[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

built-in open function

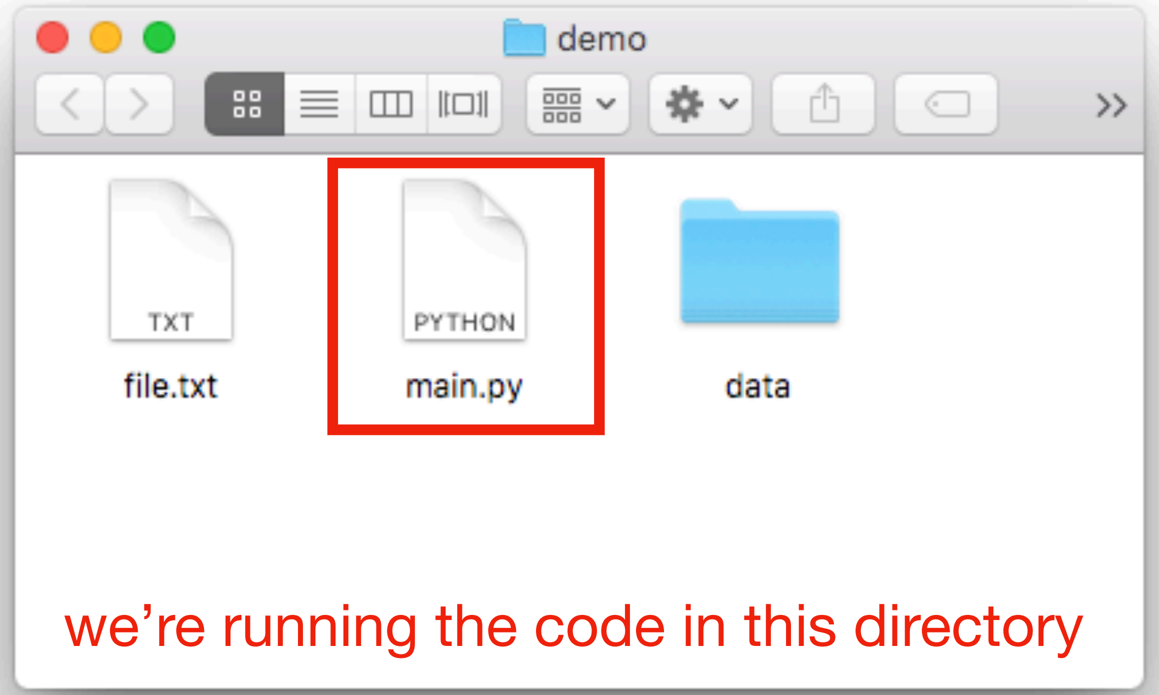
main.py:

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

file object

file path

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



f.close()
```



we're running the code in this directory

File objects

built-in open function

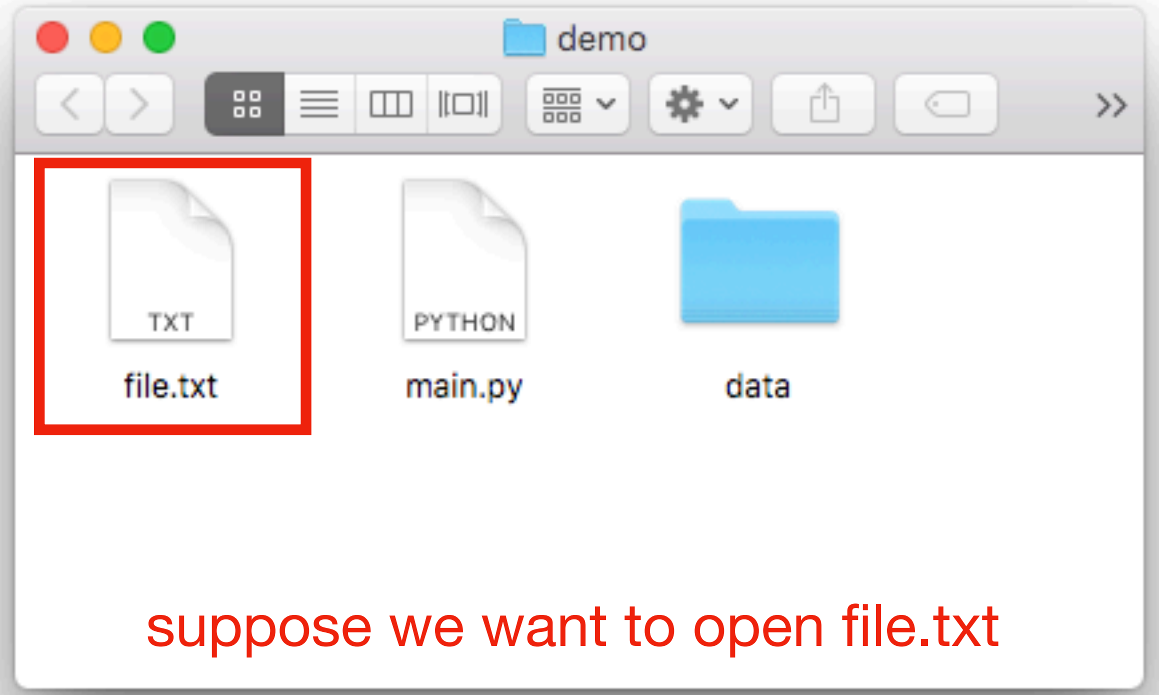
main.py:

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

file object

file path

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



f.close()
```



suppose we want to open file.txt

File objects

built-in open function

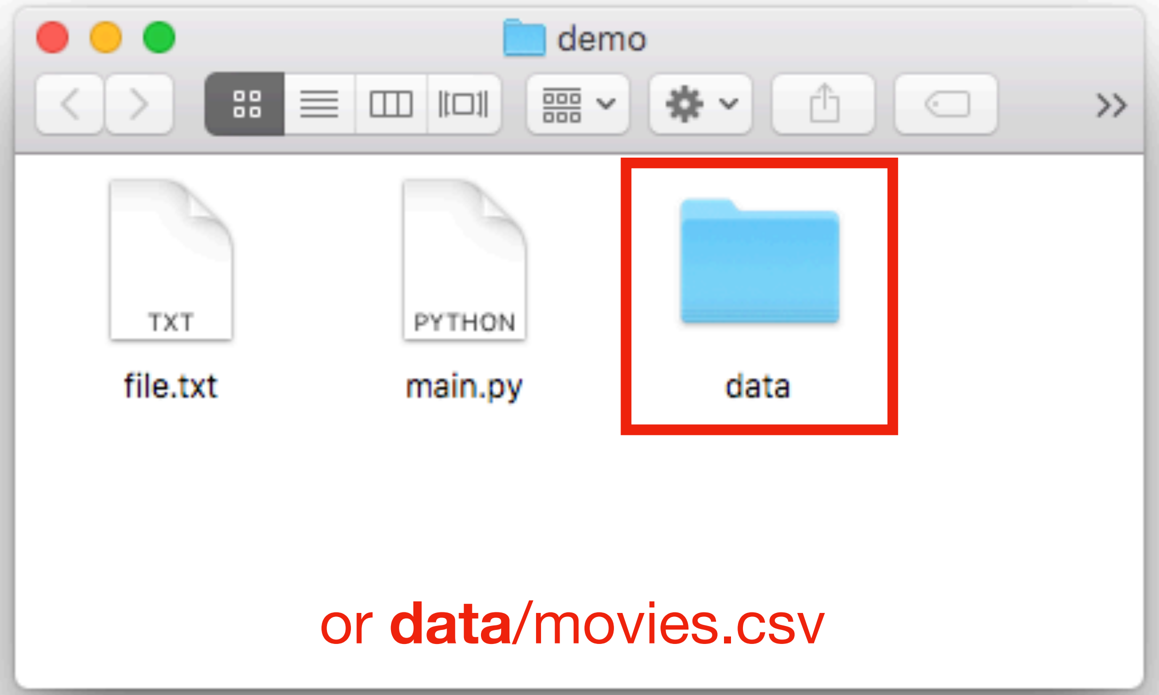
main.py:

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

file object

file path

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



f.close()
```



or **data**/movies.csv

File objects

built-in open function

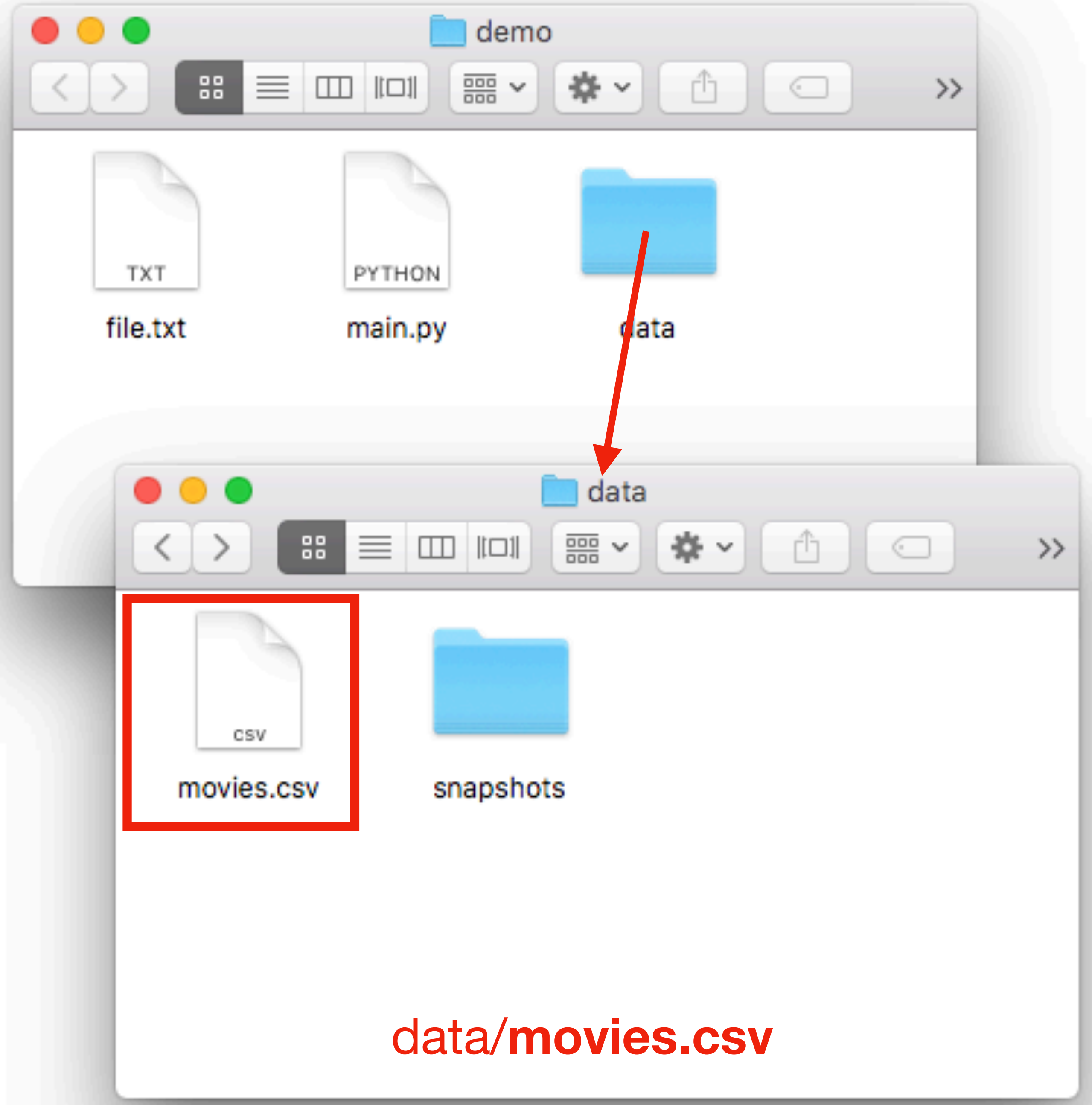
main.py:

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

file object

file path

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



f.close()
```



File objects

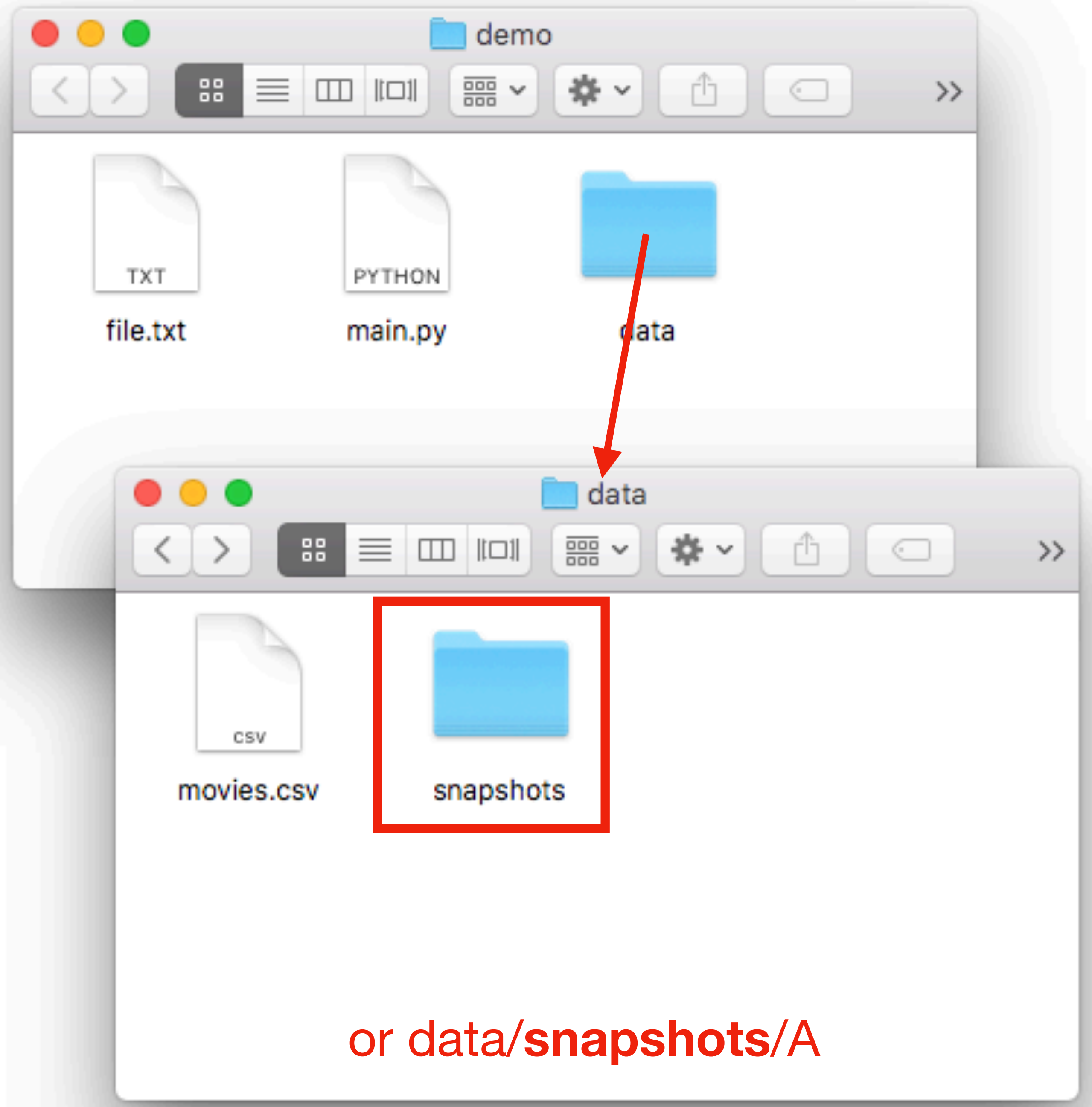
built-in open function

main.py:

