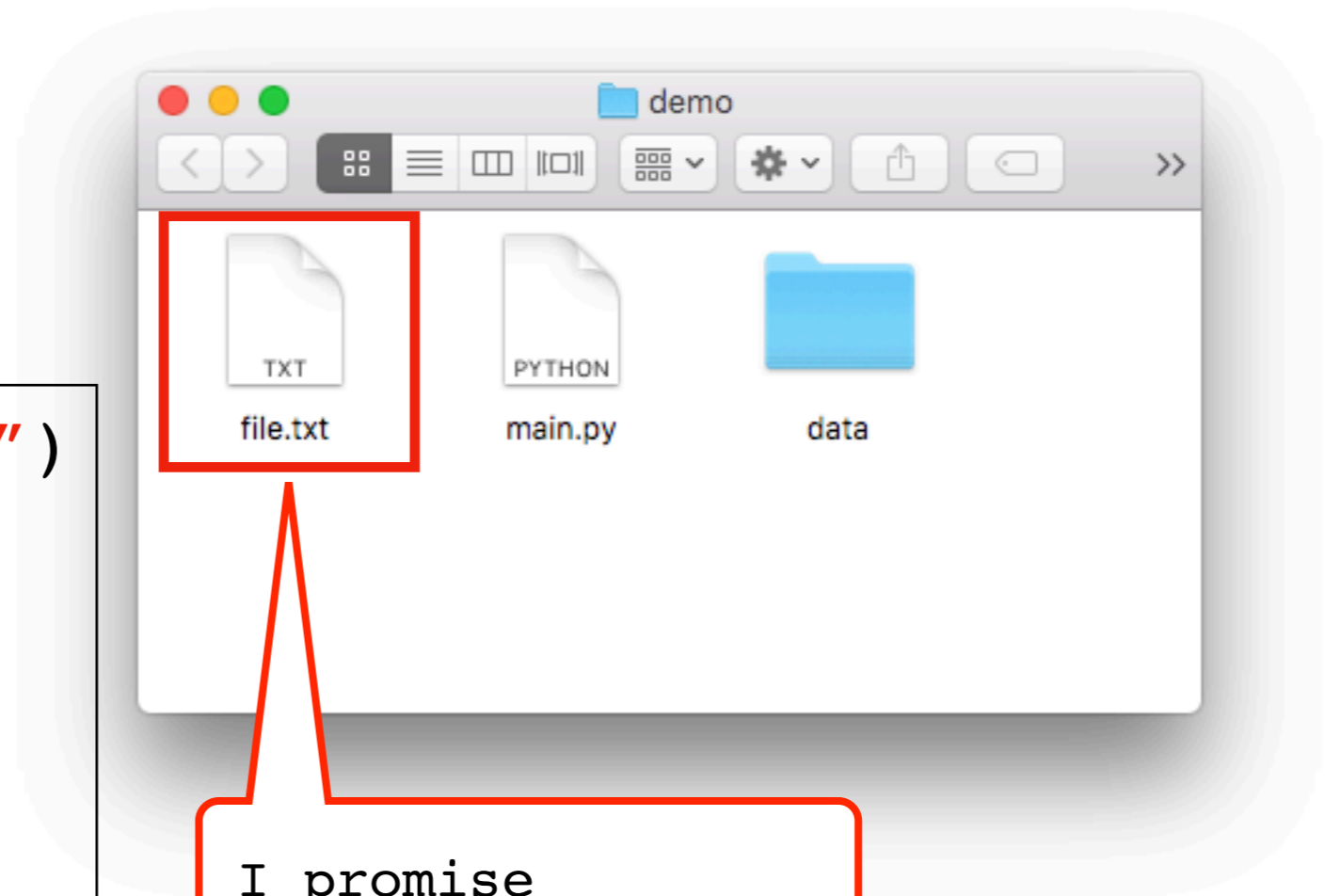
# Write a file

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

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

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

```
f.close()
```



I promise  
to always  
close my files

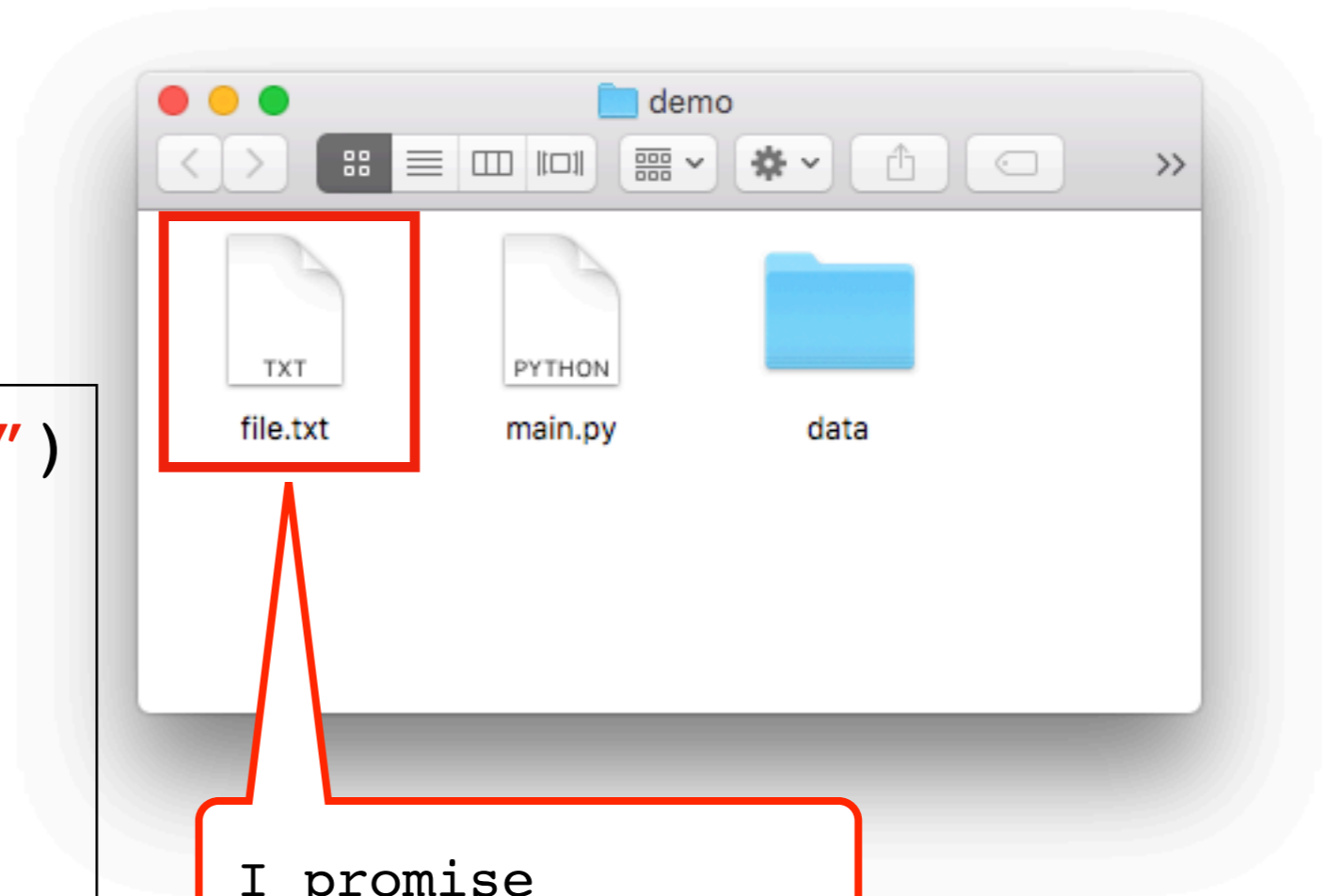
# Write a file

“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

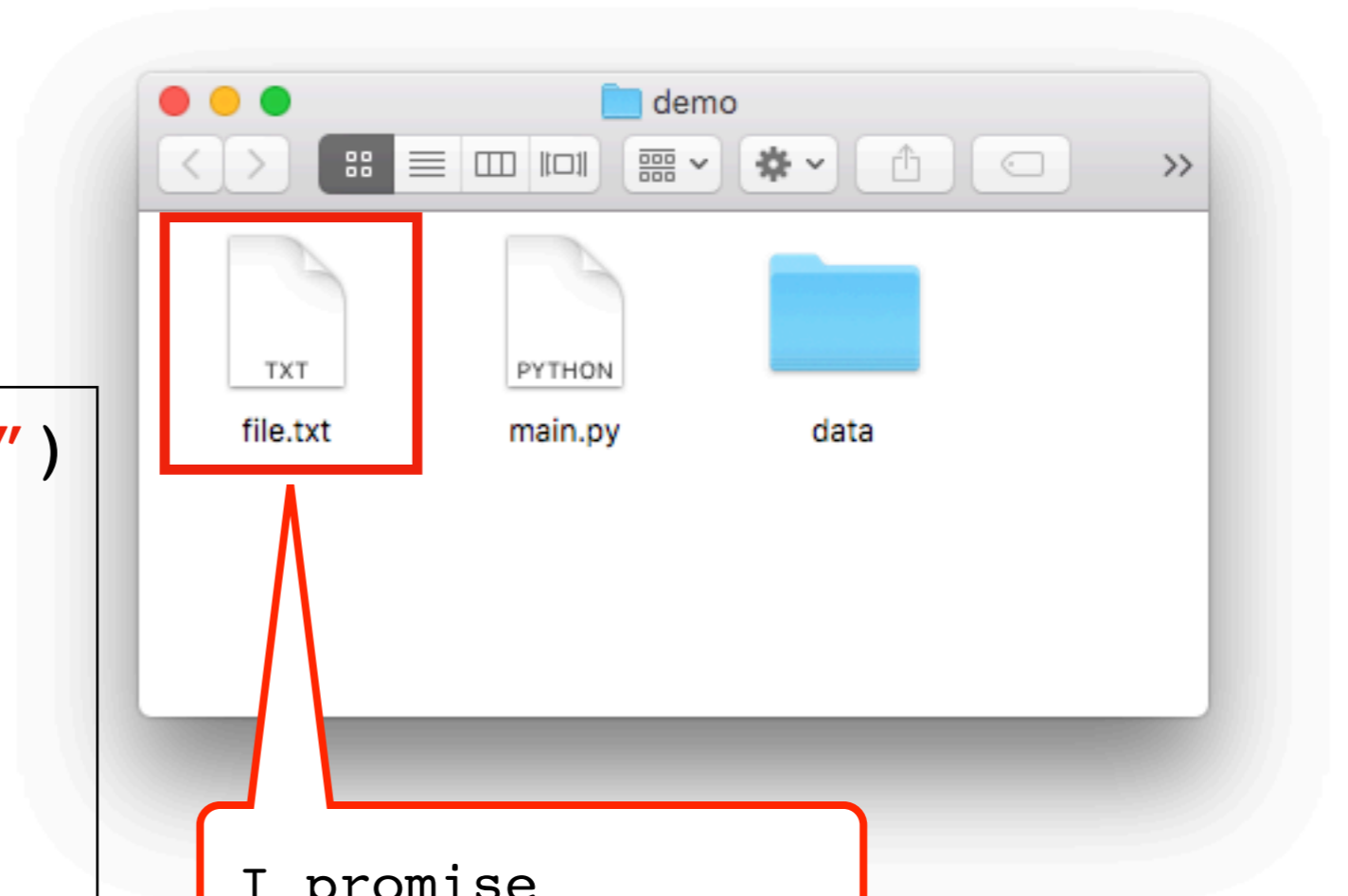
# Write a file

“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()
```



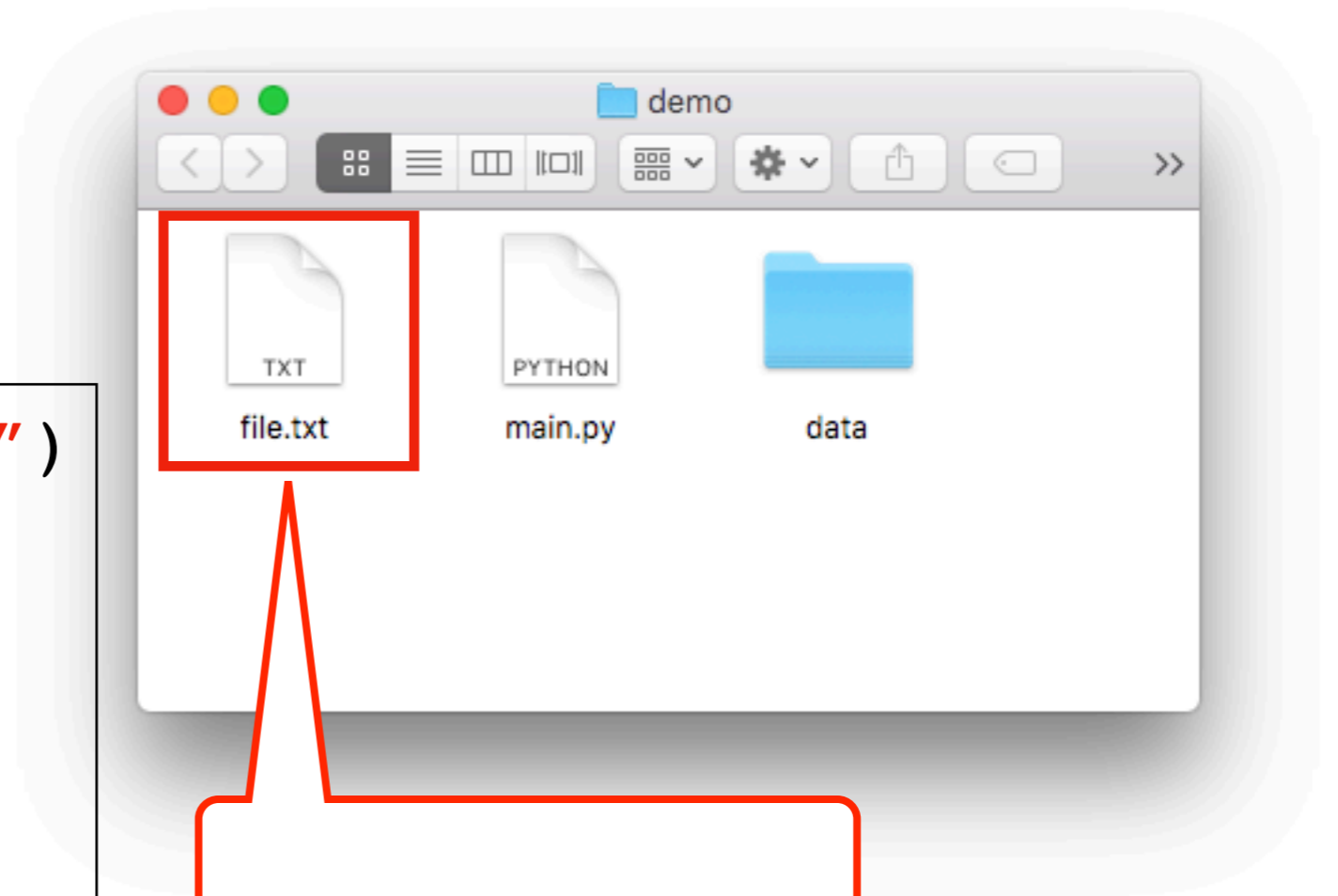
I promise  
to always  
close my files

**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()
```

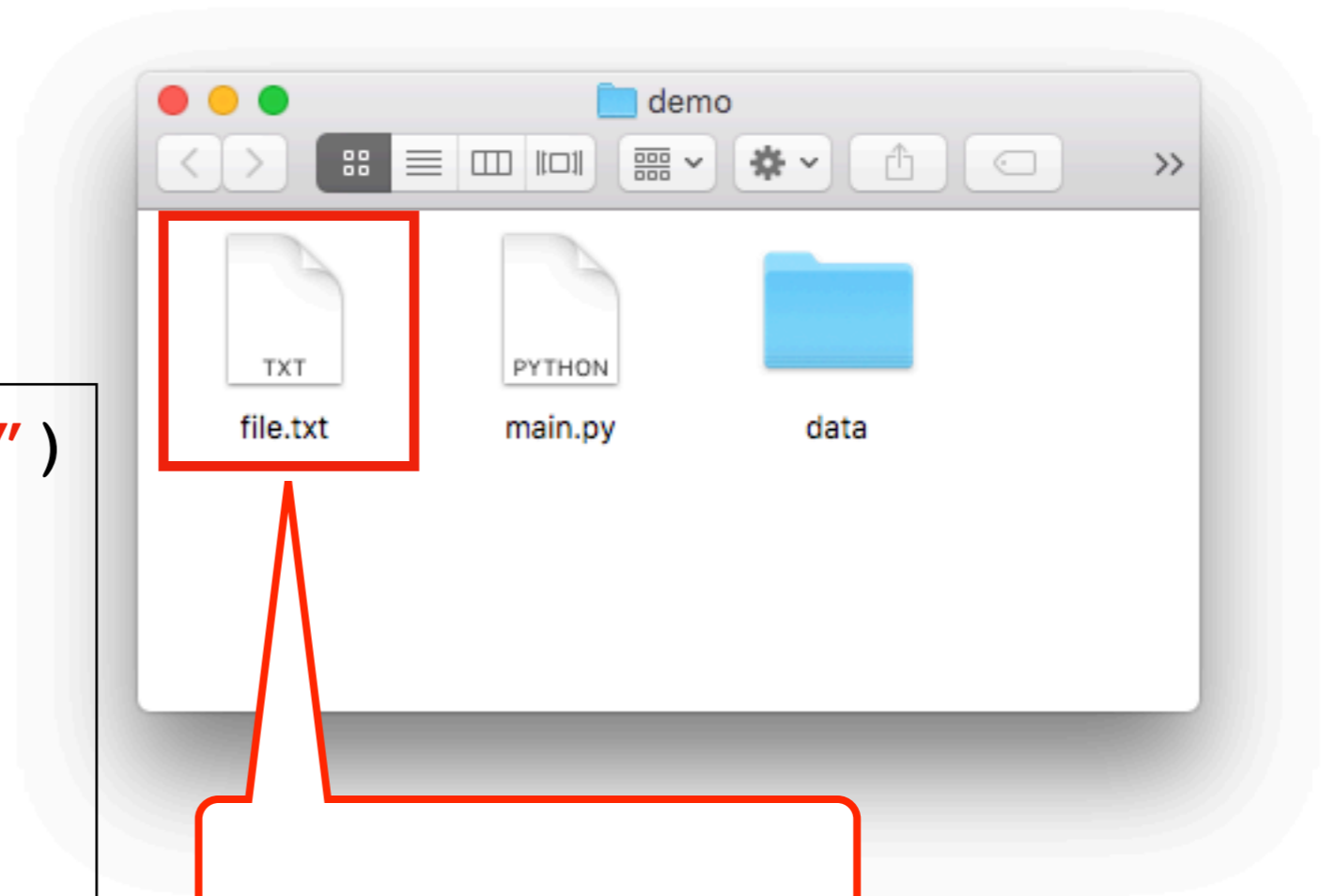


opening with “w” is dangerous. It immediately wipes out your file.