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



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



f.close()
```

file object

file path



or data/**snapshots**/A

File objects

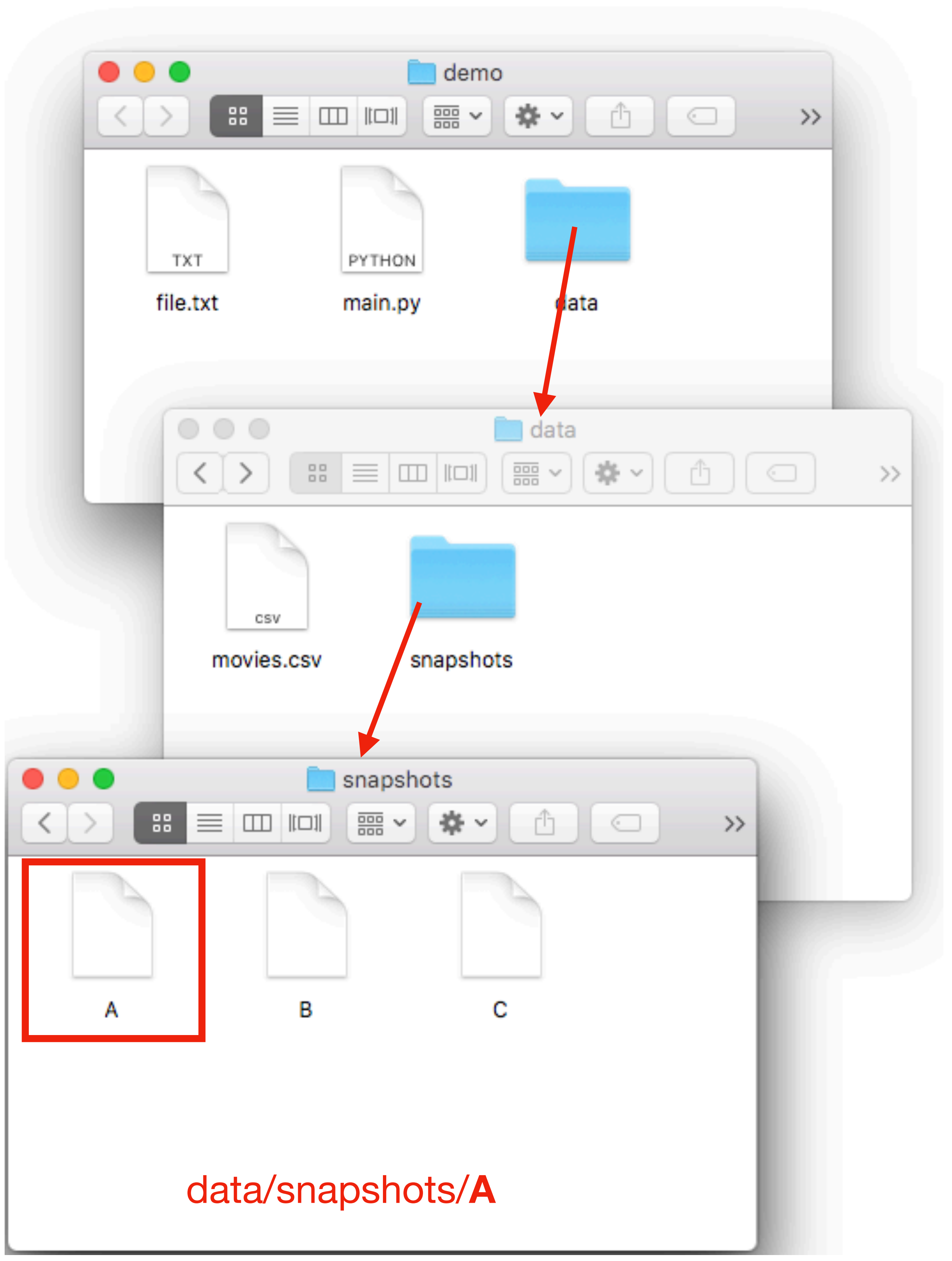
built-in open function

main.py:

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

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


f.close()
```

file object

file path



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


f.close()
```

using file

File objects

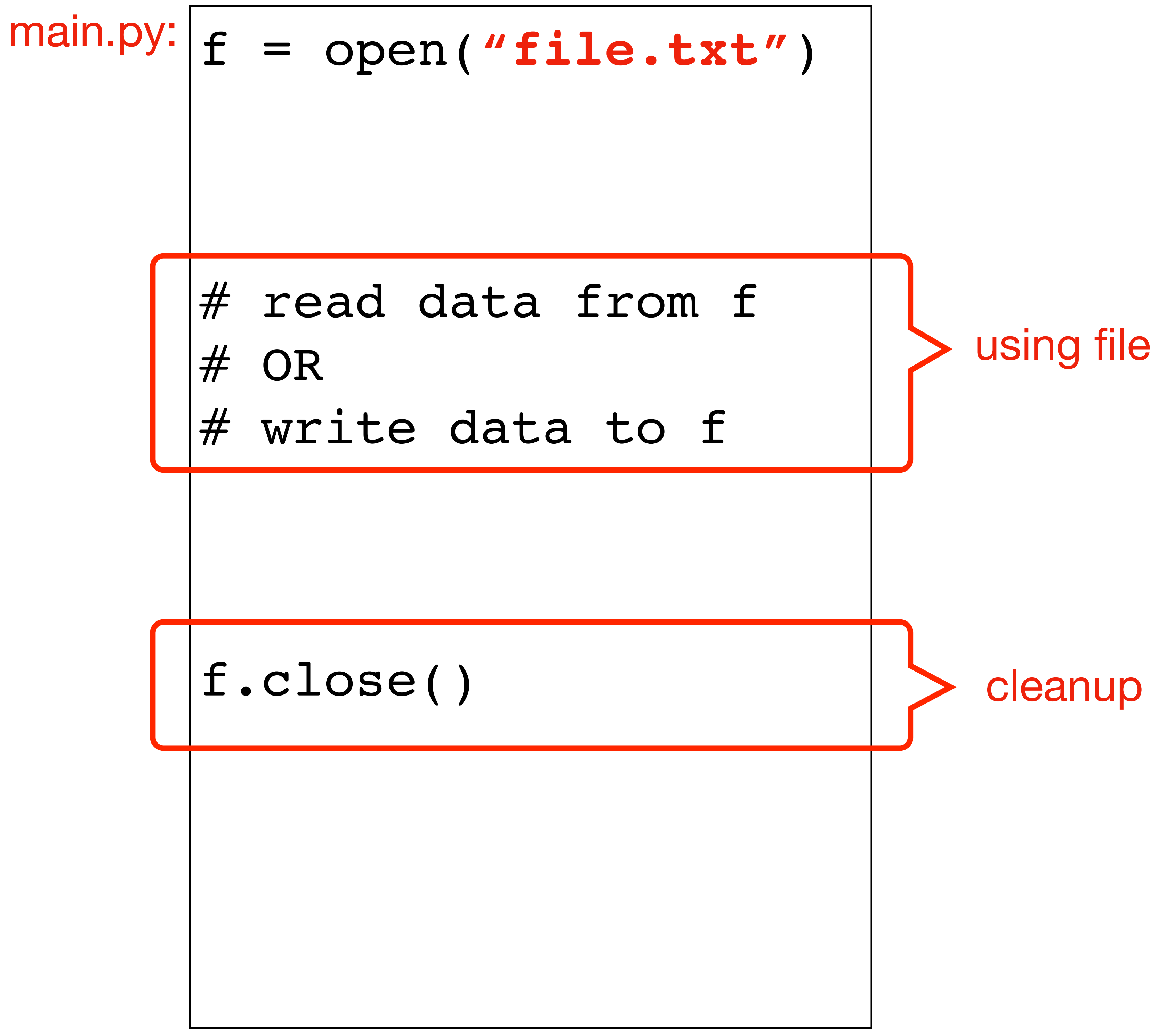
main.py:

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



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



f.close()
```

using file

cleanup

File objects


main.py:

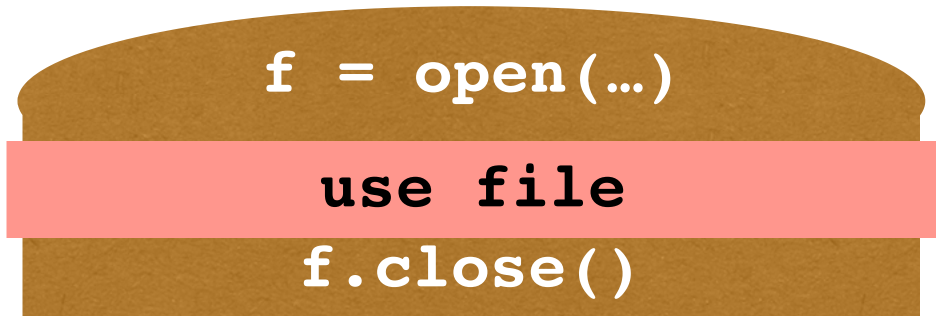
```
f = open("file.txt")


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


f.close()
```

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


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

File formats

- JSON
- CSV

OS module

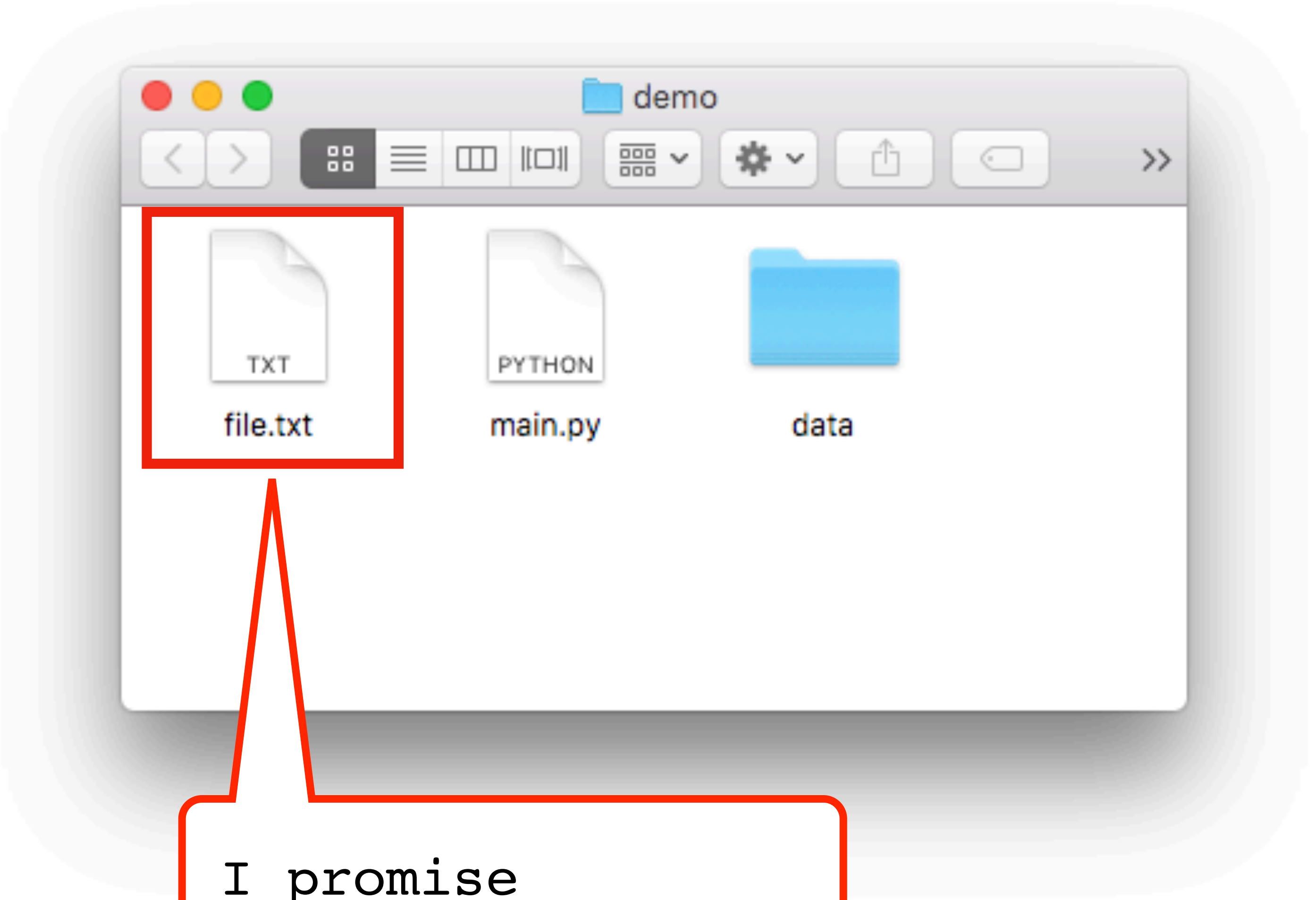
- listdir, mkdir, exists, isdir, isfile, join

File exceptions

Encodings

Reading a file

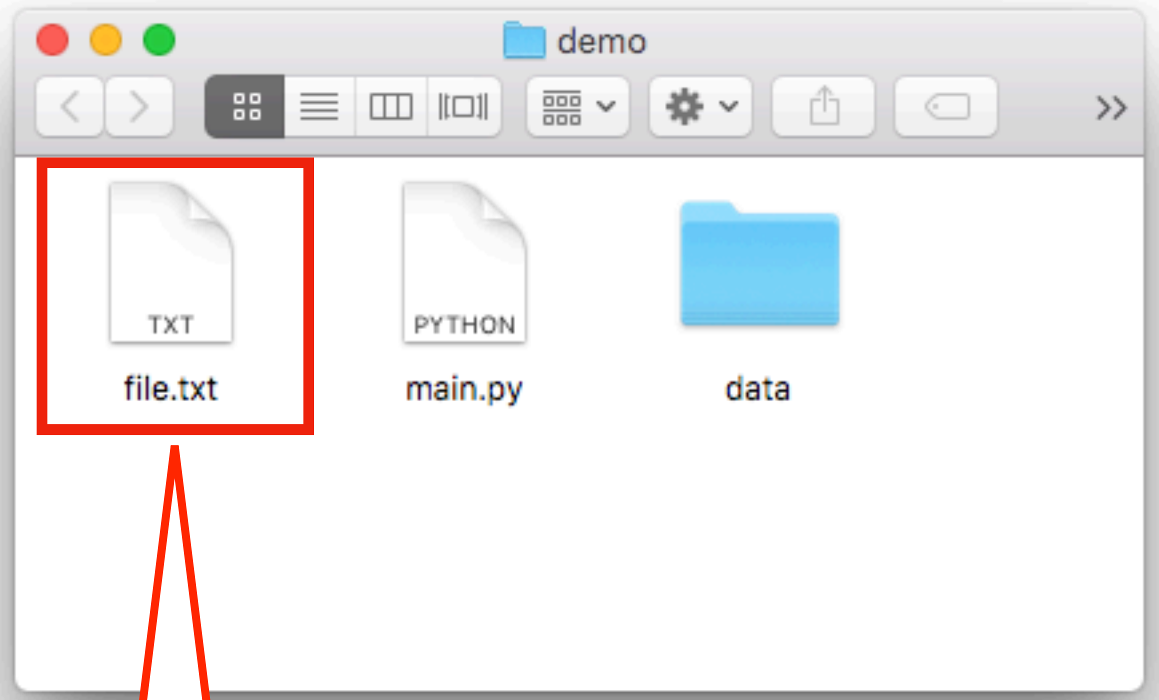
```
f = open("file.txt")



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



f.close()
```