# 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)



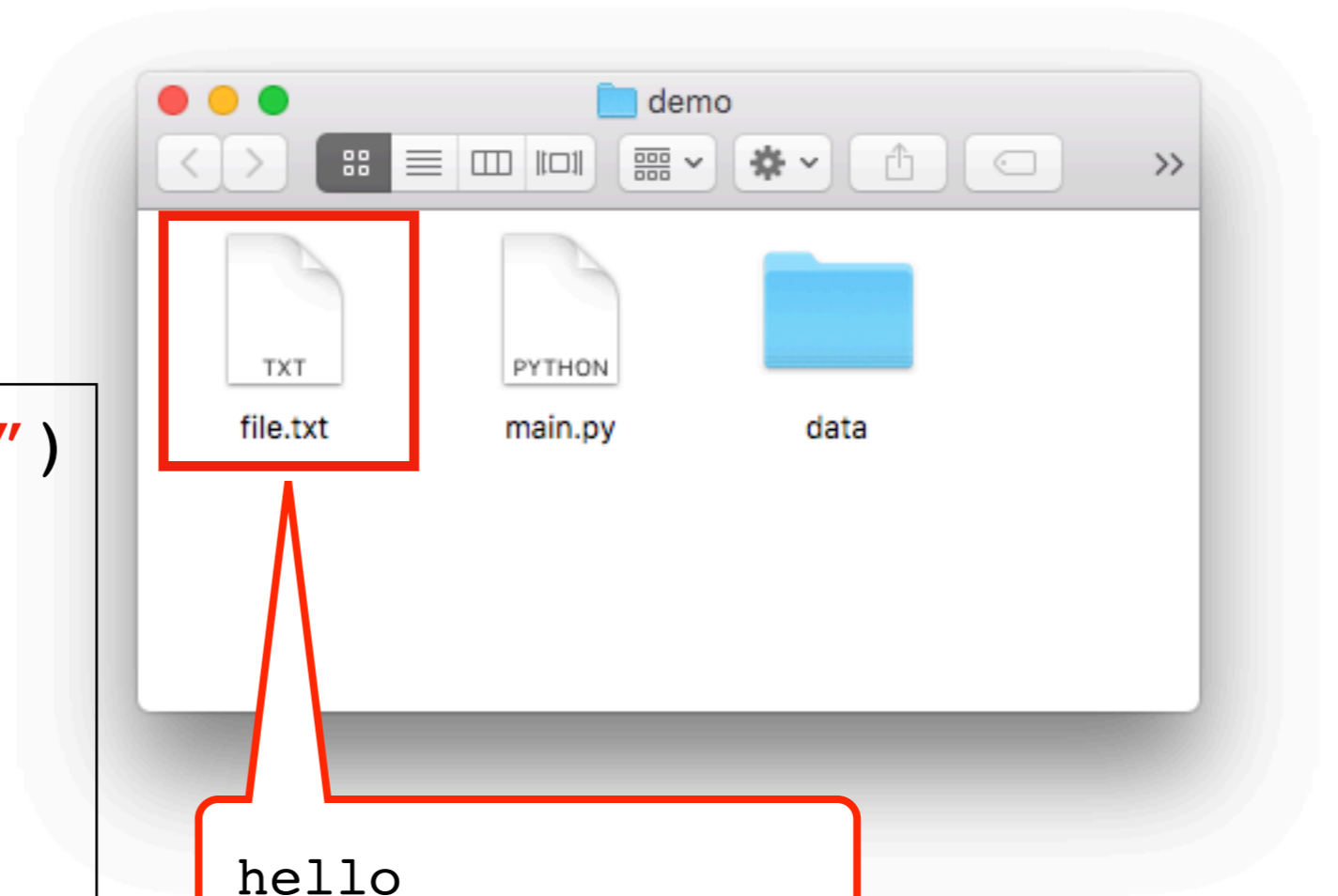
# Write a file

“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

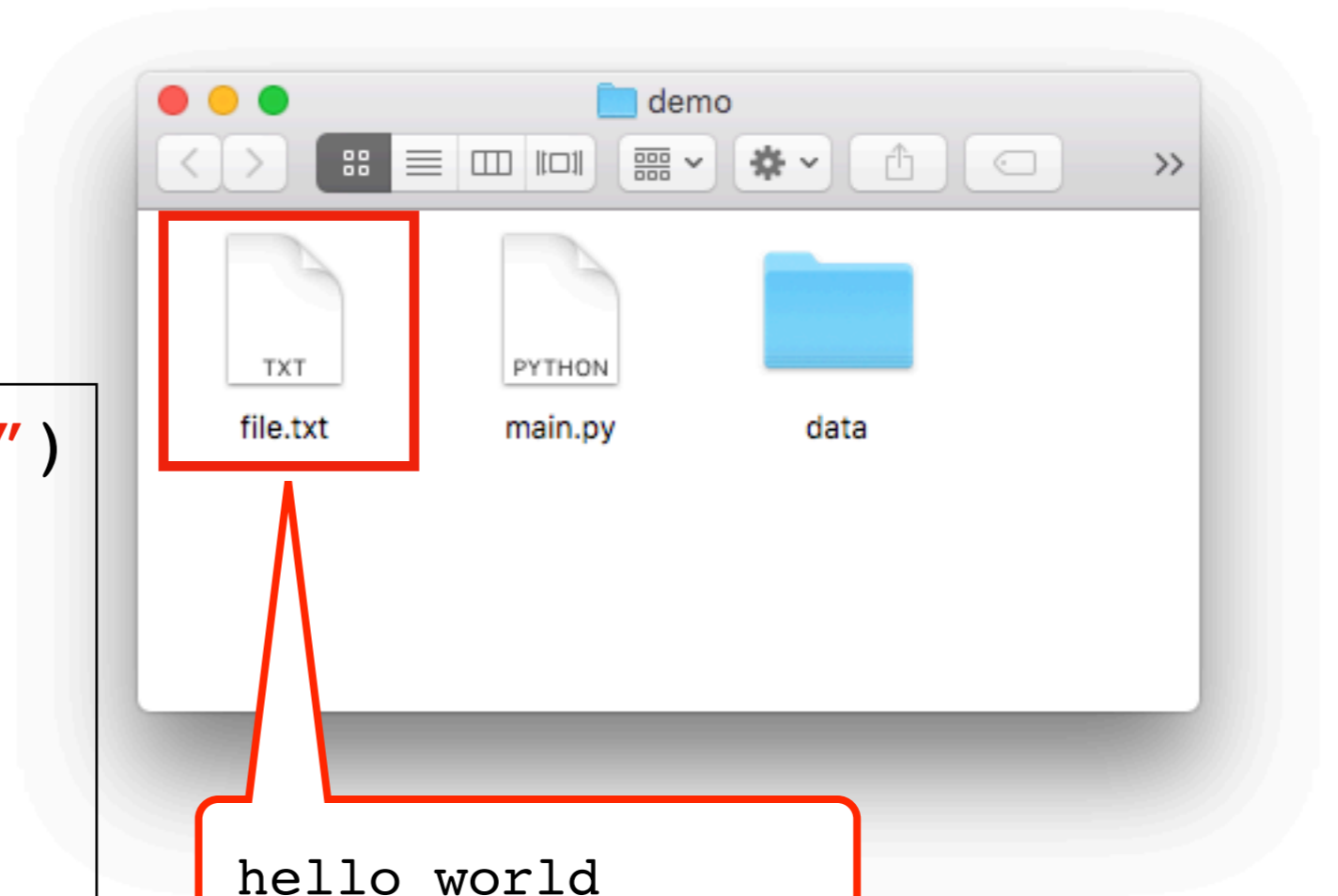
# Write a file

“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

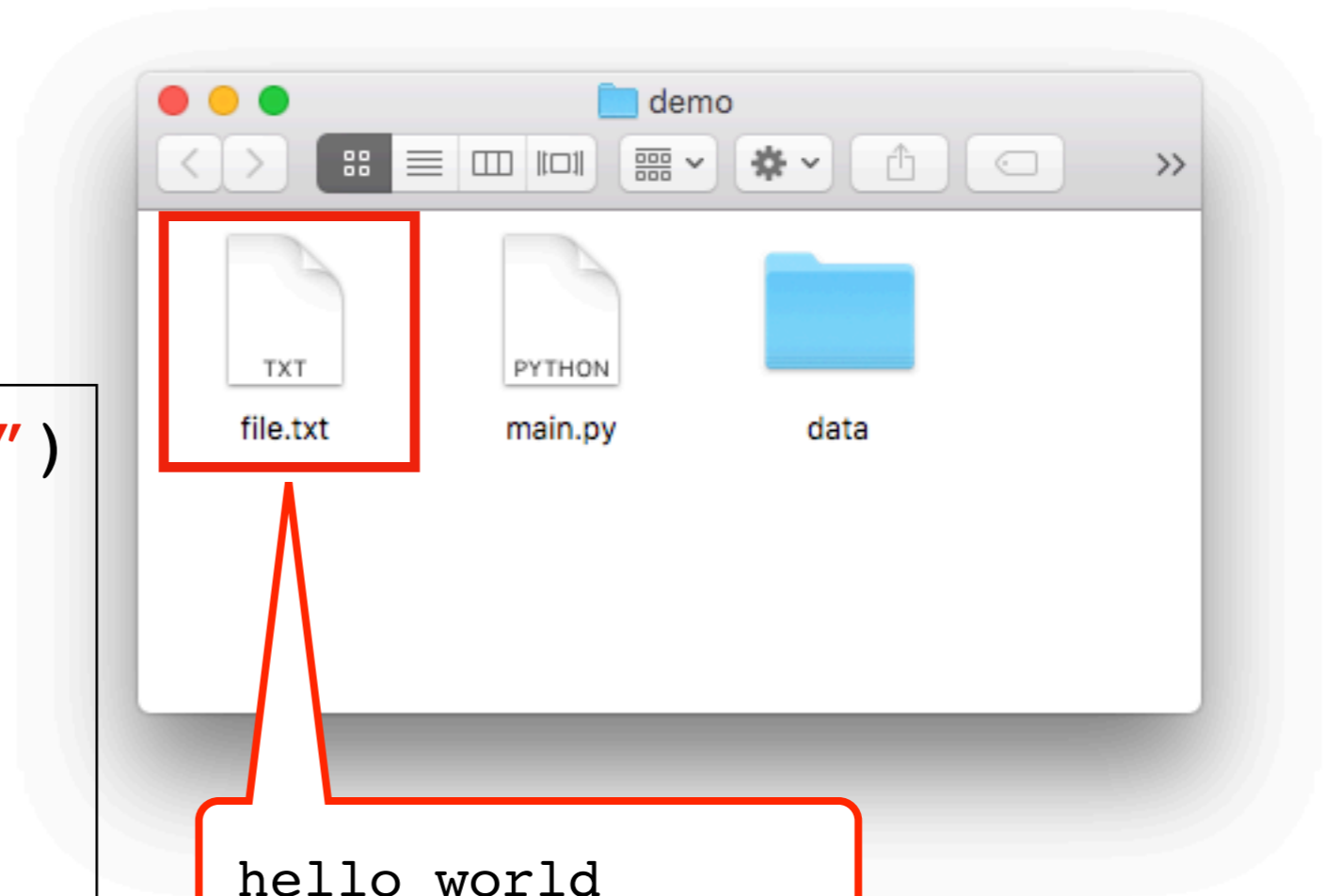
# Write a file

“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()`

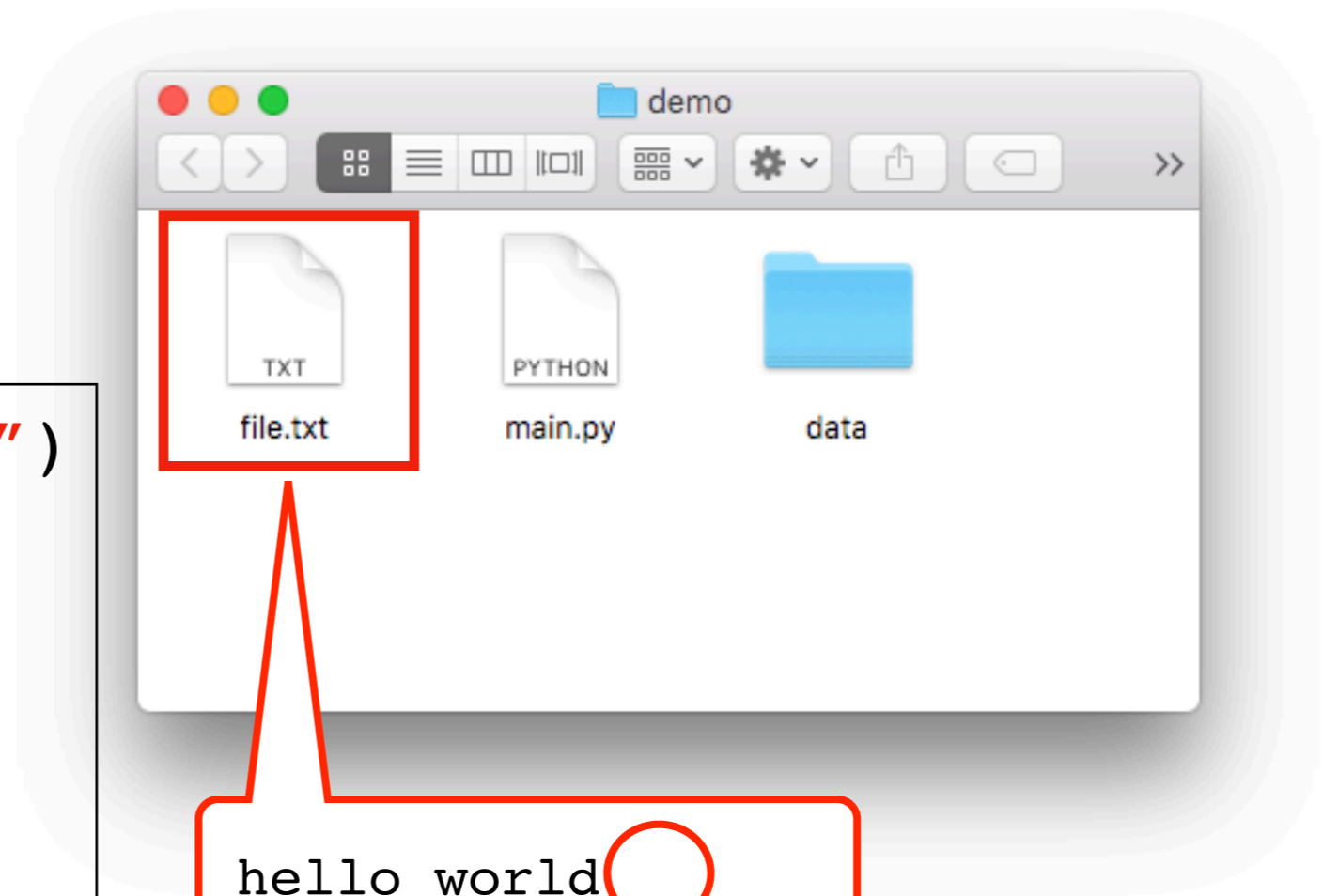


hello world  
!!!!!!

# 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

# 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

# Reading JSON

`fileobj.read()`

- operates on a file object
- return file contents as a string

`json.loads(jstr)`

- takes a string containing JSON
- returns Python structures (lists, dicts, etc)

# Reading JSON

`fileobj.read()`

- operates on a file object
- return file contents as a string

`json.loads(jstr)`

- takes a string containing JSON
- returns Python structures (lists, dicts, etc)



# Reading JSON

`fileobj.read()`

- operates on a file object
- return file contents as a string

`json.loads(jstr)`

- takes a string containing JSON
- returns Python structures (lists, dicts, etc)

data.json

```
{  
  "alice": [1,2,3],  
  "bob": [4,5,6]  
}
```



```
f = open("data.json")  
raw = f.read()
```



# Reading JSON

`fileobj.read()`

- operates on a file object
- return file contents as a string

`json.loads(jstr)`

- takes a string containing JSON
- returns Python structures (lists, dicts, etc)

data.json

```
{  
  "alice": [1,2,3],  
  "bob": [4,5,6]  
}
```

```
f = open("data.json")  
raw = f.read()
```

```
jdata = json.loads(raw)
```

# Reading JSON

`fileobj.read()`

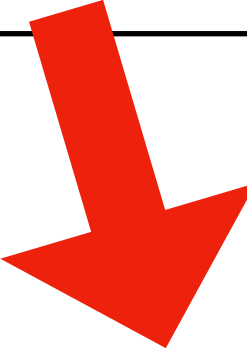
- operates on a file object
- return file contents as a string

`json.loads(jstr)`


- takes a string containing JSON
- returns Python structures (lists, dicts, etc)

data.json

```
{  
  "alice": [1,2,3],  
  "bob": [4,5,6]  
}
```



```
f = open("data.json")  
raw = f.read()
```



```
jdata = json.loads(raw)
```

```
for name in jdata:  
    print(name,  
          sum(jdata[name]))
```

# Reading JSON

`fileobj.read()`

- operates on a file object
- return file contents as a string

`json.loads(jstr)`

- takes a string containing JSON
- returns Python structures (lists, dicts, etc)

this pattern is so common  
there's a shortcut for it

data.json

```
{  
  "alice": [1,2,3],  
  "bob": [4,5,6]  
}
```

```
f = open("data.json")  
raw = f.read()
```

```
jdata = json.loads(raw)
```

```
for name in jdata:  
    print(name,  
          sum(jdata[name]))
```

# Reading JSON

`fileobj.read()`

- operates on a file object
- return file contents as a string

`json.loads(jstr)`

- takes a string containing JSON
- returns Python structures (lists, dicts, etc)

this pattern is so common  
there's a shortcut for it

data.json

```
{  
  "alice": [1,2,3],  
  "bob": [4,5,6]  
}
```

```
f = open("data.json")  
raw = f.read()
```

```
jdata = json.loads(raw)  
jdata = json.load(f)
```

```
for name in jdata:  
    print(name,  
          sum(jdata[name]))
```

# Writing JSON

```
players = [  
    {"name": "alice", "score": 15},  
    {"name": "bob", "score": 10}  
    {"name": "cindy", "score": 5}  
]
```

```
jstr = json.dumps(players)
```

```
f = open("players.json", "w")  
f.write(jstr)  
f.close()
```

# Writing JSON

```
players = [  
    {"name": "alice", "score": 15},  
    {"name": "bob", "score": 10}  
    {"name": "cindy", "score": 5}  
]
```

```
jstr = json.dumps(players)
```

```
f = open("players.json", "w")
```

```
f.write(jstr)
```

```
f.close()
```