I promise
to always
close my files

Reading a file

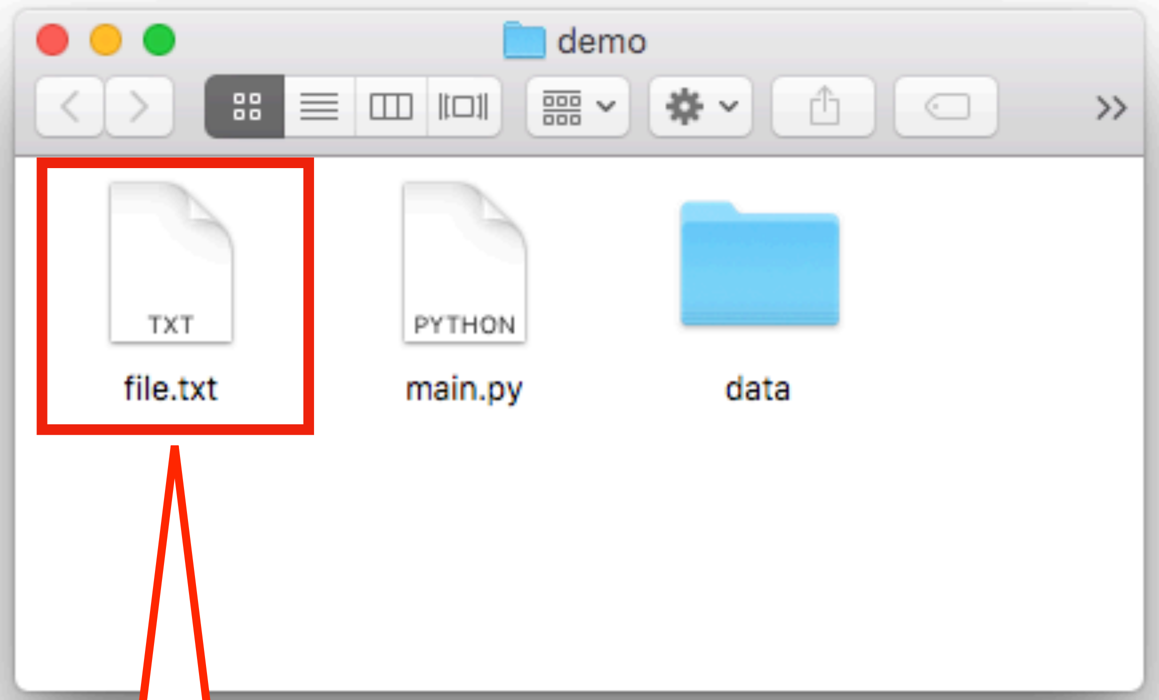
```
f = open("file.txt")



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

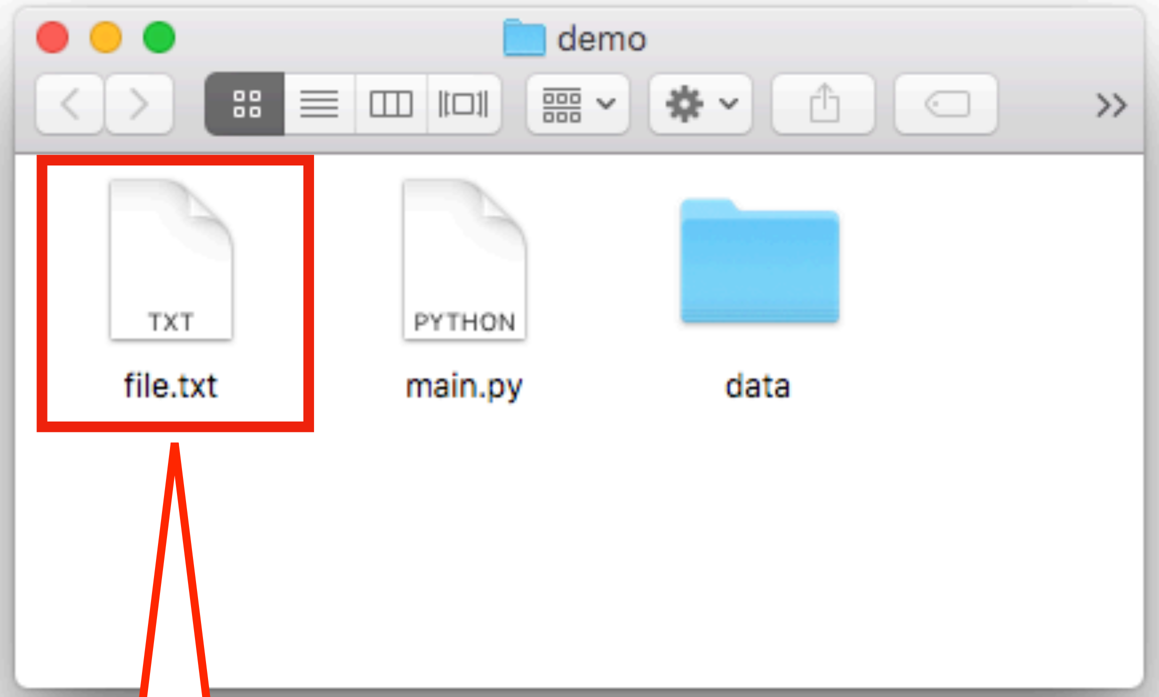
`read()` method

- fetch entire file contents
- return as a string

Reading a file

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



data = f.read()

print(data)


f.close()
```

data is: "I promise\nto always\nclose my files"



```
I promise
to always
close my files
```

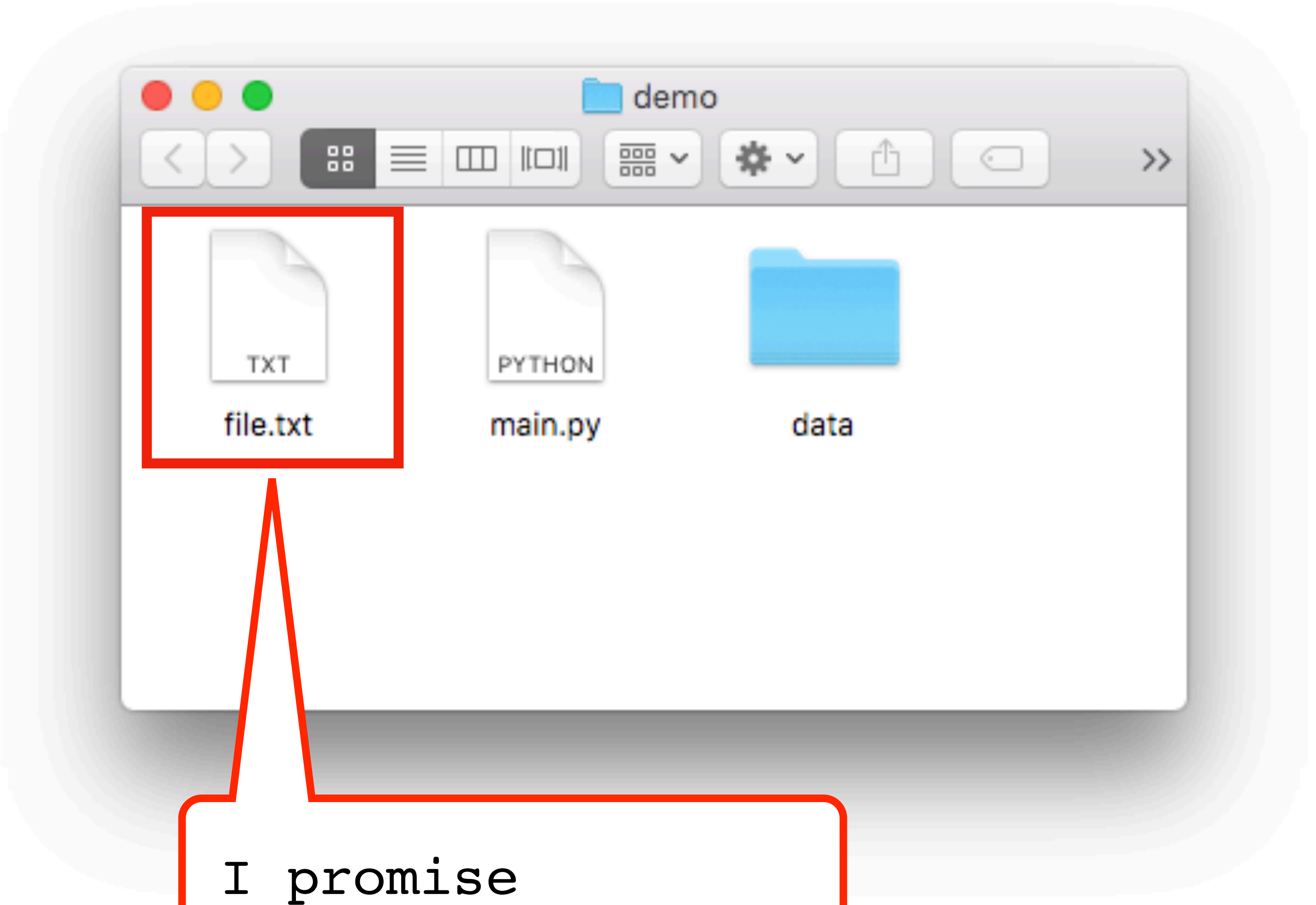
`read()` method

- fetch entire file contents
- return as a string

Reading a file

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



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


print(data)


f.close()
```

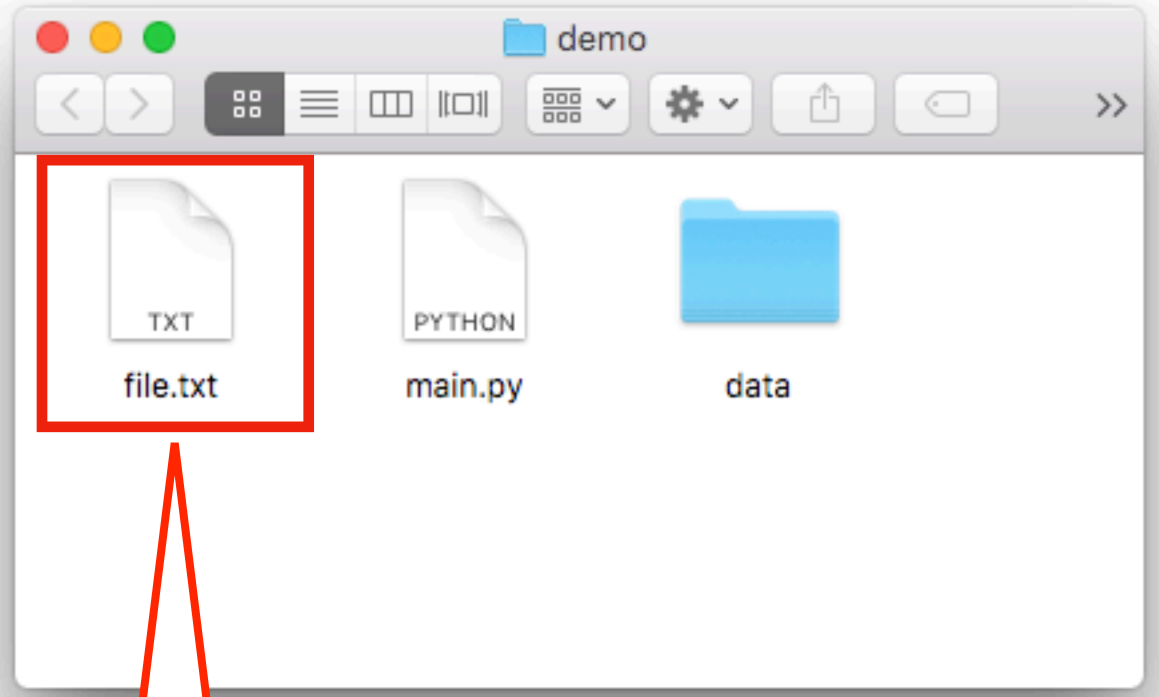
data is: ["I promise", "to always", "close my files"]



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

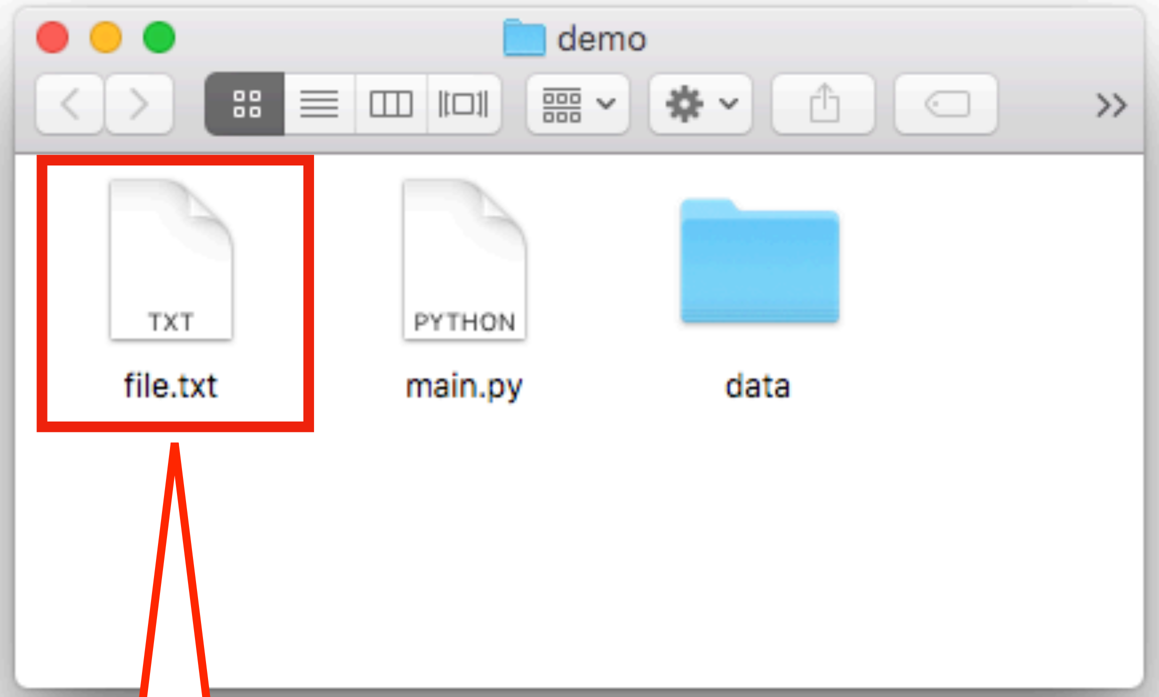
recall a **file object** is an **iterator**

- **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"]`



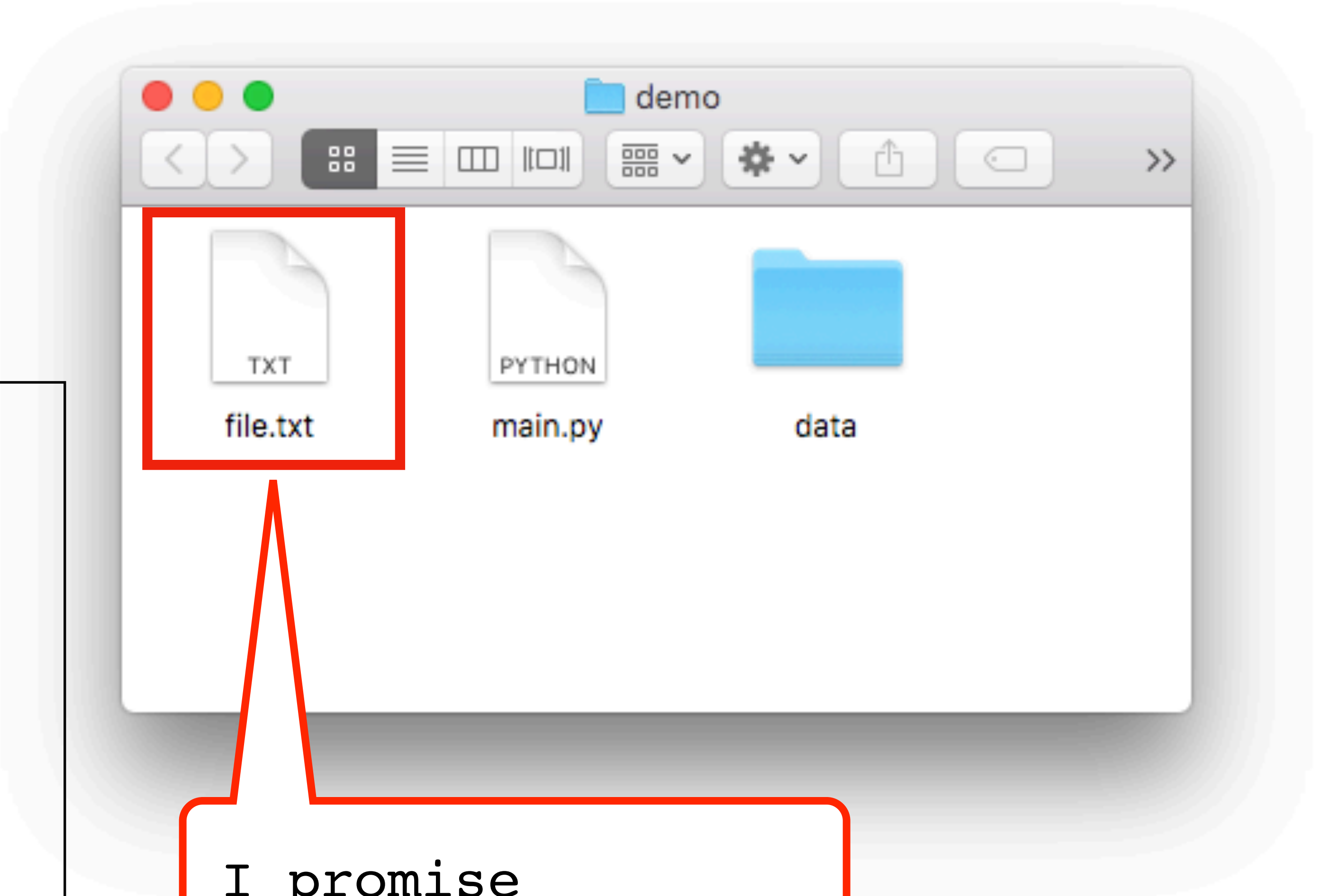
```
I promise
to always
close my files
```

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()
```

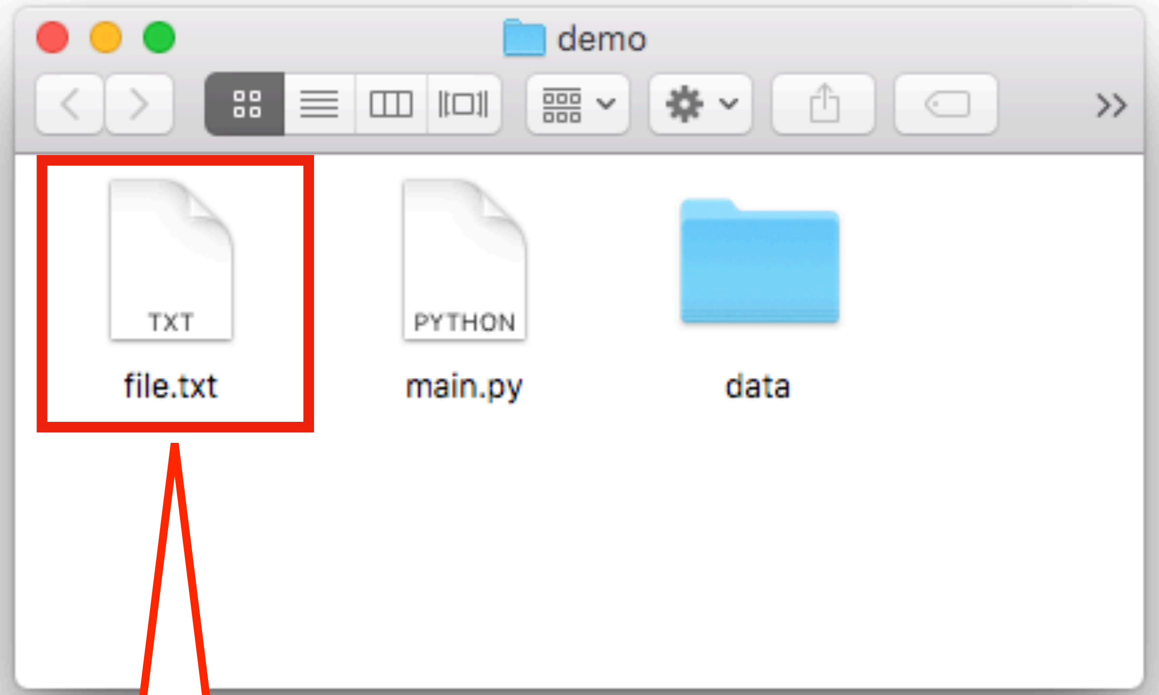


I promise
to always
close my files

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()
```

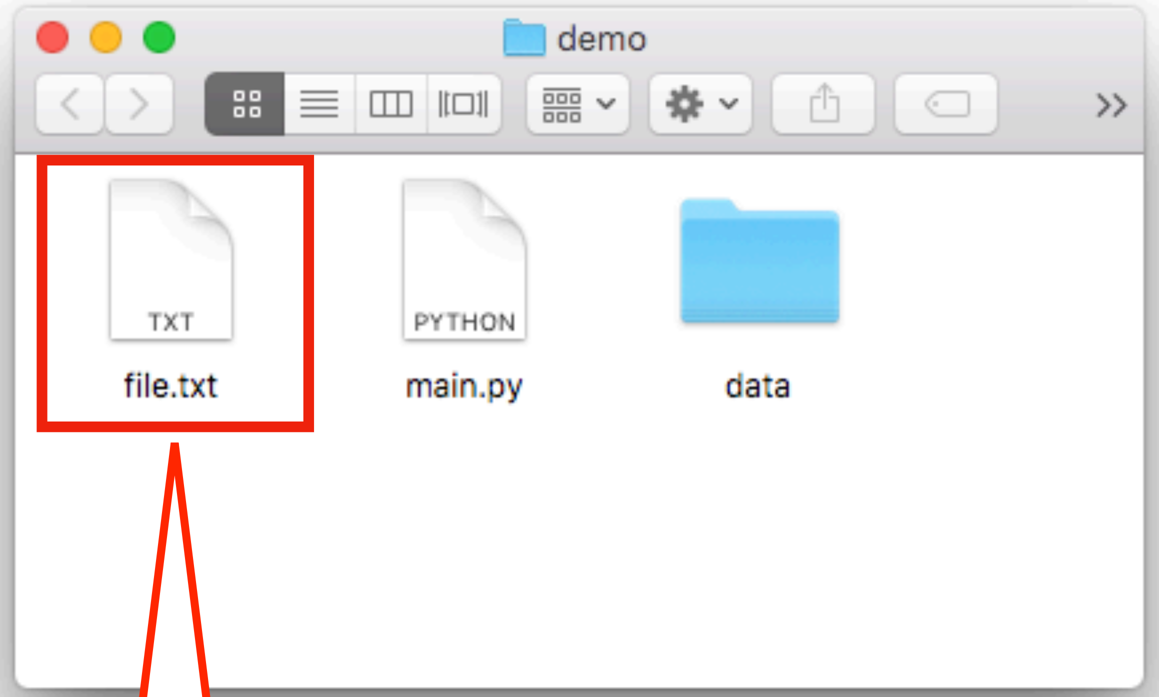


I promise
to always
close my files

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()
```

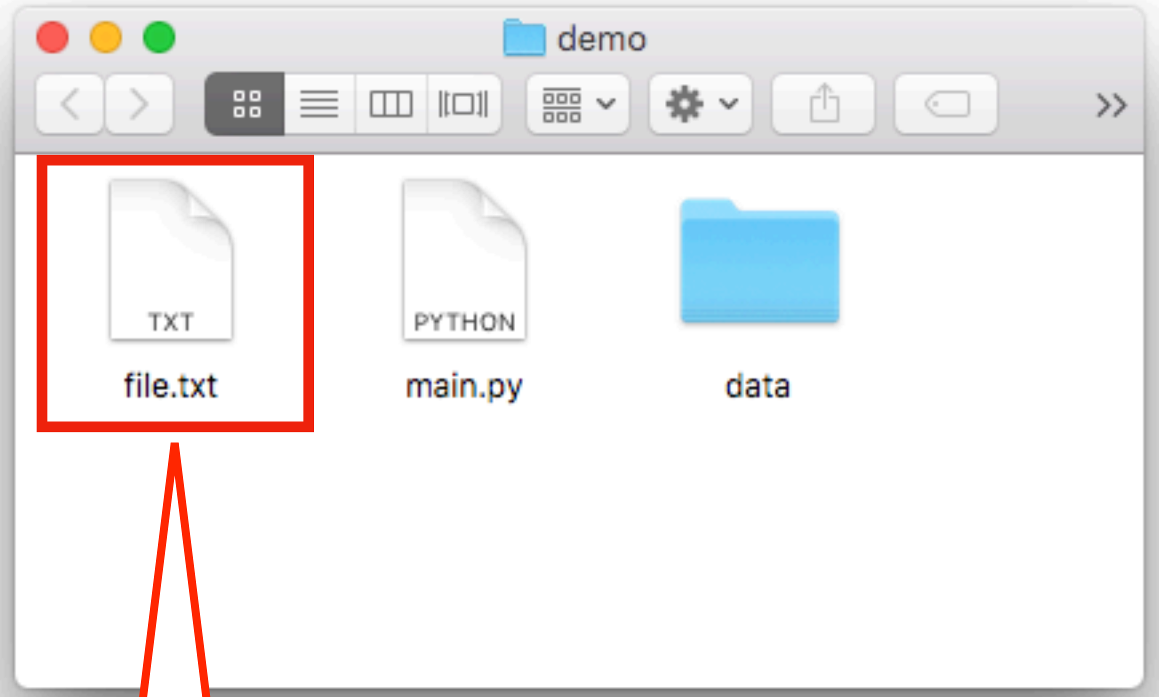


I promise to always close my files

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()
```



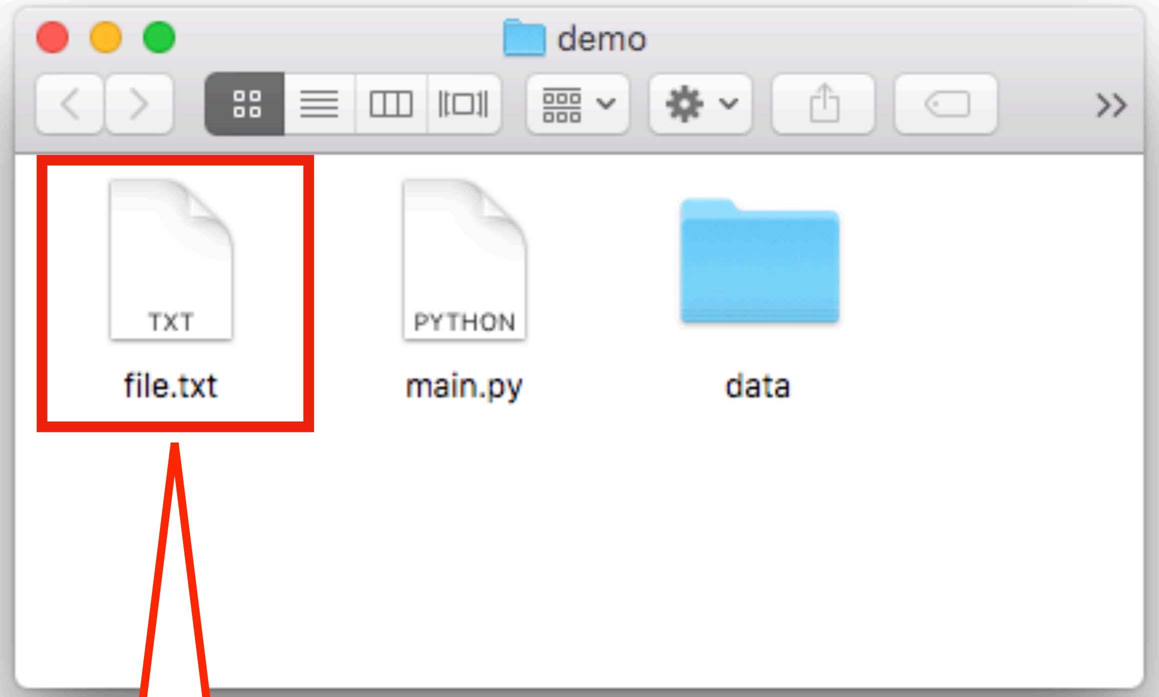
I promise to always close my files

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()
```

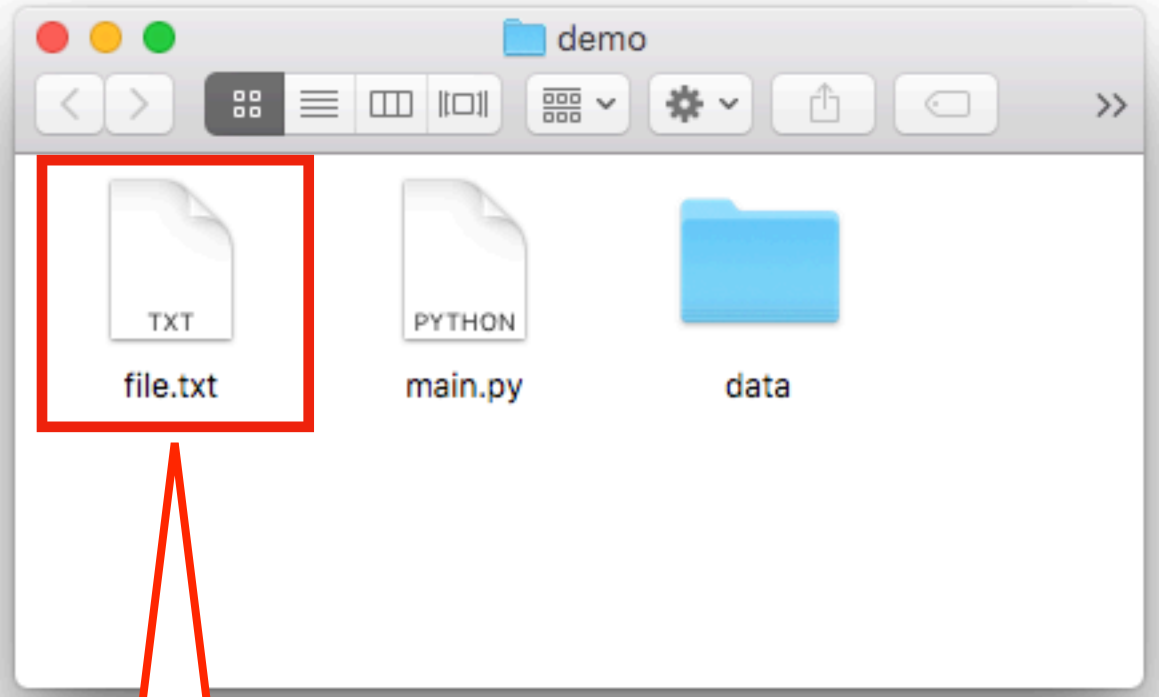


opening with “w” is dangerous. It immediately wipes out your file.