# Writing JSON

```
players = [  
    {"name": "alice", "score": 15},  
    {"name": "bob", "score": 10},  
    {"name": "cindy", "score": 5}  
]
```

```
jstr = json.dumps(players)
```

```
f = open("players.json", "w")
```

```
f.write(jstr)
```

```
json.dump(players, f)
```

```
f.close()
```

# 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



# Reading CSVs

```
def csv_reader(fileobj):  
    for line in fileobj:  
        row = line.split(',')  
        yield row
```

# Reading CSVs

```
def csv_reader(fileobj):  
    for line in fileobj:  
        row = line.split(',')  
        yield row  
  
f = open('data.csv')  
reader = csv_reader(f)  
  
for row in reader:  
    print(row)  
  
f.close()
```

# Reading CSVs

```
def csv_reader(fileobj):  
    for line in fileobj:  
        row = line.split(',')  
        yield row
```

```
f = open('data.csv')  
reader = csv_reader(f)
```

```
for row in reader:  
    print(row)
```

```
f.close()
```



data.csv

```
A,B,C  
1,2,3  
4,5,6
```



```
['A', 'B', 'C\n']  
['1', '2', '3\n']  
['4', '5', '6\n']
```

# Reading CSVs

```
def csv_reader(fileobj):  
    for line in fileobj:  
        row = line.split(',')  
        yield row
```

```
f = open('data.csv')  
reader = csv_reader(f)
```

```
for row in reader:  
    print(row)
```

```
f.close()
```

data.csv

```
title,actors  
movie 1,"A,B,C"  
movie 2,"D,E,F"
```



**Oops! This code doesn't  
work with commas within fields**



```
['title', 'actors\n']  
['movie 1', '"A', 'B', 'C"\n']  
['movie 2', '"D', 'E', 'F"']
```

# Reading CSVs

```
def csv_reader(fileobj):  
    for line in fileobj:  
        row = line.split(',')  
        yield row
```

```
import csv  
f = open('data.csv')  
reader = csv.reader(f)
```

```
for row in reader:  
    print(row)
```

```
f.close()
```

data.csv

```
title,actors  
movie 1,"A,B,C"  
movie 2,"D,E,F"
```



use `csv.reader` to handle  
such special cases



```
['title', 'actors']  
['movie 1', 'A,B,C']  
['movie 2', 'D,E,F']
```

# 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

# OS Module (Operating System)

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

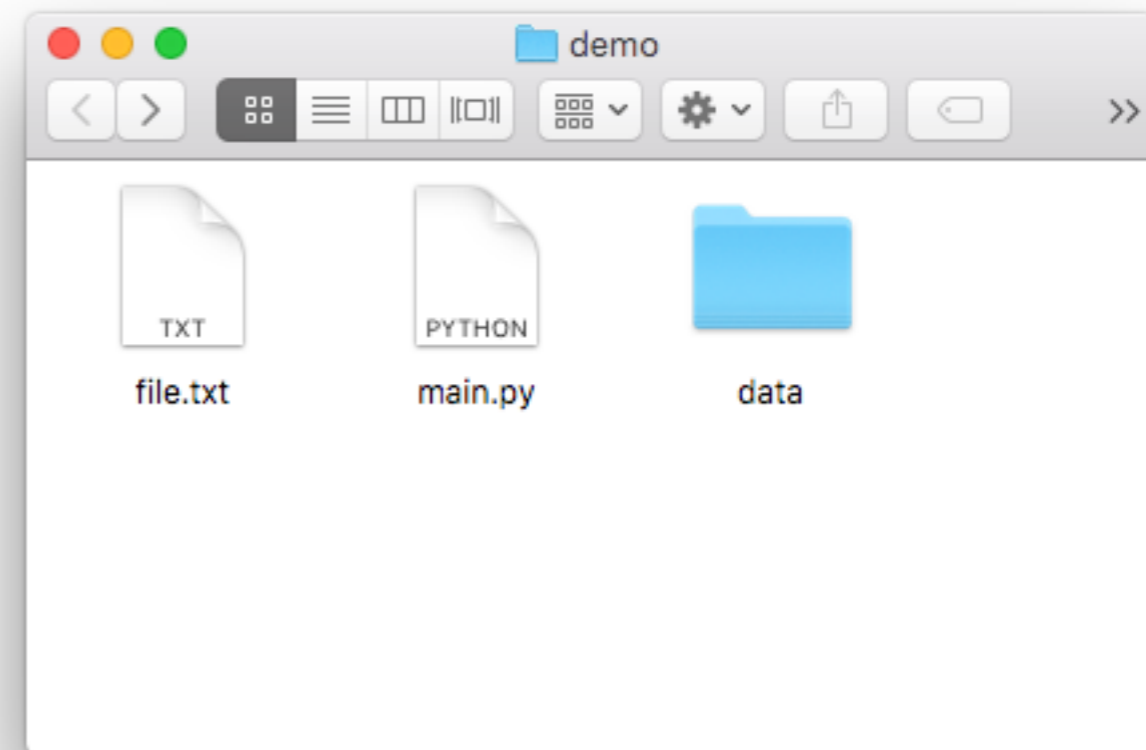
- `os.listdir`
- `os.mkdir`
- `os.path.exists`
- `os.path.isfile`
- `os.path.isdir`
- `os.path.join`

# 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"]
```



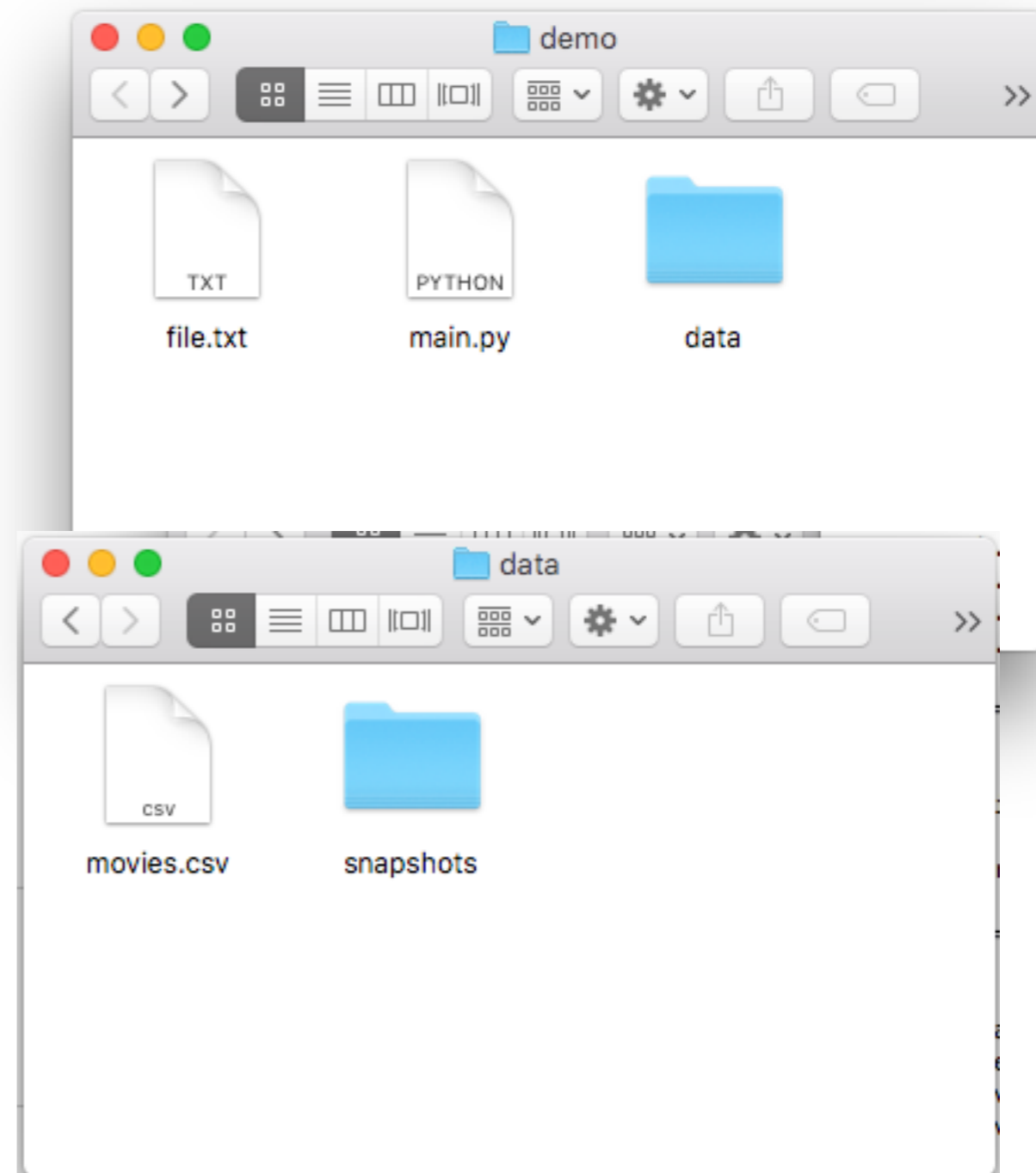


# OS Module (Operating System)

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", "data"]
```

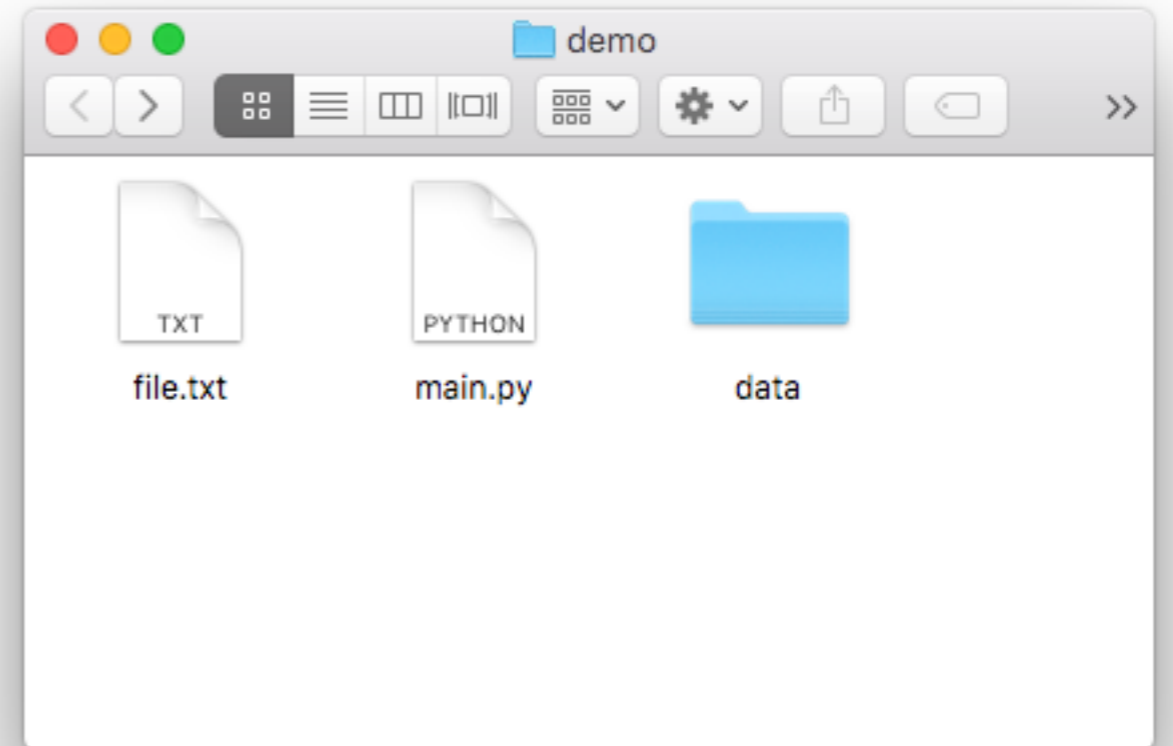


# OS Module (Operating System)

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.mkdir("test")
```

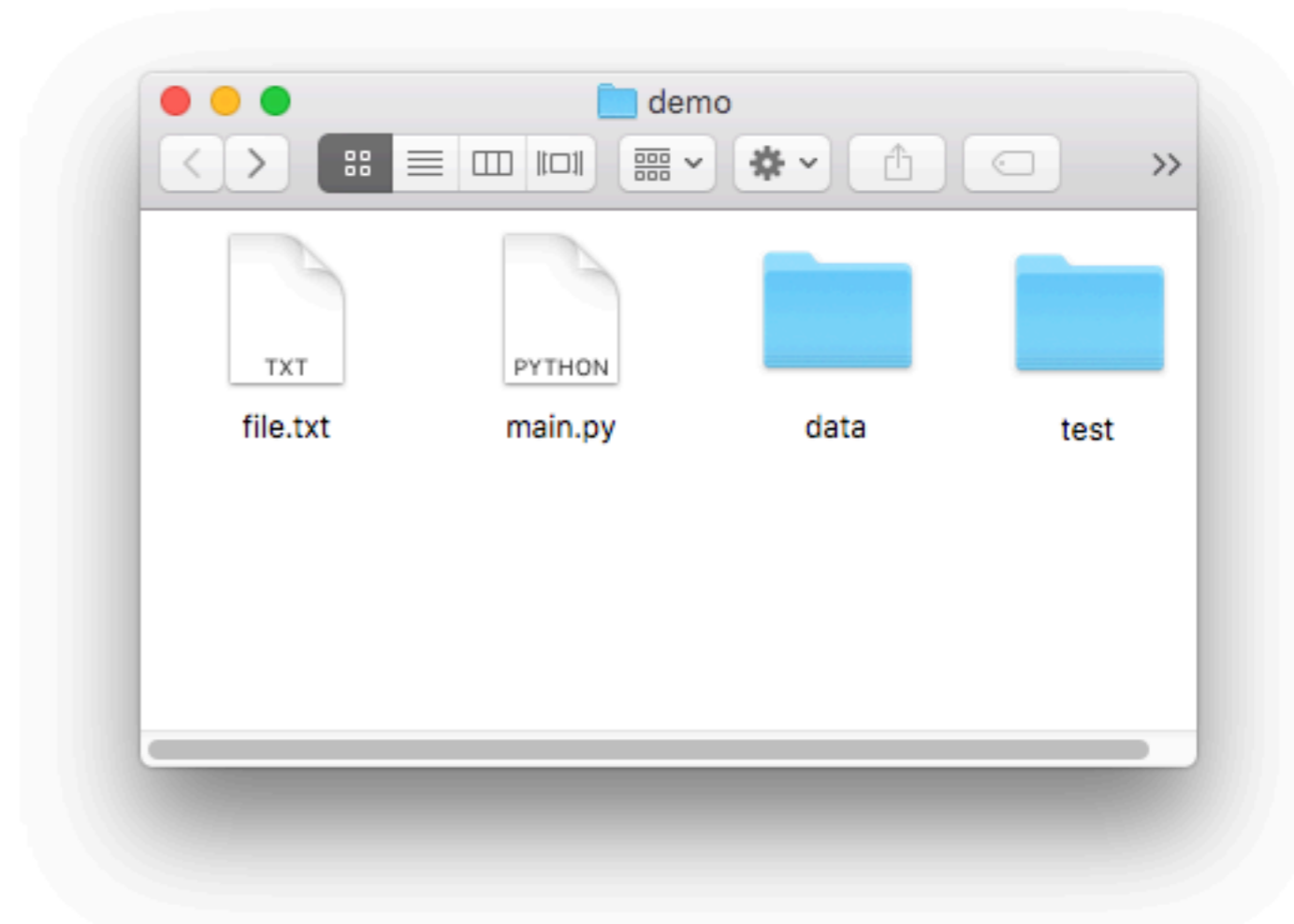


# OS Module (Operating System)

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")
```

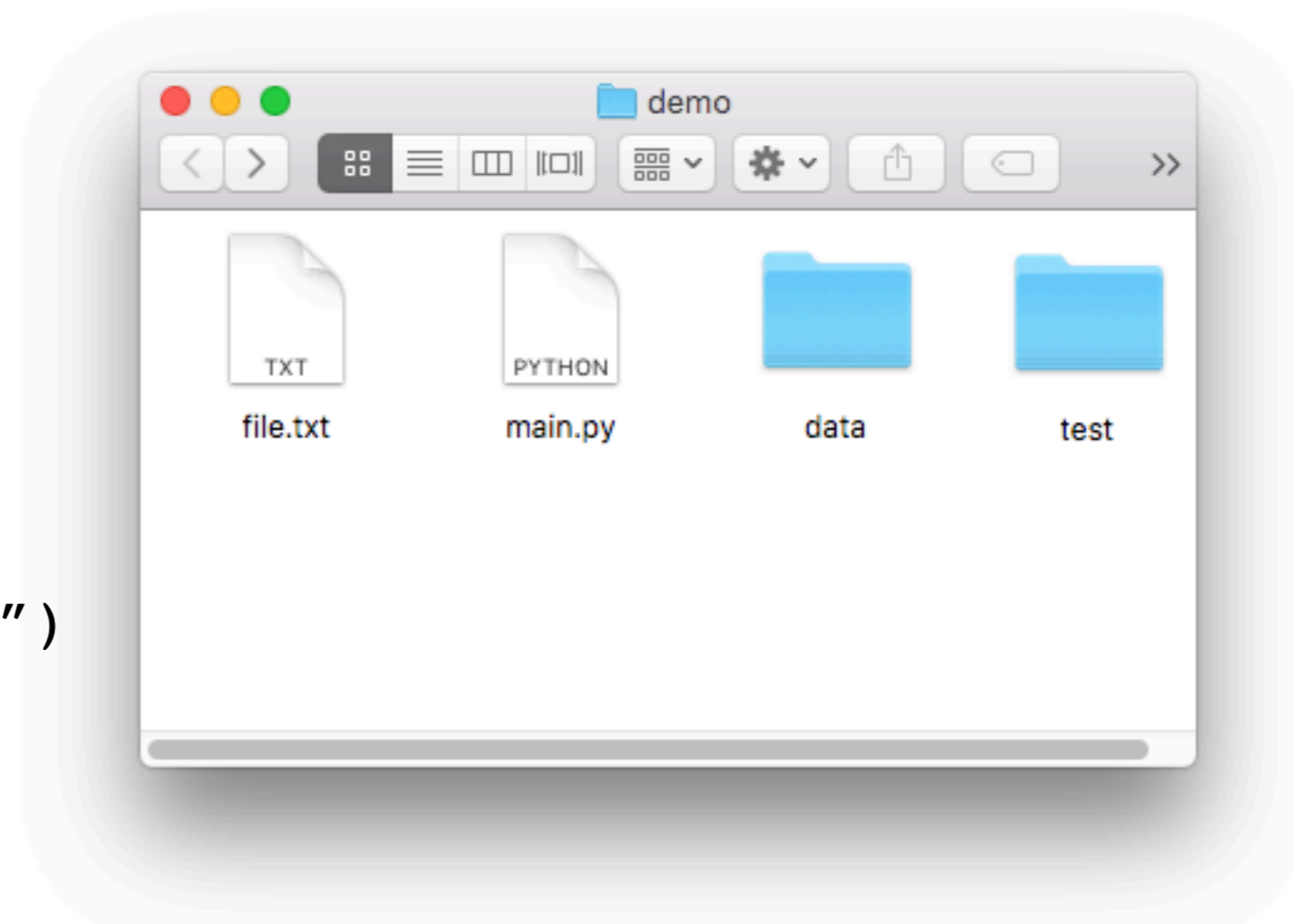


# OS Module (Operating System)

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
```

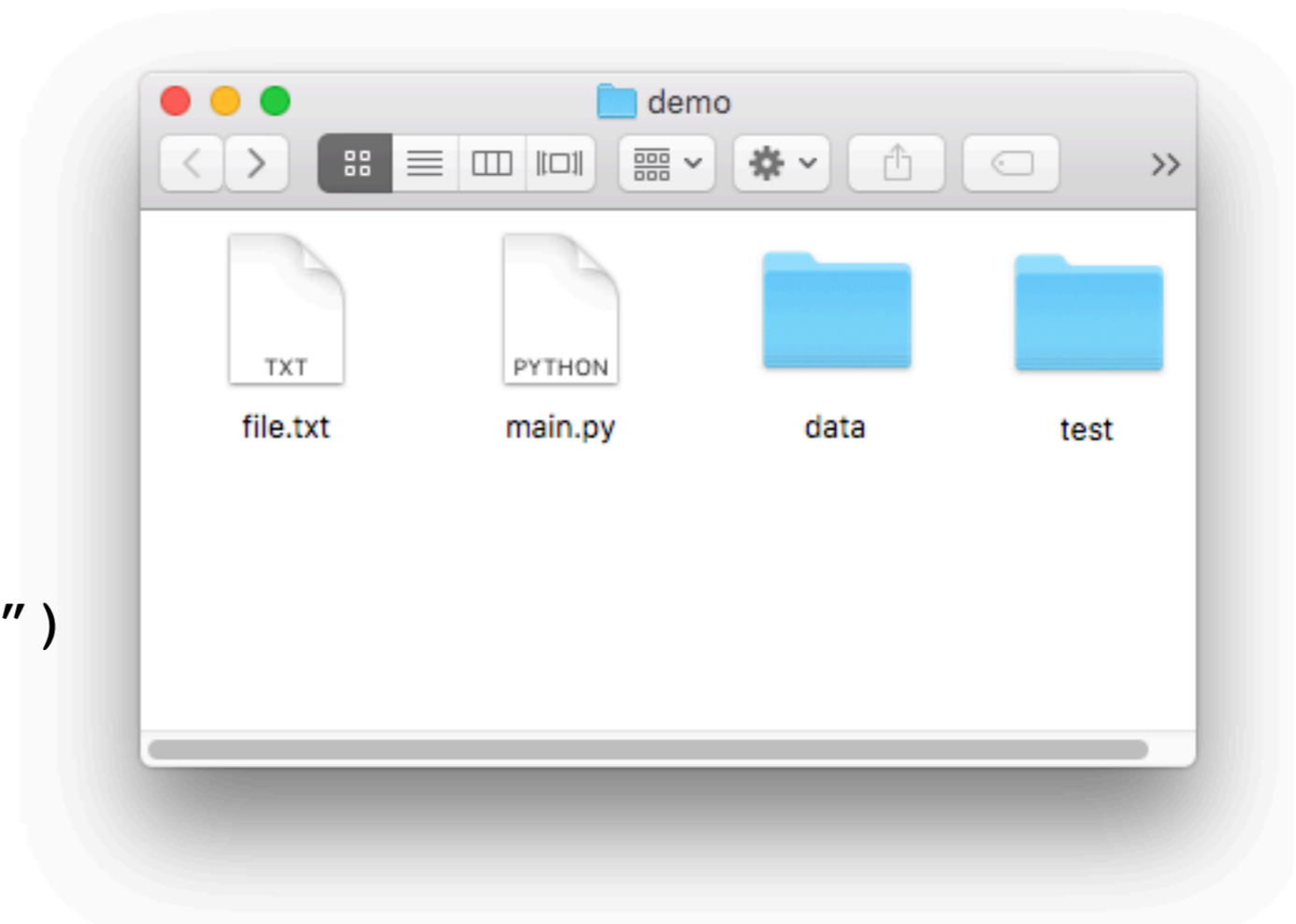


# OS Module (Operating System)

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
```