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()
```



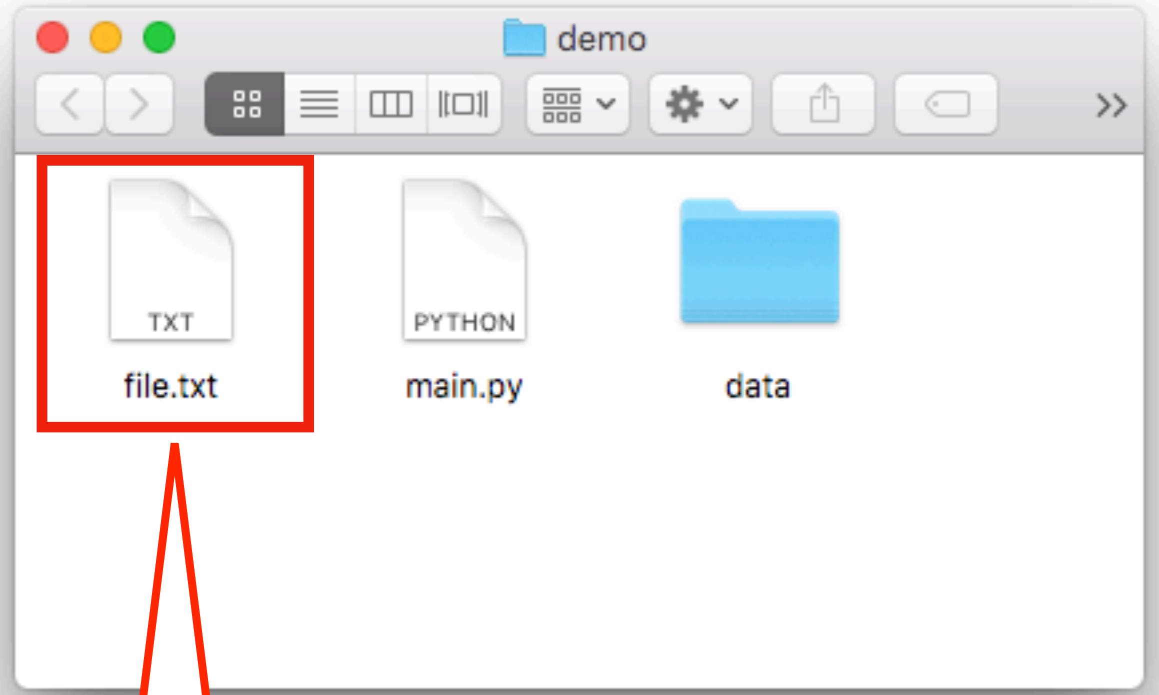
opening with "w" is dangerous. It immediately wipes out your file.

(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()
```

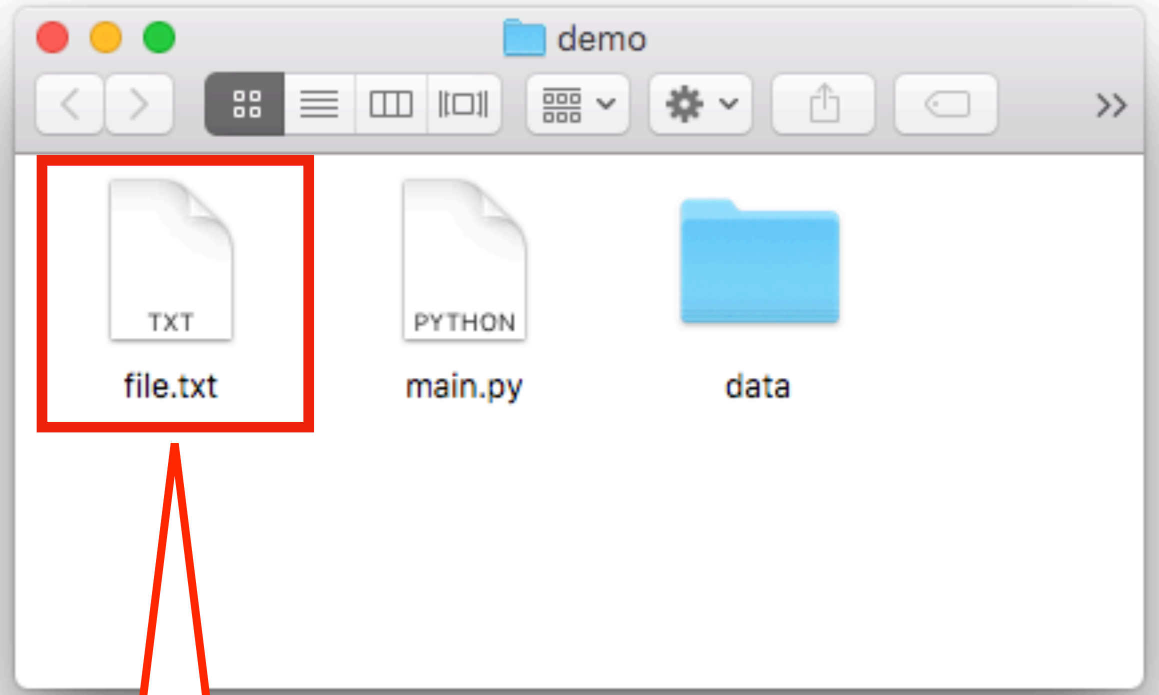


```
hello
```

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()
```

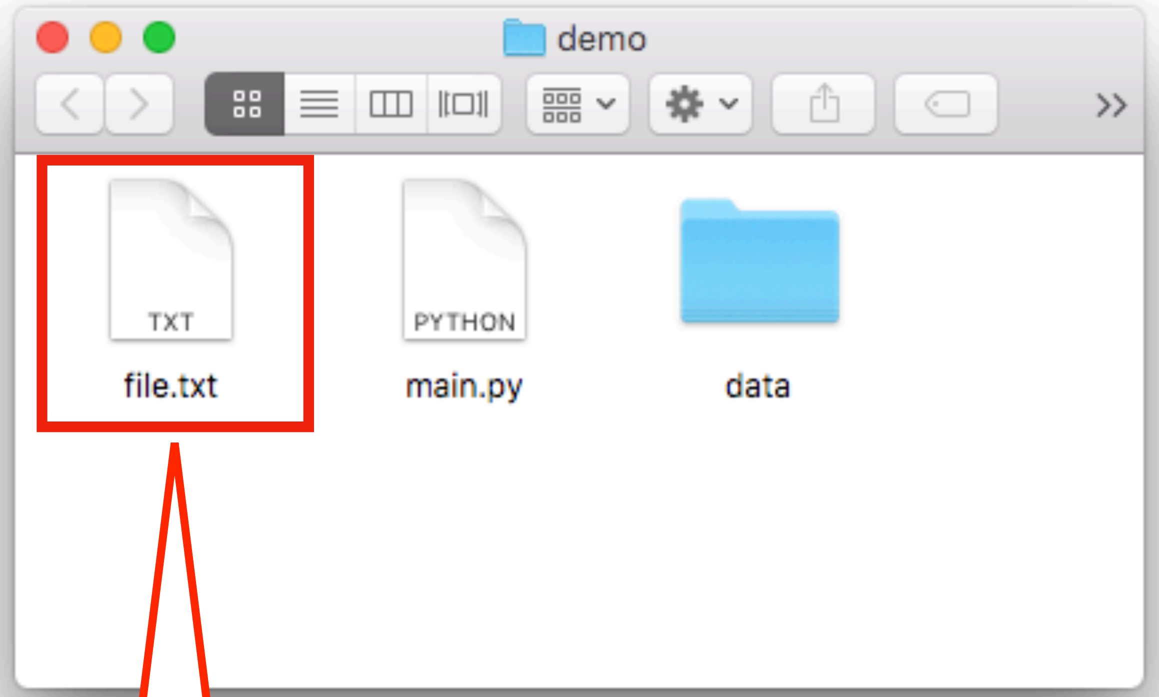


```
hello world
```

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()
```

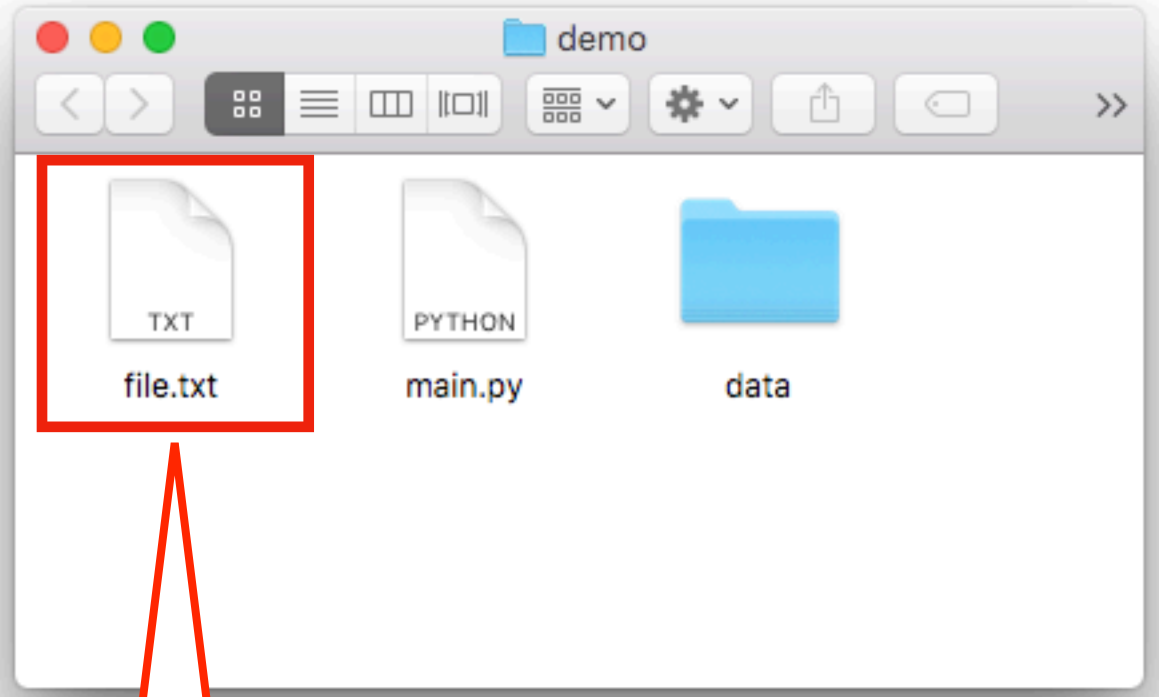


```
hello world
!!!!!!
```

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

data.json

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

`fileobj.read()`

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

```
f = open(“data.json”)
raw = f.read()
```

`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()
```

```
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

data.json

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

`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)

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

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

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

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

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

data.csv

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

```
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()
```

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

Reading CSVs

data.csv

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

```
import csv

~~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()
```

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

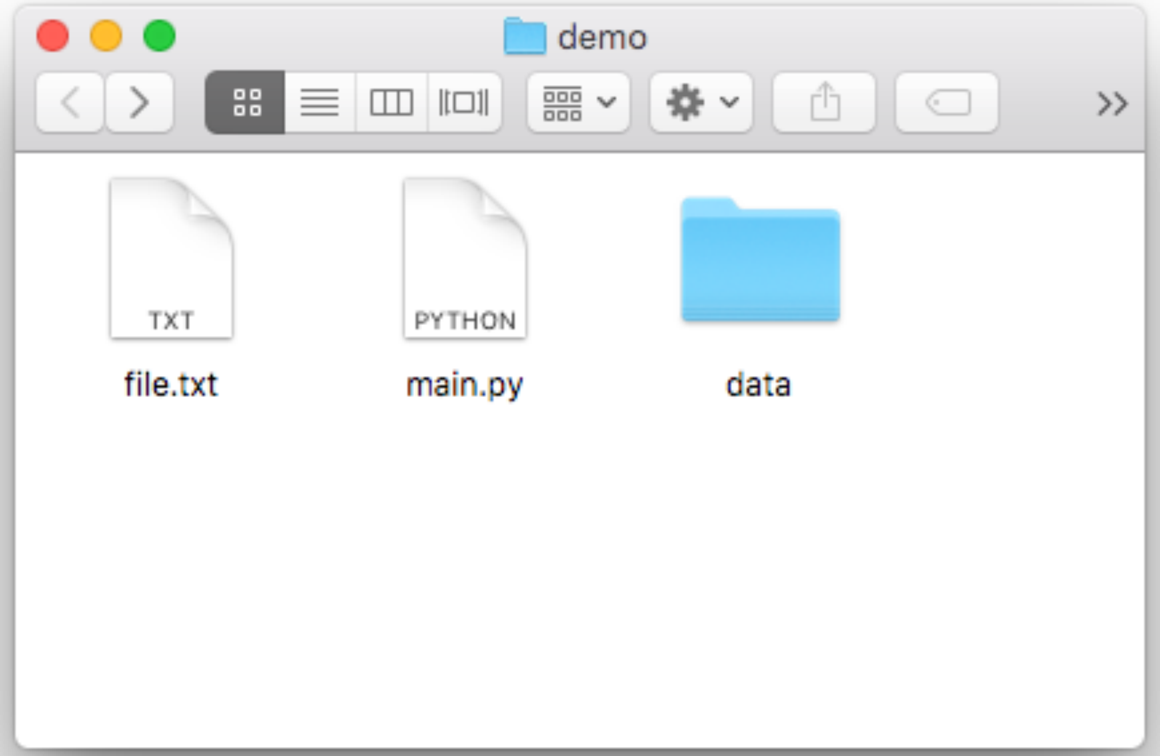
- os.listdir
- os.mkdir
- os.path.exists
- os.path.isfile
- os.path.isdir
- os.path.join

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”]
```

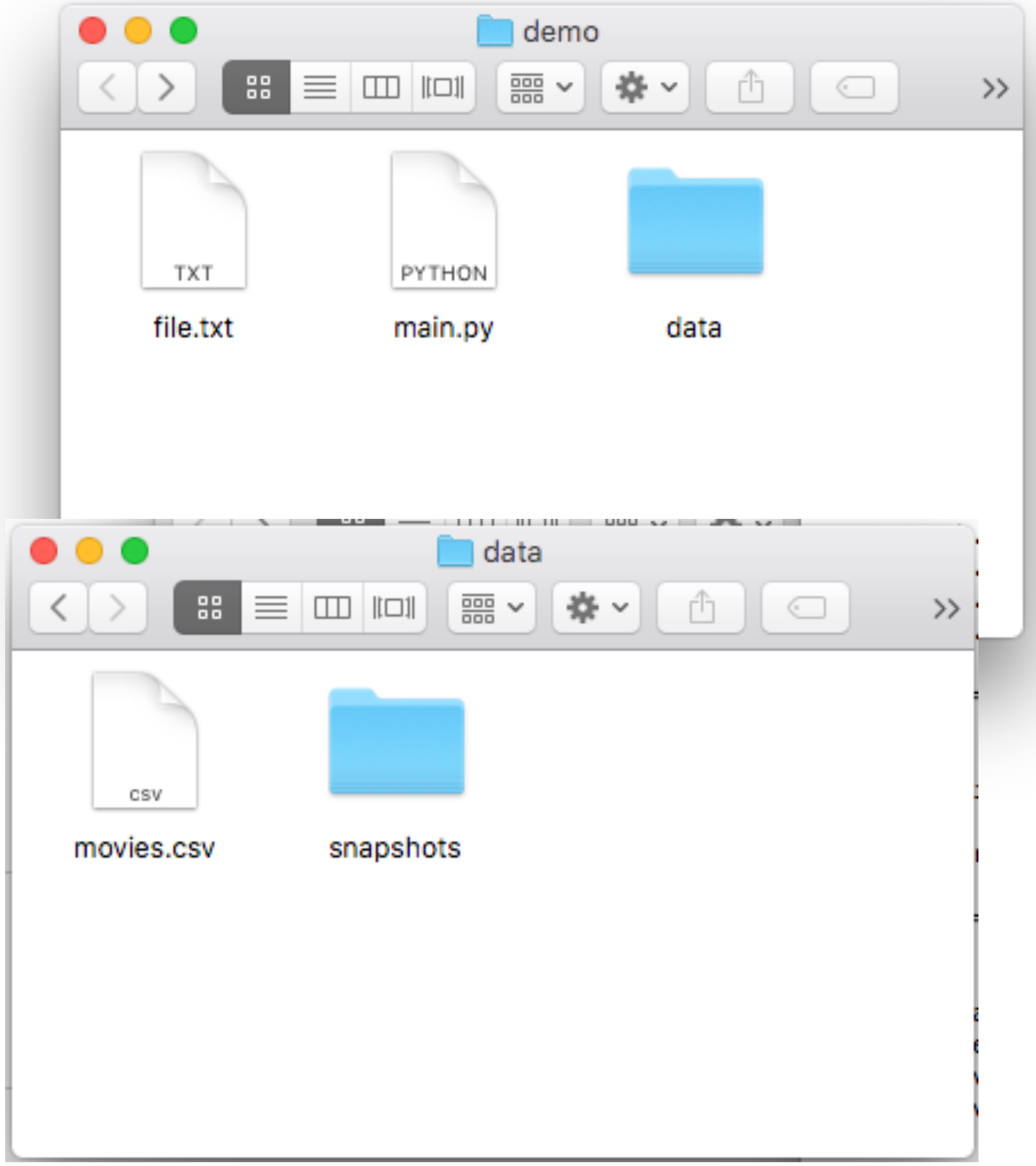


OS Module (Operating System)

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"]
```

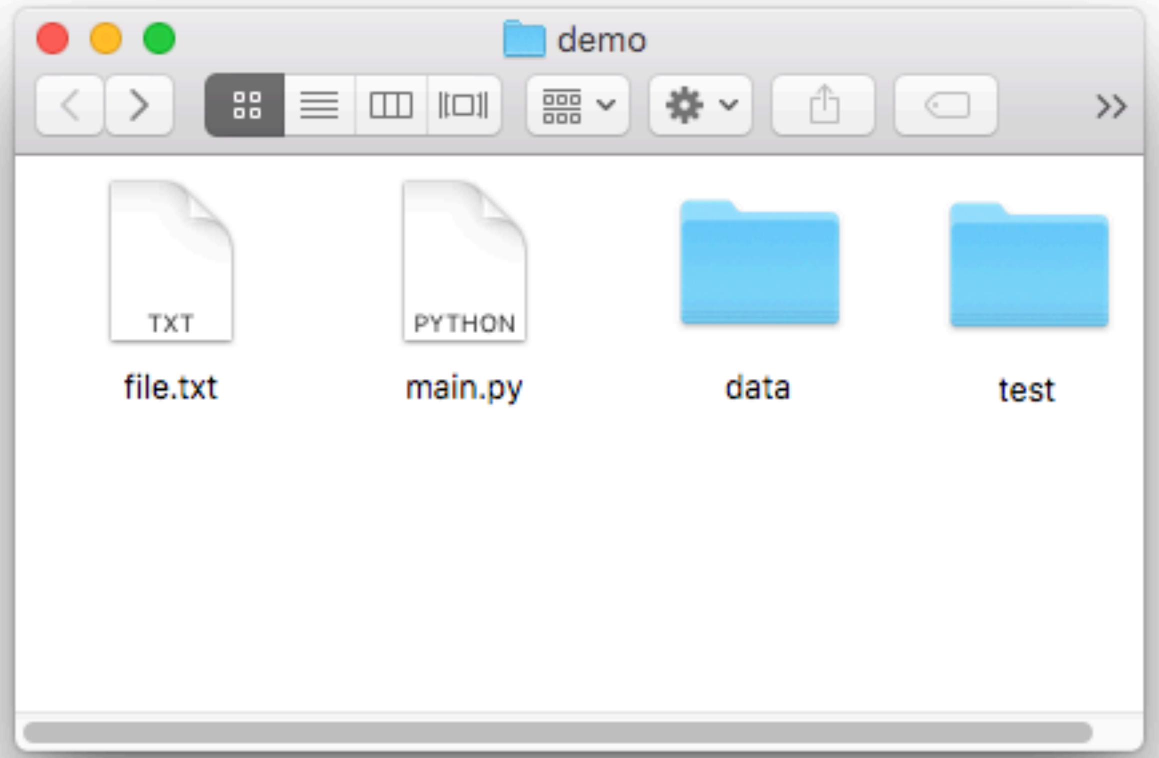


OS Module (Operating System)

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”)
```

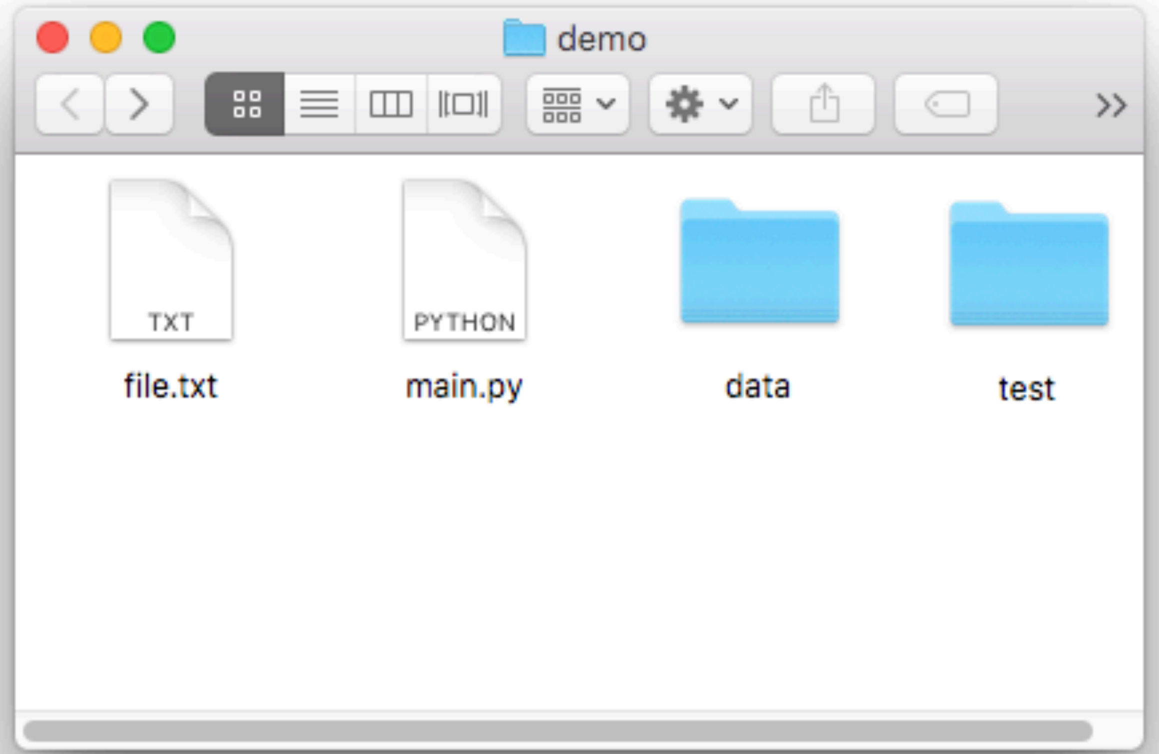


OS Module (Operating System)

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.path.exists("file.txt")
True
```

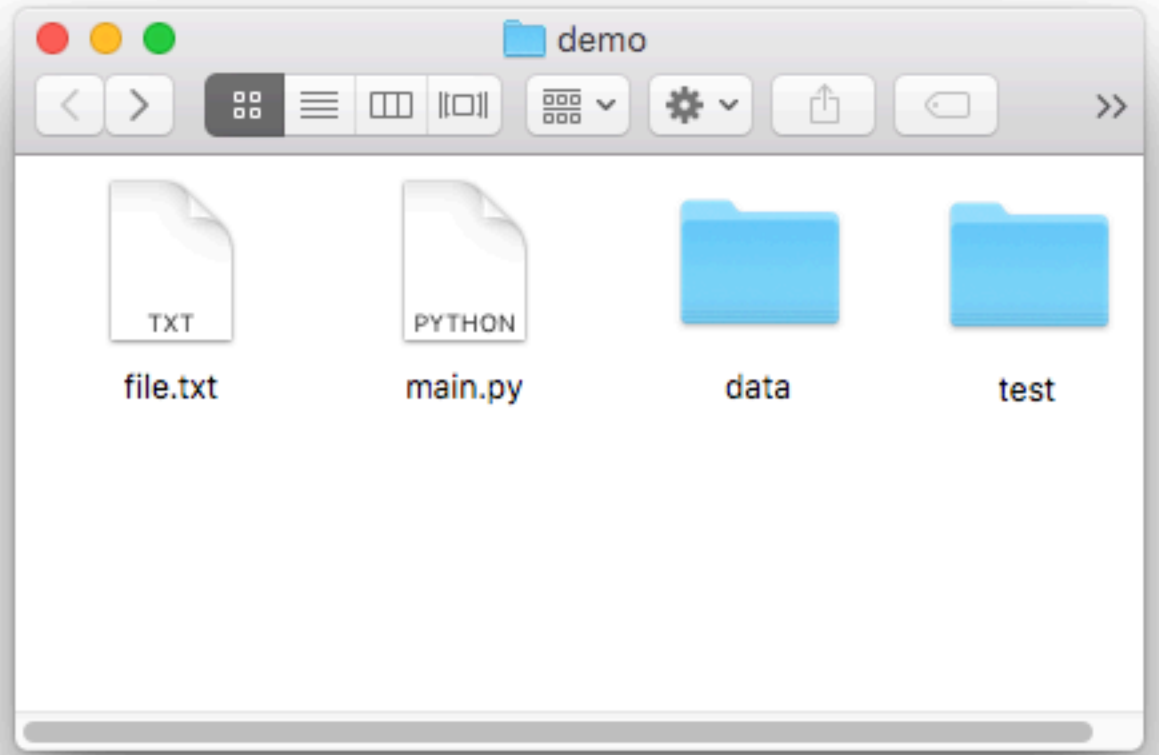


OS Module (Operating System)

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
```

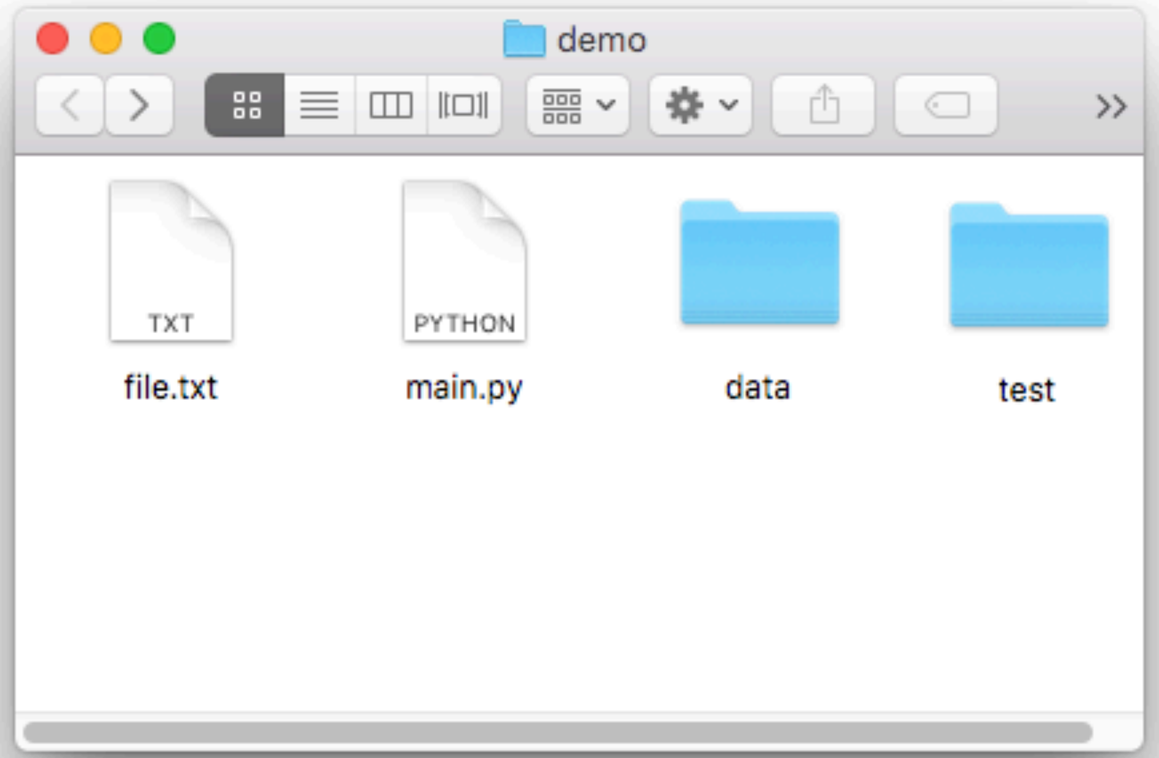


OS Module (Operating System)

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
```

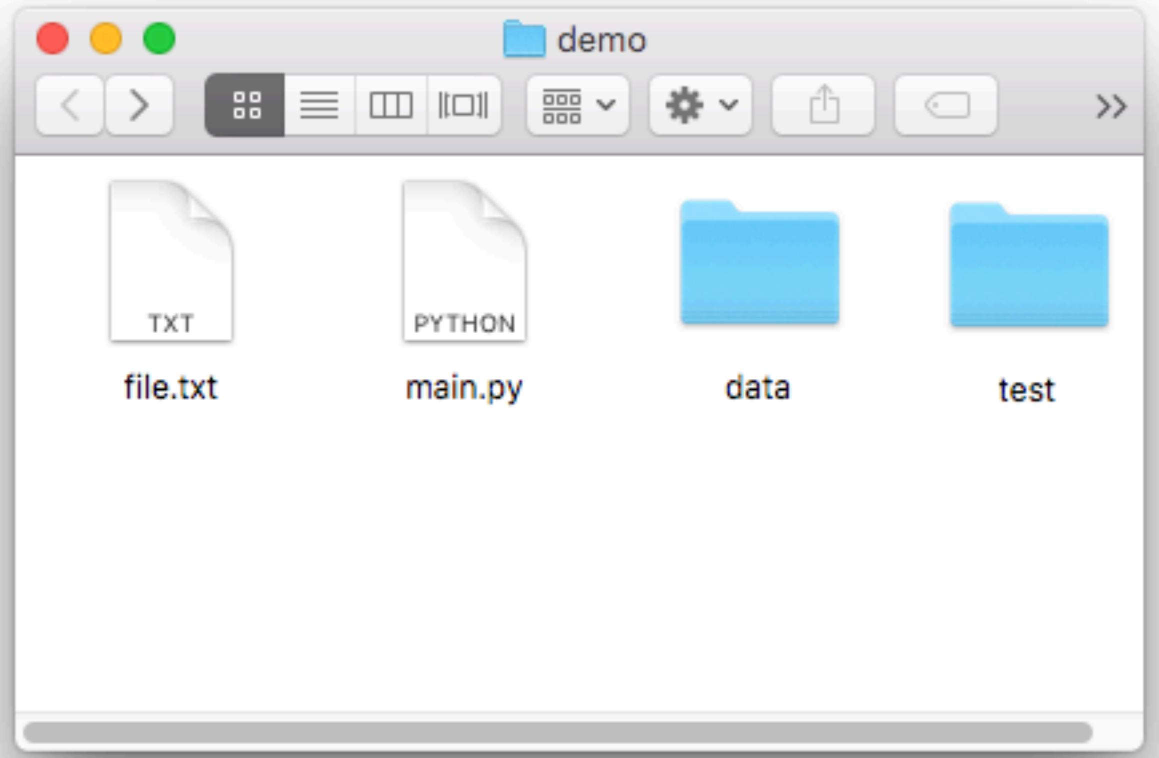


OS Module (Operating System)