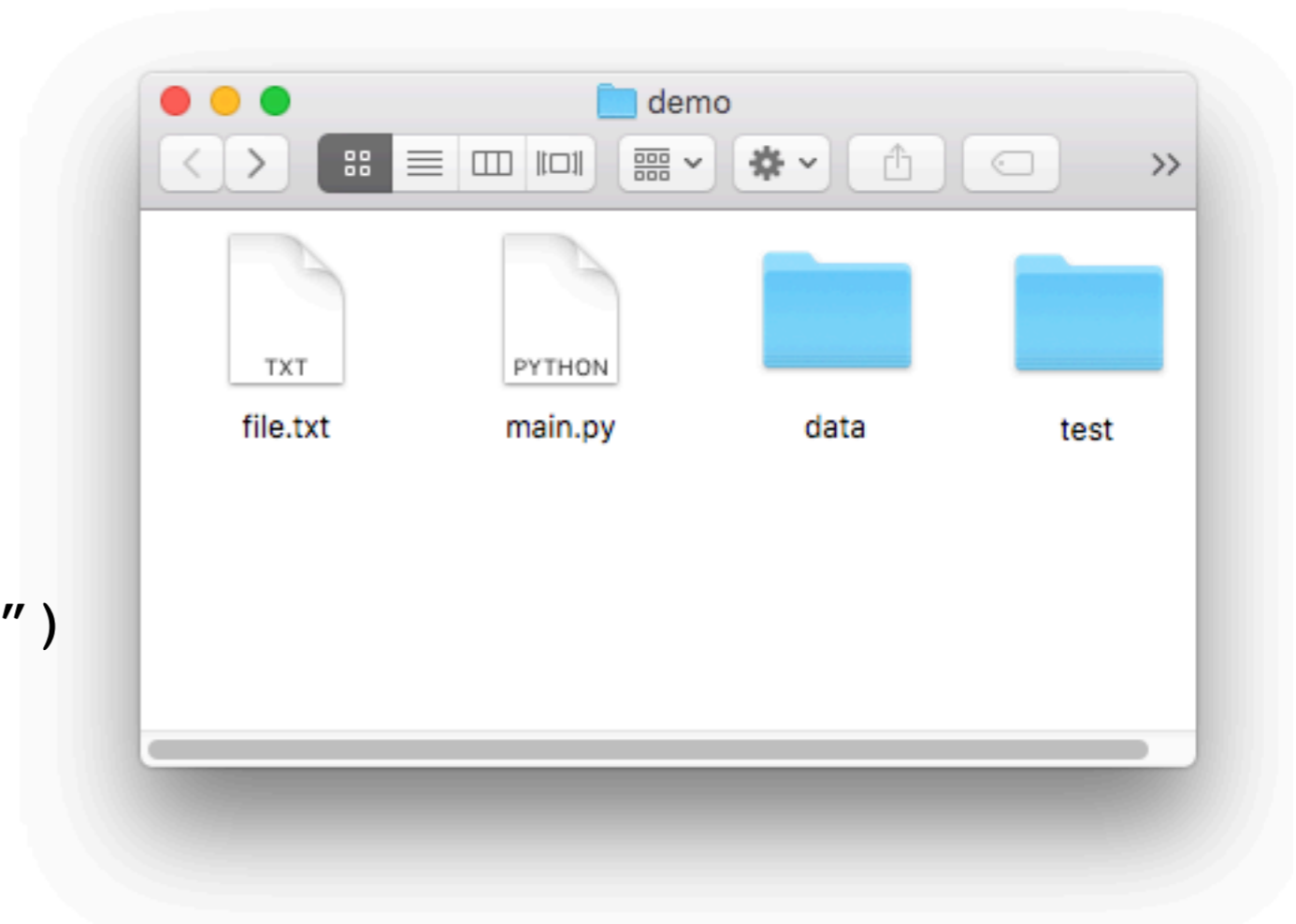


# OS Module (Operating System)

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
```

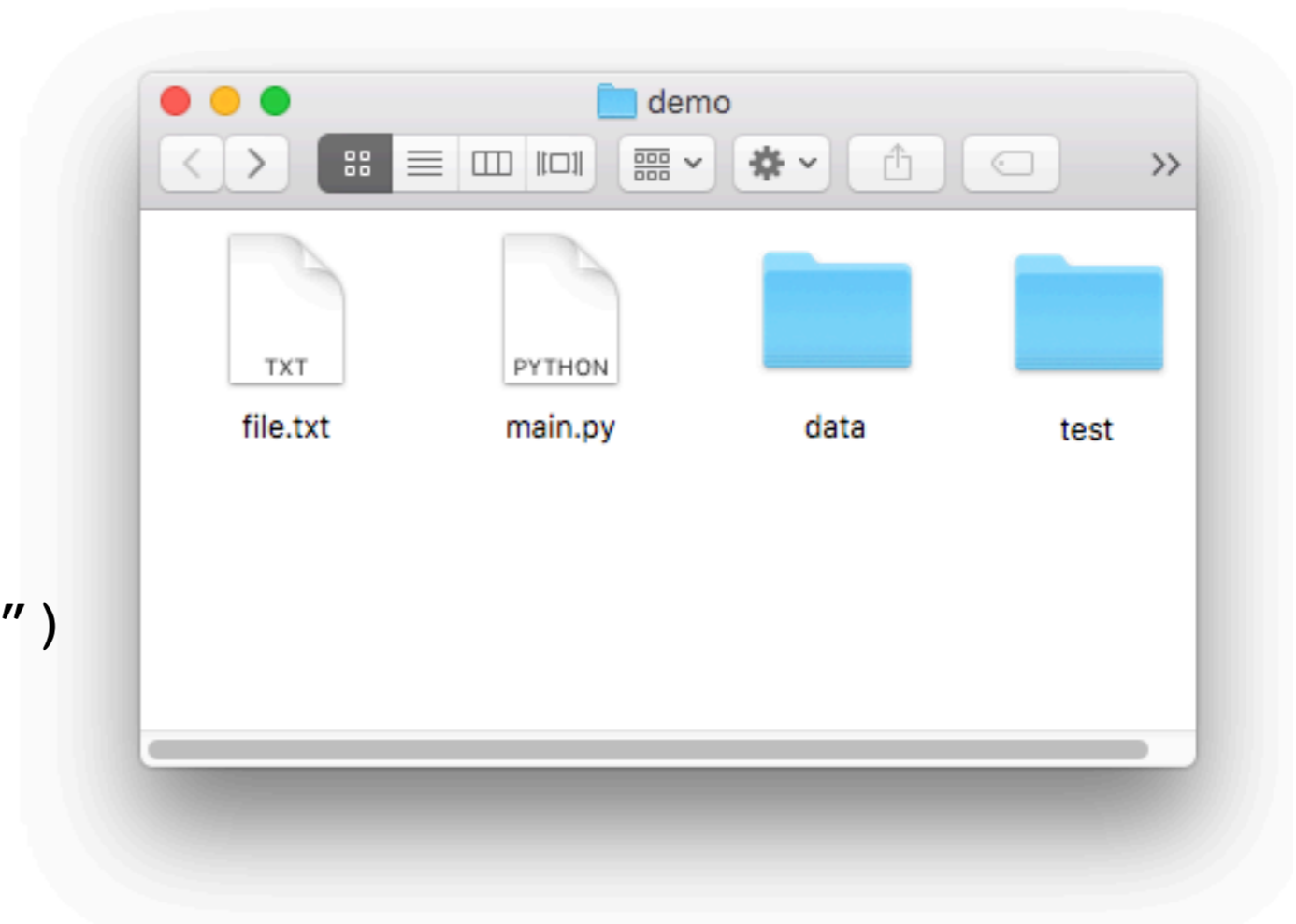


# OS Module (Operating System)

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
```