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
```

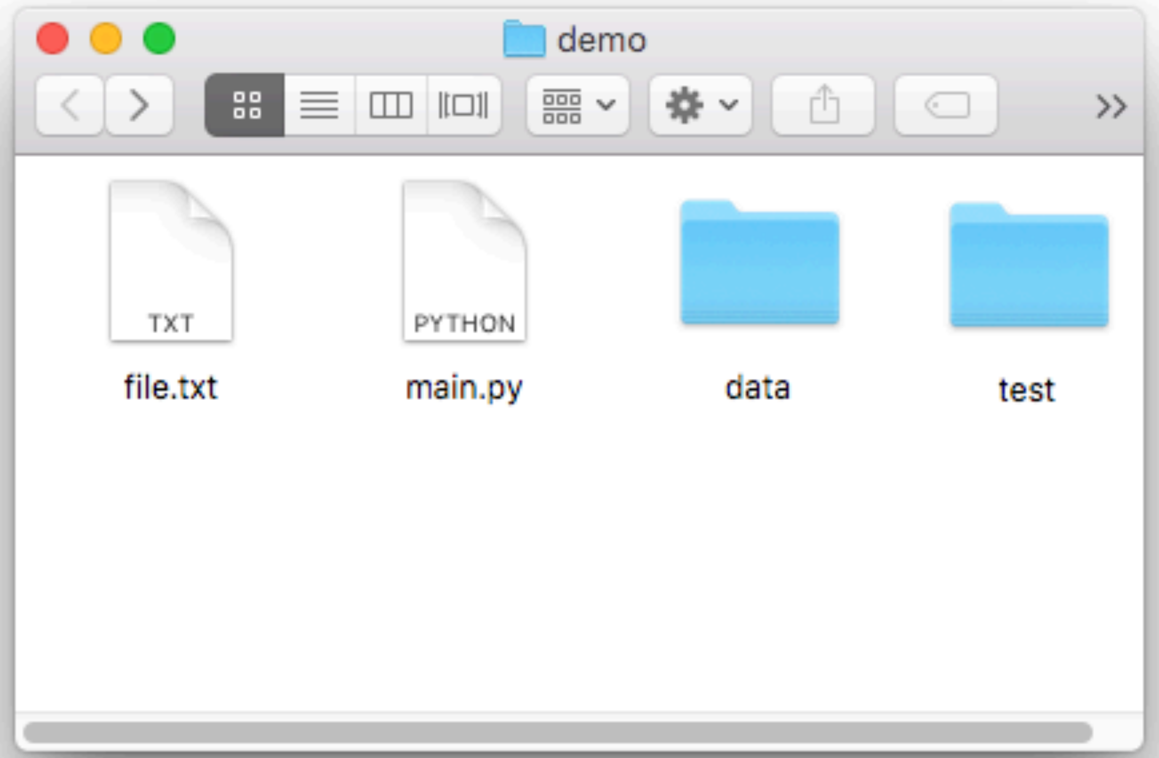


OS Module (Operating System)

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
```

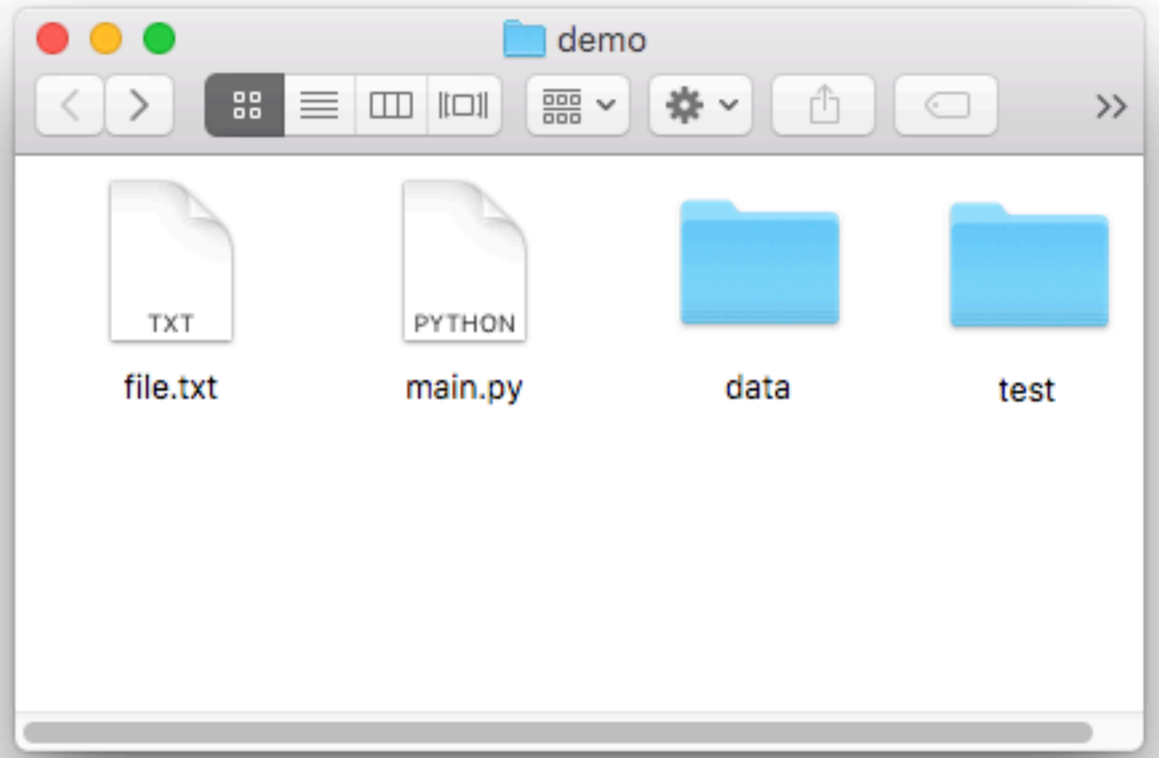


OS Module (Operating System)

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)

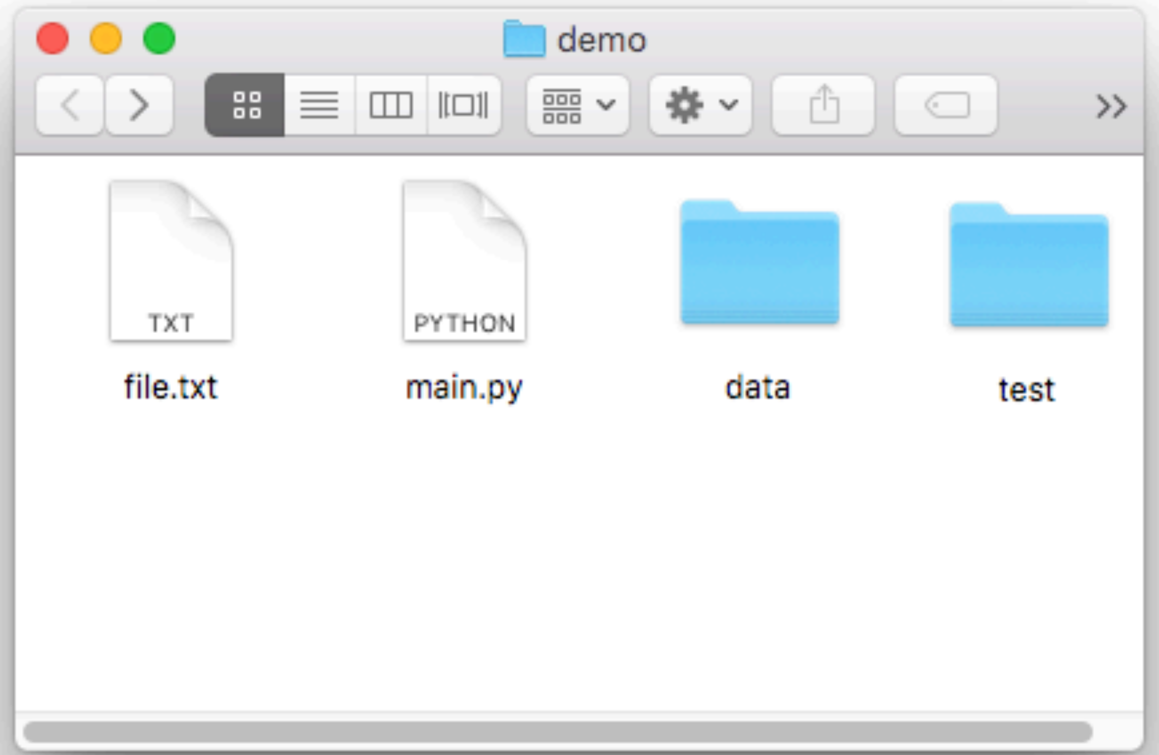
Many functions in os and os.path for working w/ files

- 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)

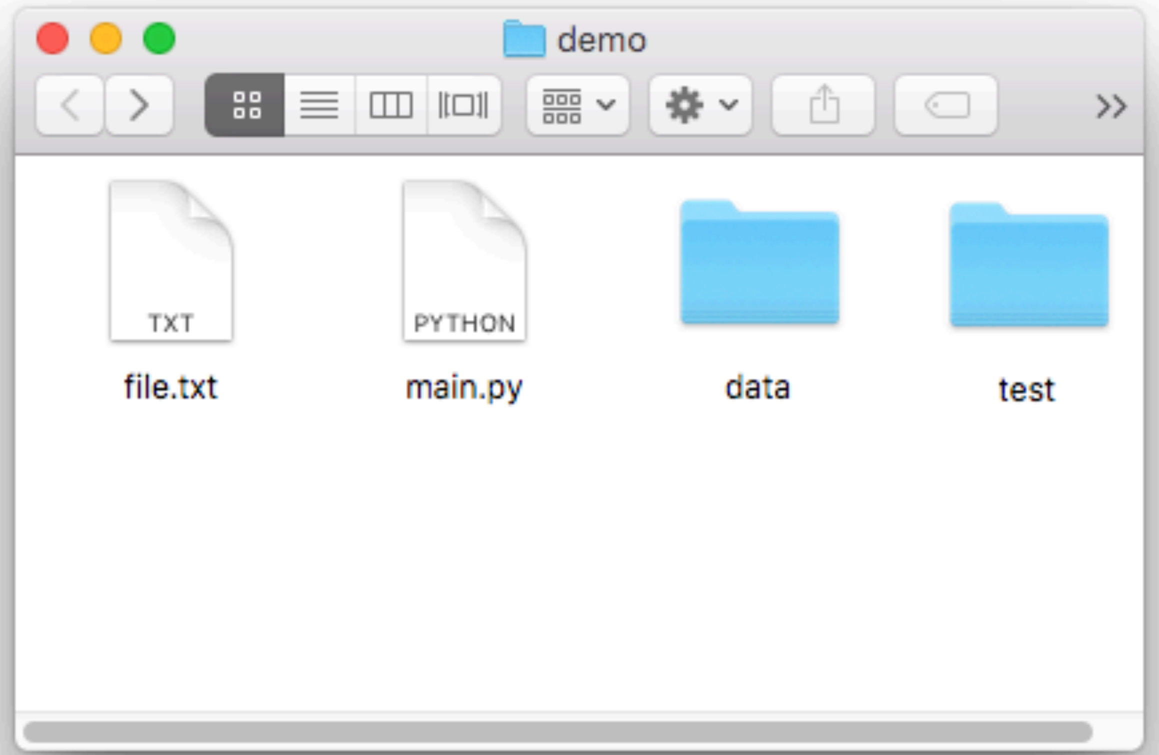
Many functions in os and os.path for working w/ files

- 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

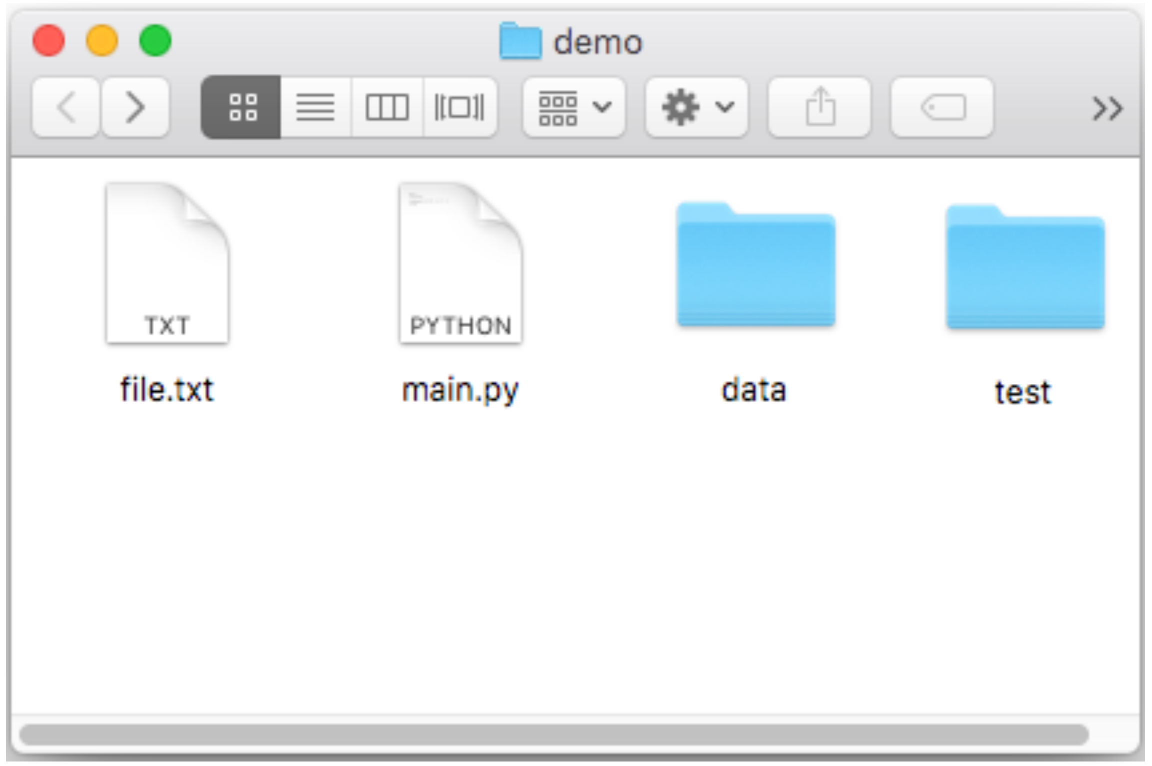
Working with files leads to many exceptions