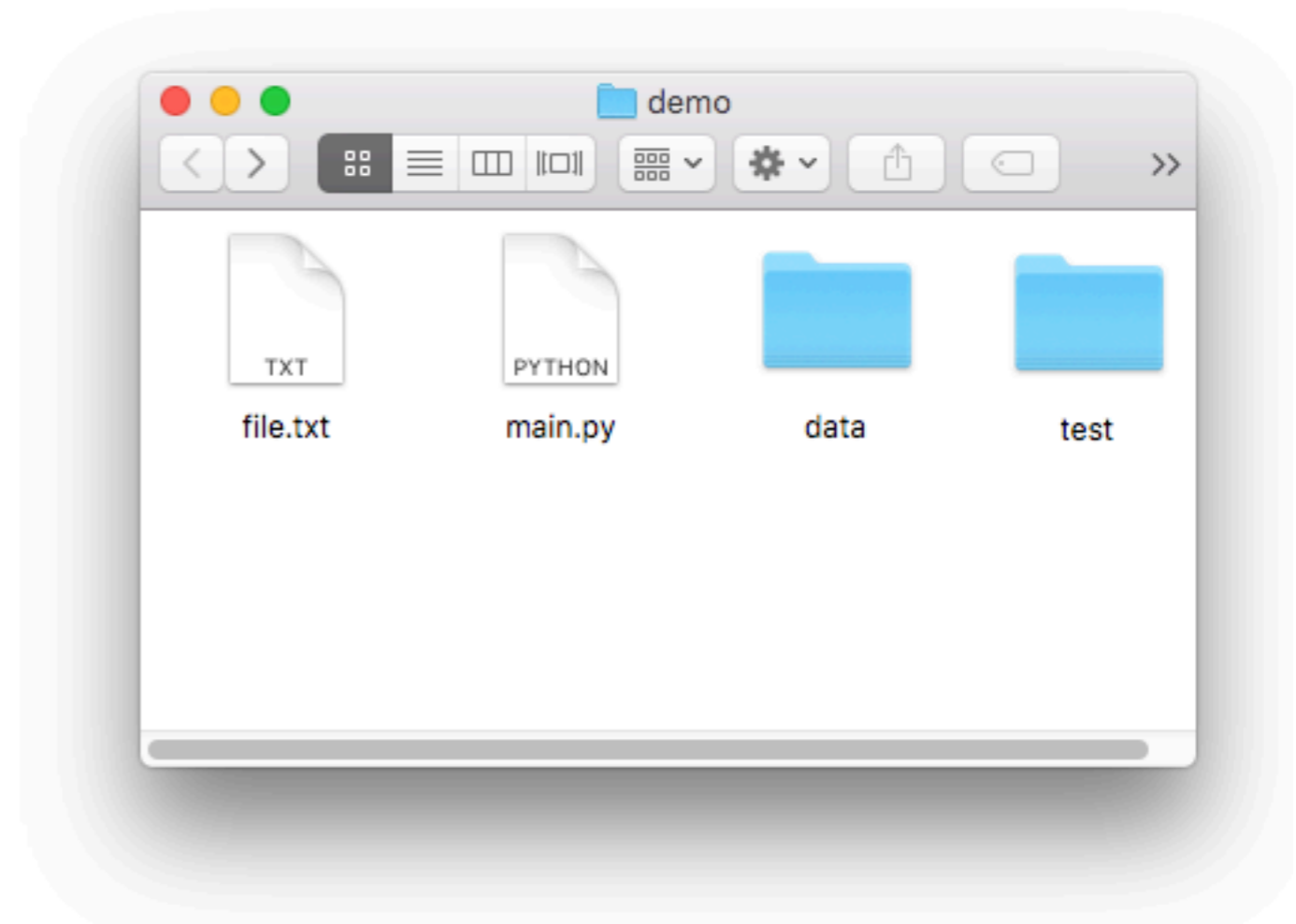


# OS Module (Operating System)

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
```



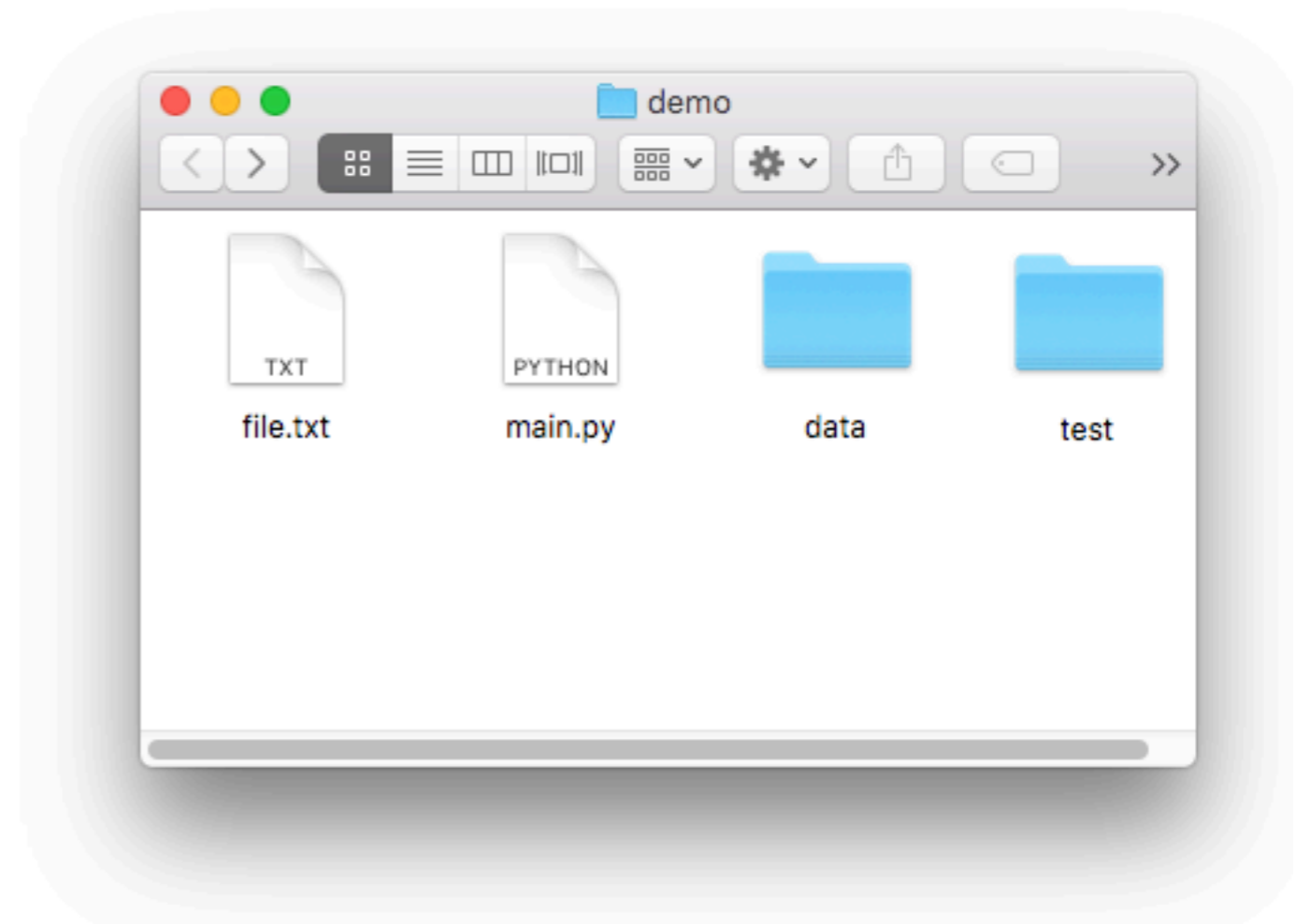


# OS Module (Operating System)

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

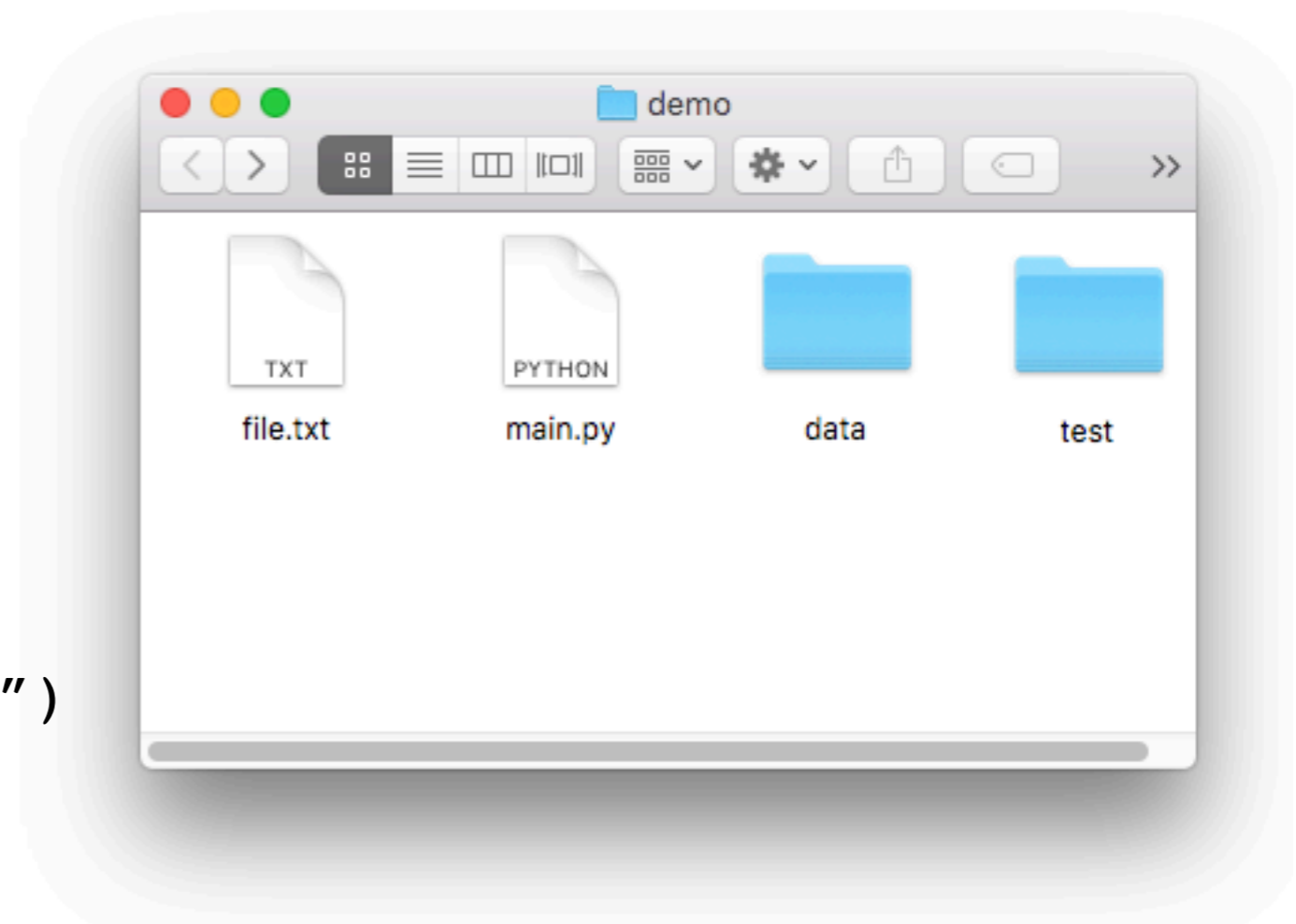
- `os.listdir`
- `os.mkdir`
- `os.path.exists`
- `os.path.isfile`
- `os.path.isdir`
- **`os.path.join`**

```
>>> 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

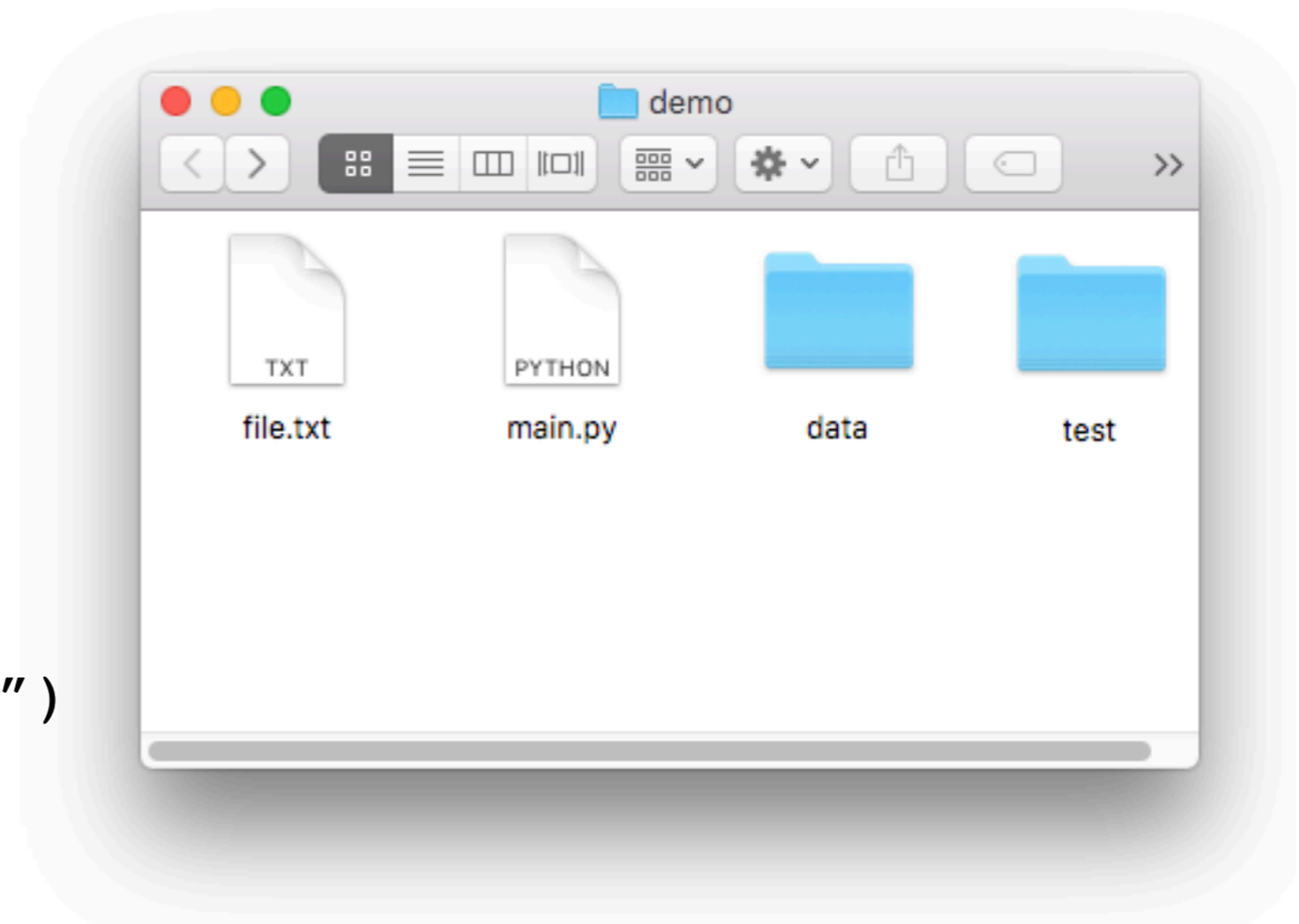
- `os.listdir`
- `os.mkdir`
- `os.path.exists`
- `os.path.isfile`
- `os.path.isdir`
- **`os.path.join`**

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

```
data\movies.csv
```



on Windows



# 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

# Exceptions

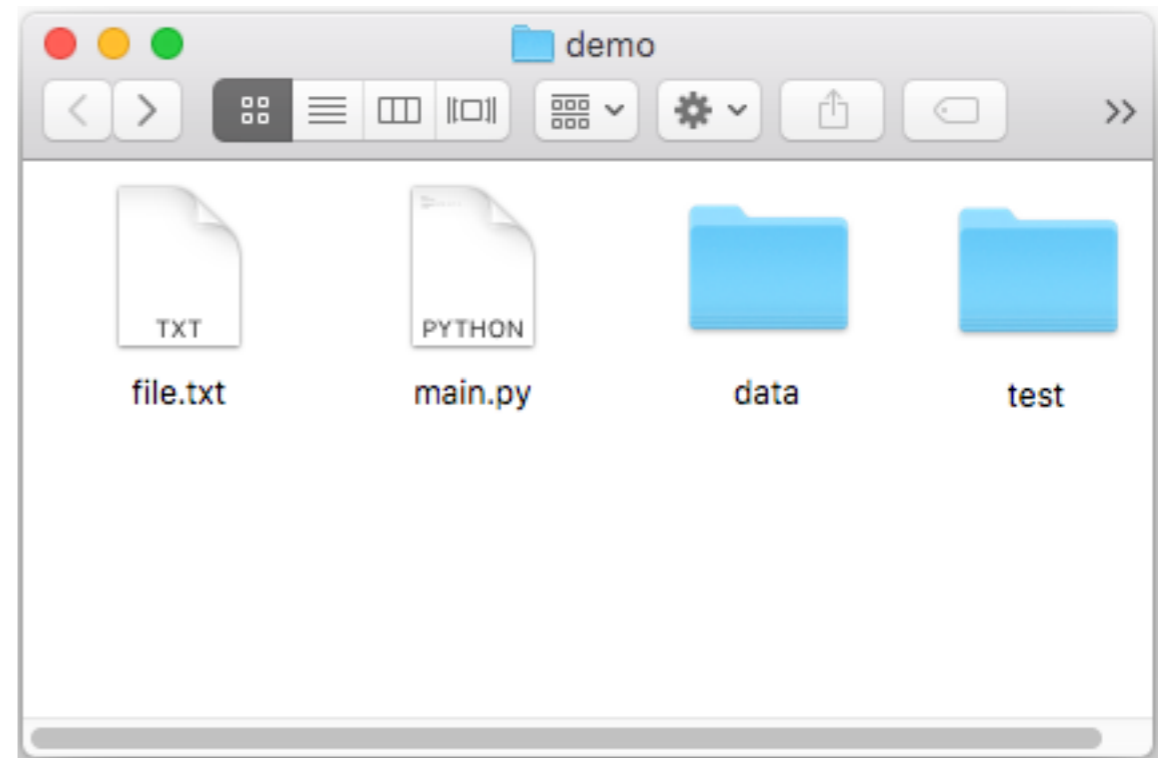
Working with files leads to many exceptions