- 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

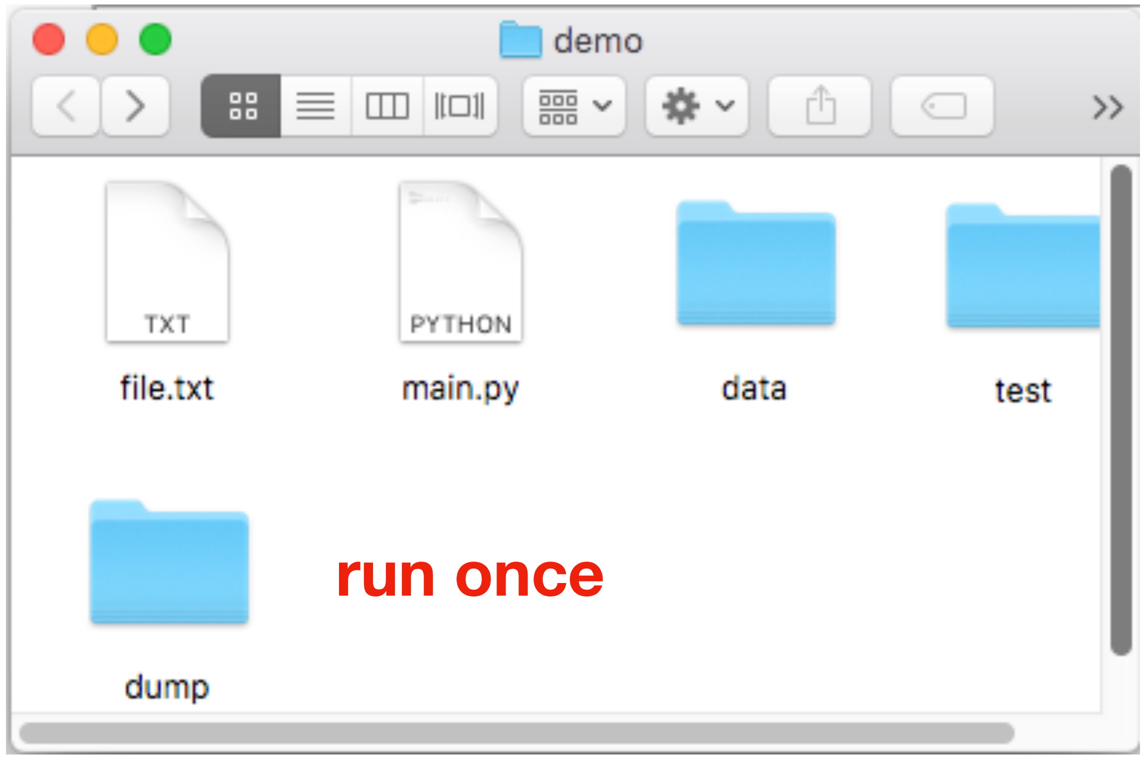


```
import os

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



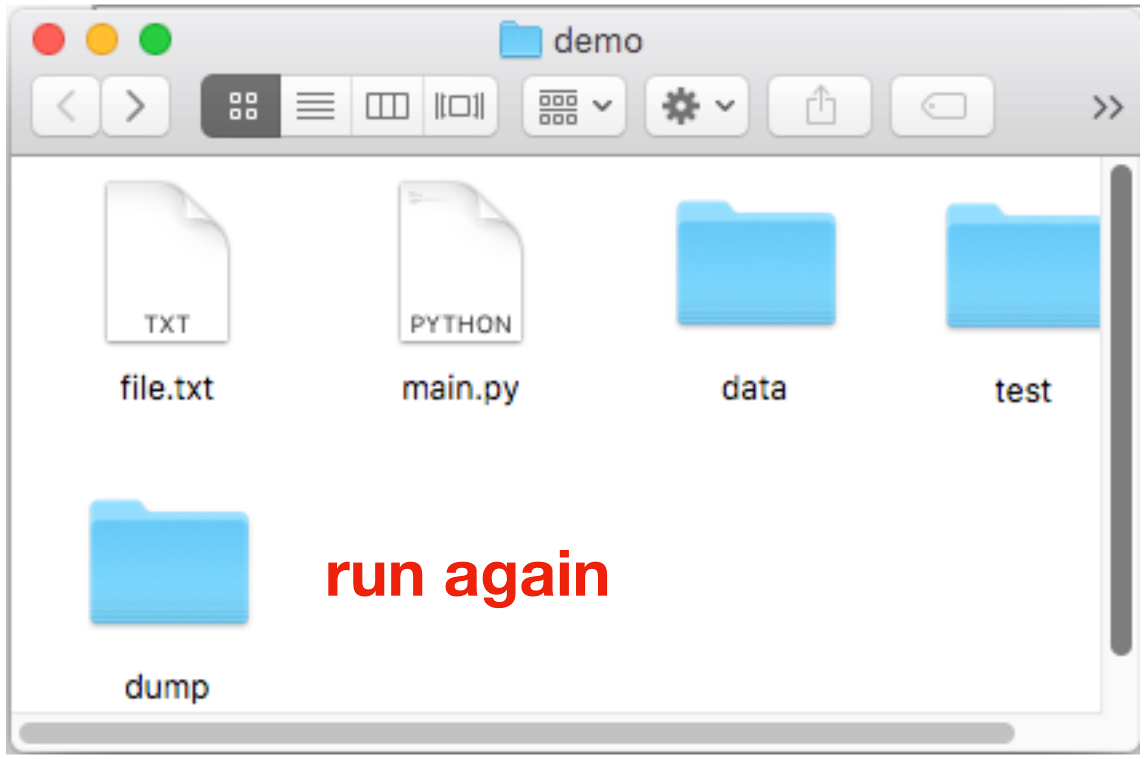
run once

```
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

```
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

Character	**Bits**
A	1100
B	0011
C	1111
D	0000

$$\textbf{“CAB”} \xrightarrow{\text{encoding A}} \mathbf{1111\ \ 1100\ \ 0011}$$

Encodings

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

This translation is called encoding.

encoding A

Character	Bits
A	1100
B	0011
C	1111
D	0000

$$\begin{tikzcd} \textbf{“CAB”} \arrow[r, "\text{encoding A}"] & \textbf{1111 1100 0011} \arrow[r, "\text{decoding A}"] & \textbf{“CAB”} \end{tikzcd}$$

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

$$\begin{tikzcd}
\textbf{“CAB”} \arrow[r, "\text{encoding A}"] & \mathbf{1111\ \ 1100\ \ 0011} \arrow[r, "\text{decoding B}"] & \textbf{?}
\end{tikzcd}$$

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

Character	**Bits**
A	1100
B	0011
C	1111
D	0000

encoding B

Character	**Bits**
A	1100
B	1111
C	1001
D	0011

$$\begin{tikzcd}
\textbf{“CAB”} \arrow[r, "\text{encoding A}"] & \textbf{1111 1100 0011} \arrow[r, "\text{decoding B}"] & \textbf{“BAD”}
\end{tikzcd}$$

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.

```python
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.

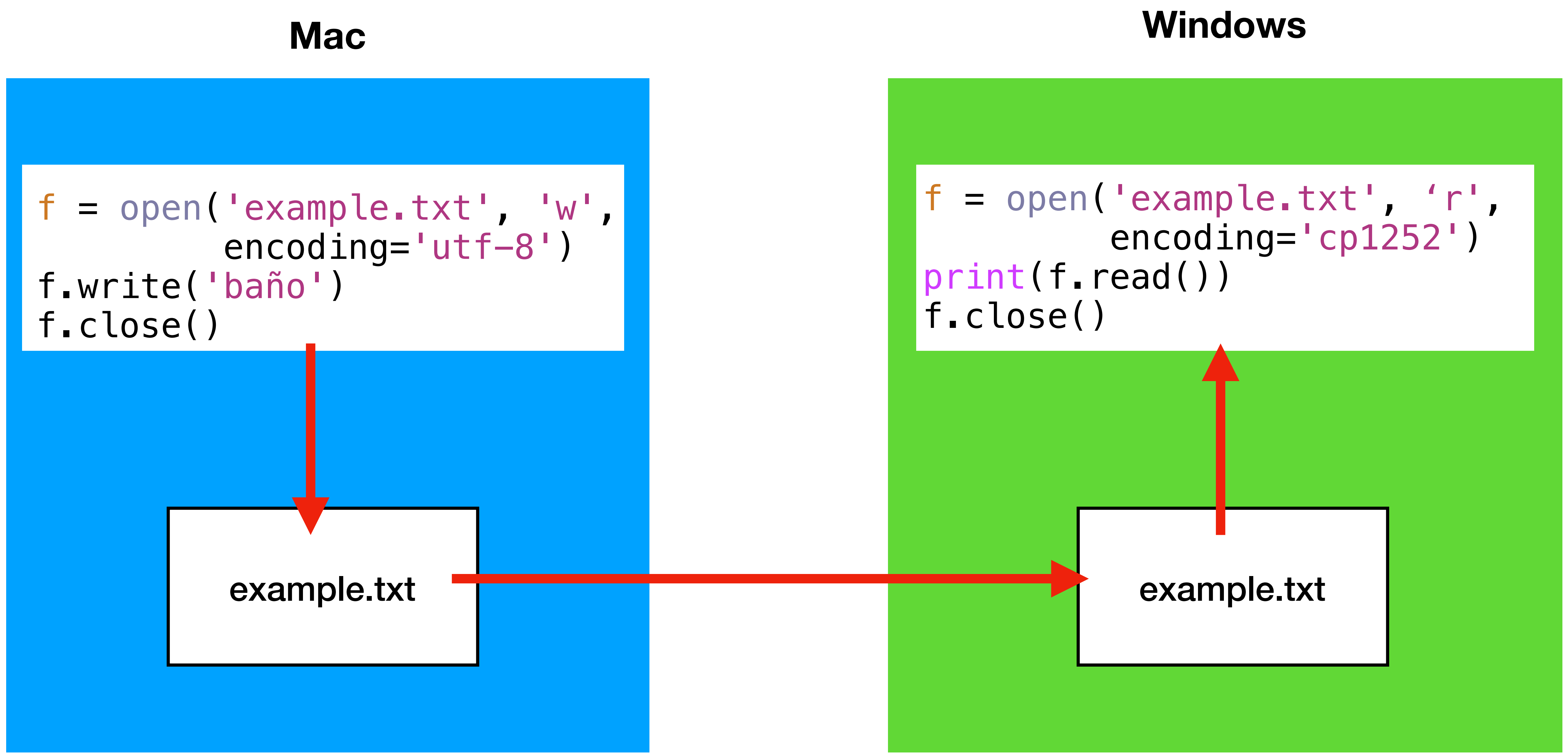
Python uses different default encodings on Mac and Windows.

Problematic for special characters.

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

override default

Encoding Defaults Done Wrong



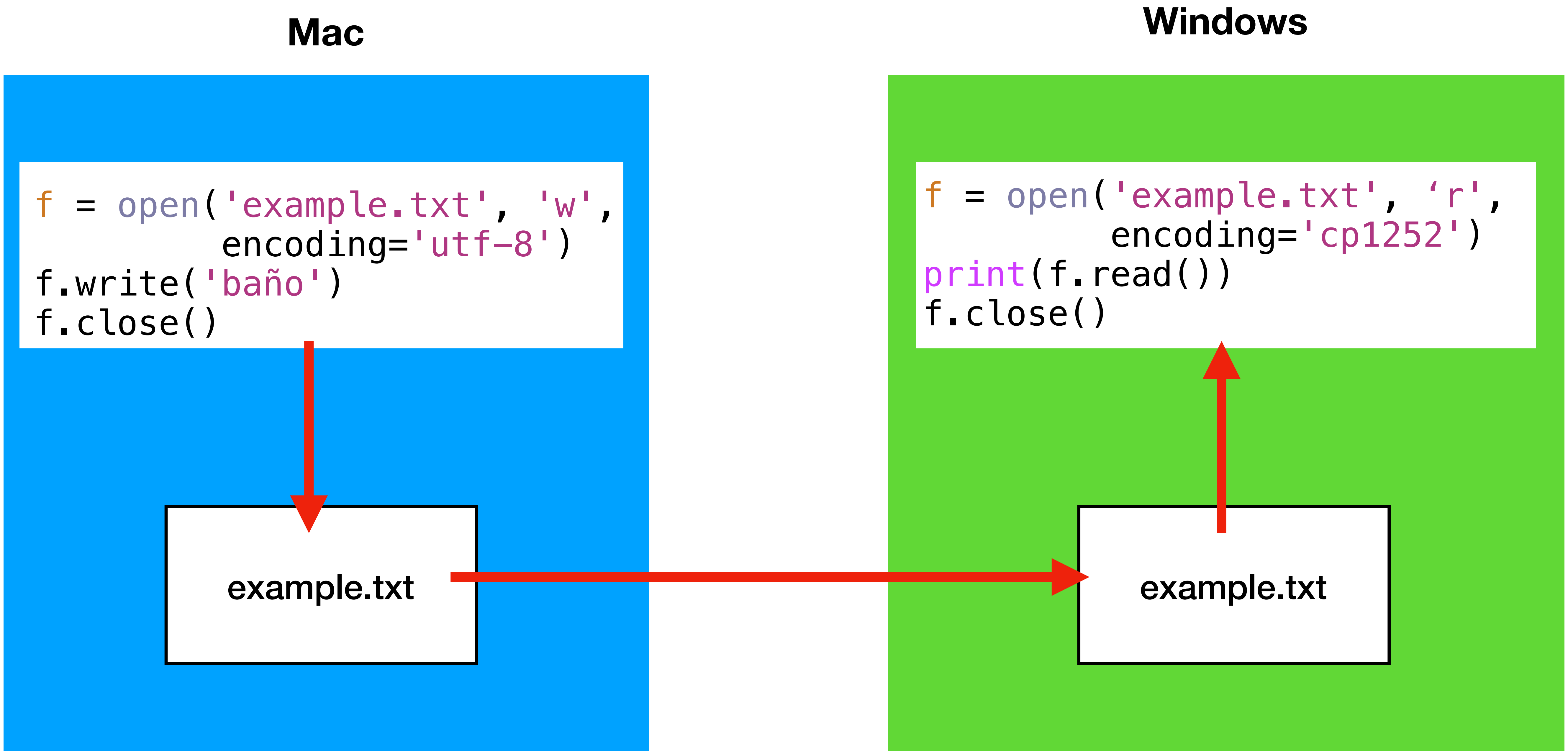
Mac

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

Windows

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

Encoding Defaults Done Wrong

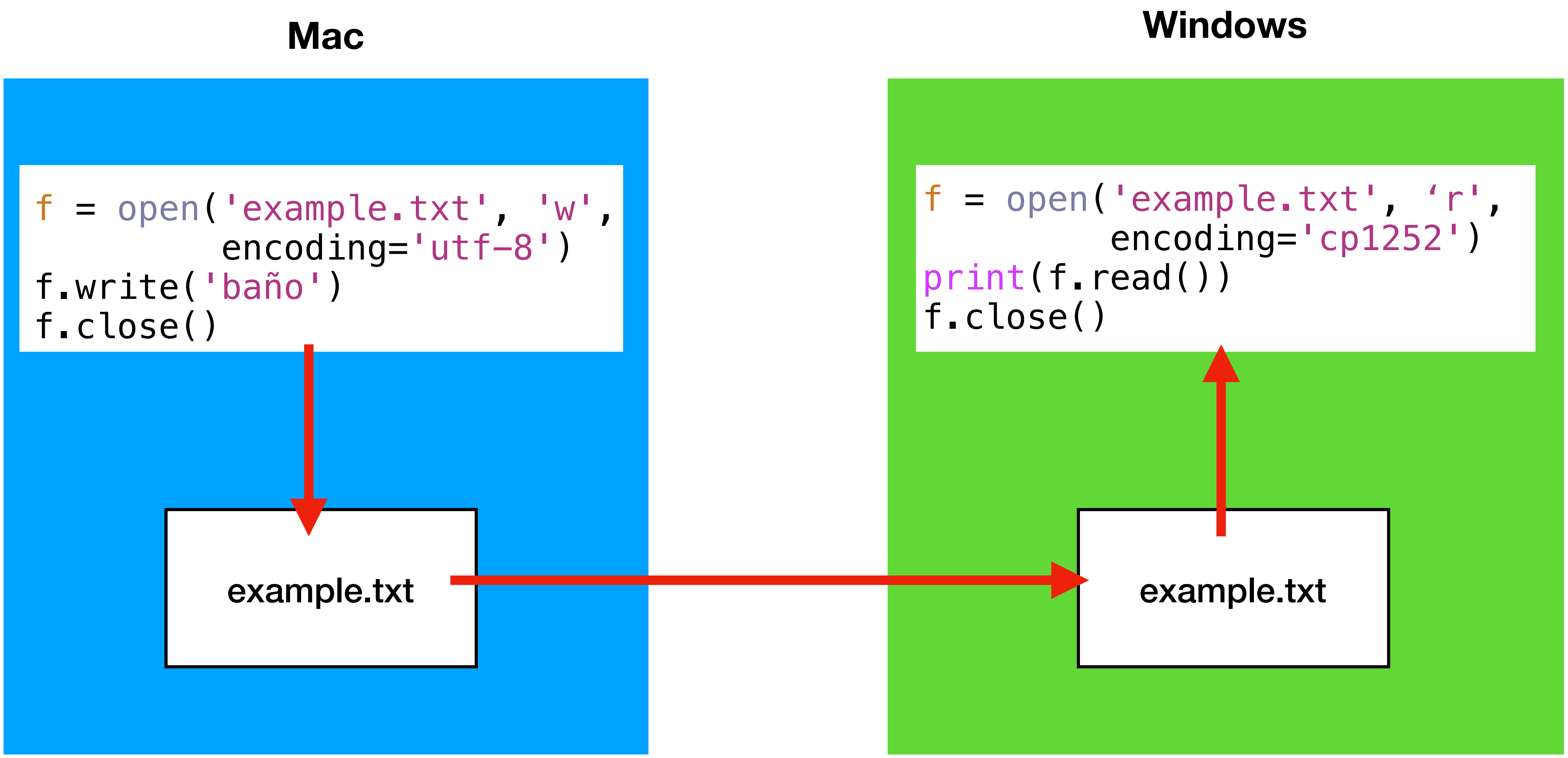


```
f = open('example.txt', 'w',
         encoding='utf-8')
f.write('baño')
f.close()
```

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

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

Encoding Defaults Done Wrong



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

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

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