- 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



```
import os
```

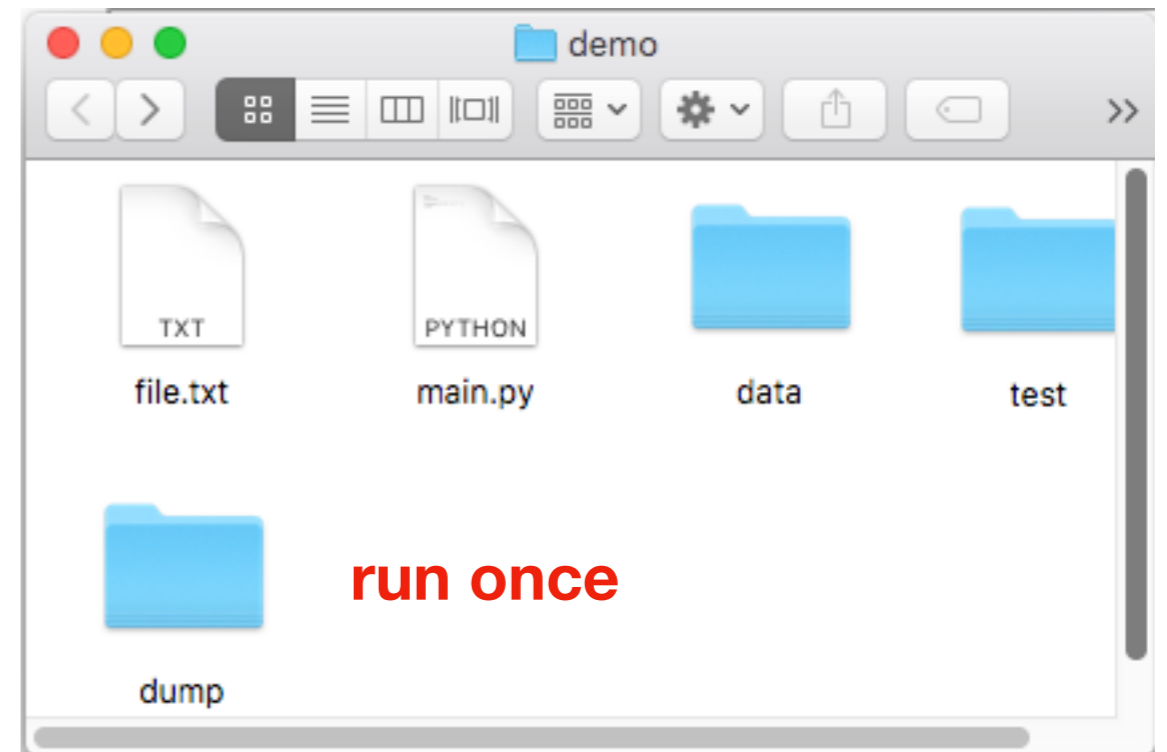
```
os.mkdir('dump')
```

```
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
```

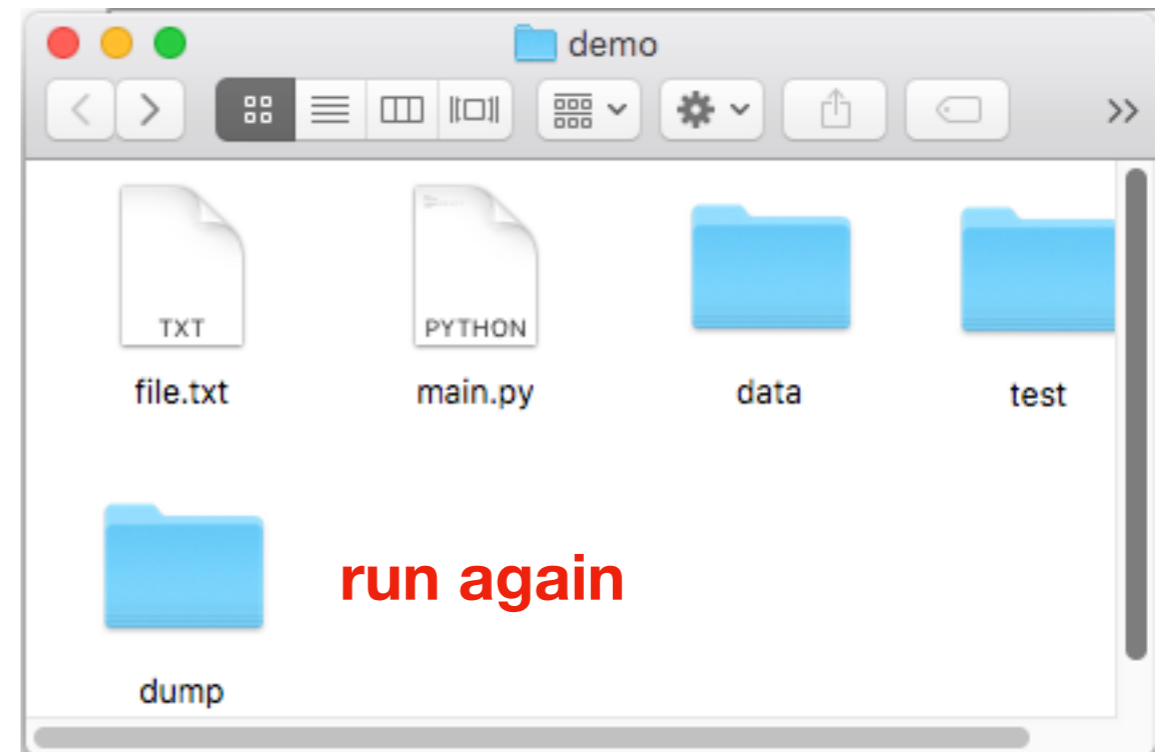
```
os.mkdir('dump')
```

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

# 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')
```

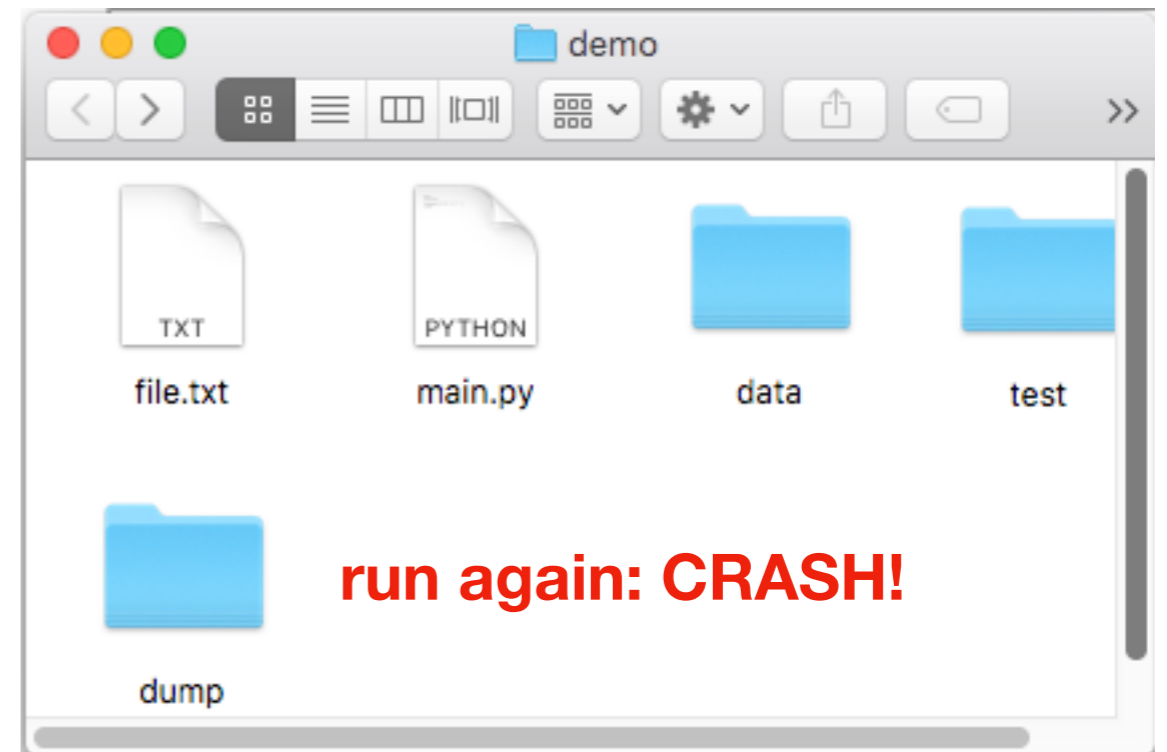
```
f = open(os.path.join('dump', 'out.txt'), 'w')  
f.write('hi')  
f.close()
```



# 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>
```

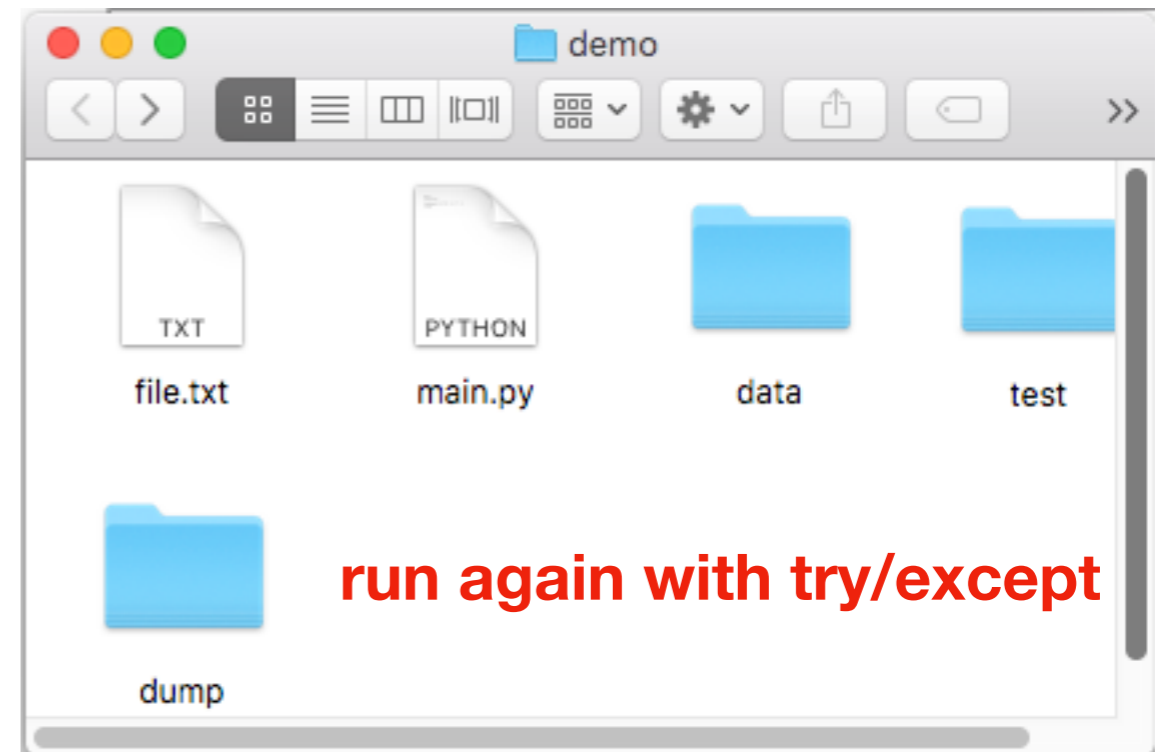
```
os.mkdir('dump')
```

```
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

## File formats

- JSON
- CSV

## OS module

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

## File exceptions

## Encodings

# Encodings

When you read/write a file,  
Python must decide what bits to use for each character.

This translation is called encoding.

# Encodings

When you read/write a file,  
Python must decide what bits to use for each character.

This translation is called encoding.

**encoding A**

<b>Character</b>	<b>Bits</b>
A	1100
B	0011
C	1111
D	0000

**“CAB”**  **encoding A** **1111 1100 0011**

# Encodings

When you read/write a file,  
Python must decide what bits to use for each character.

This translation is called encoding.

**encoding A**

<b>Character</b>	<b>Bits</b>
A	1100
B	0011
C	1111
D	0000



# Encodings

When you read/write a file,  
Python must decide what bits to use for each character.

This translation is called encoding.

encoding A

Character	Bits
A	1100
B	0011
C	1111
D	0000

encoding B

Character	Bits
A	1100
B	1111
C	1001
D	0011



*what if decode with a different encoding?*

# Encodings

When you read/write a file,  
Python must decide what bits to use for each character.

This translation is called encoding.

**encoding A**

<b>Character</b>	<b>Bits</b>
A	1100
B	0011
C	1111
D	0000

**encoding B**

<b>Character</b>	<b>Bits</b>
A	1100
B	1111
C	1001
D	0011





# Encodings

When you read/write a file,  
Python must decide what bits to use for each character.

This translation is called encoding.

Python uses different **default encodings** on Mac and Windows.

Problematic for special characters.

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

# Encodings

When you read/write a file,  
Python must decide what bits to use for each character.

This translation is called encoding.

Python uses different **default encodings** on Mac and Windows.

Problematic for special characters.

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

override default

# 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

# 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 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