Introduction to Computer Programming Lecture 6.1:

Reading and Writing Files, User Input

Hemma Philamore

Department of Engineering Mathematics

Reading/Writing files

Functions for reading and writing to external files:
open, read, write, close

Data structures can be **populated** based on data in another format (text file, database, etc.)

open(file_path, mode_specifier)

`r` : open an existing file to read

`w` : open an existing file to write to.
If no file exists: creates a new file.
If file exists : over-writes previous contents.

`a` : open an existing file to write to.
If no file exists: creates a new file.
If file exists : appends text to end of file.

`r+` : open a text file to read from **or** write to.
File must already exist.
If file exists : over-writes previous contents.

`w+` : open a text file to read from **or** write to.
If no file exists: creates a new file.
If file exists : over-writes previous contents.

`a+` : open a text file to read from **or** write to.
If no file exists: creates a new file.
If file exists : appends text to end of file.

Why do we need to close a file?

1. Not automatically closed.
2. Risk of overwriting.
3. Saves changes to file.
4. Depending on OS, you may not be able to open a file simultaneously for reading and writing.

Example: If a program attempts to open a file that is already open (has not been closed), an error may be generated.

HighScore Table

The table should be saved to a file

1	John Mayer	550
2	Mary Loe	480
3	Tony Times	380
4	Kent Clear	305
5	Sara Knight	150

Writing Files

```
text = "Hemma 33"

file = open("scores.txt", "w")

file.write(text + "\n")

file.close()
```

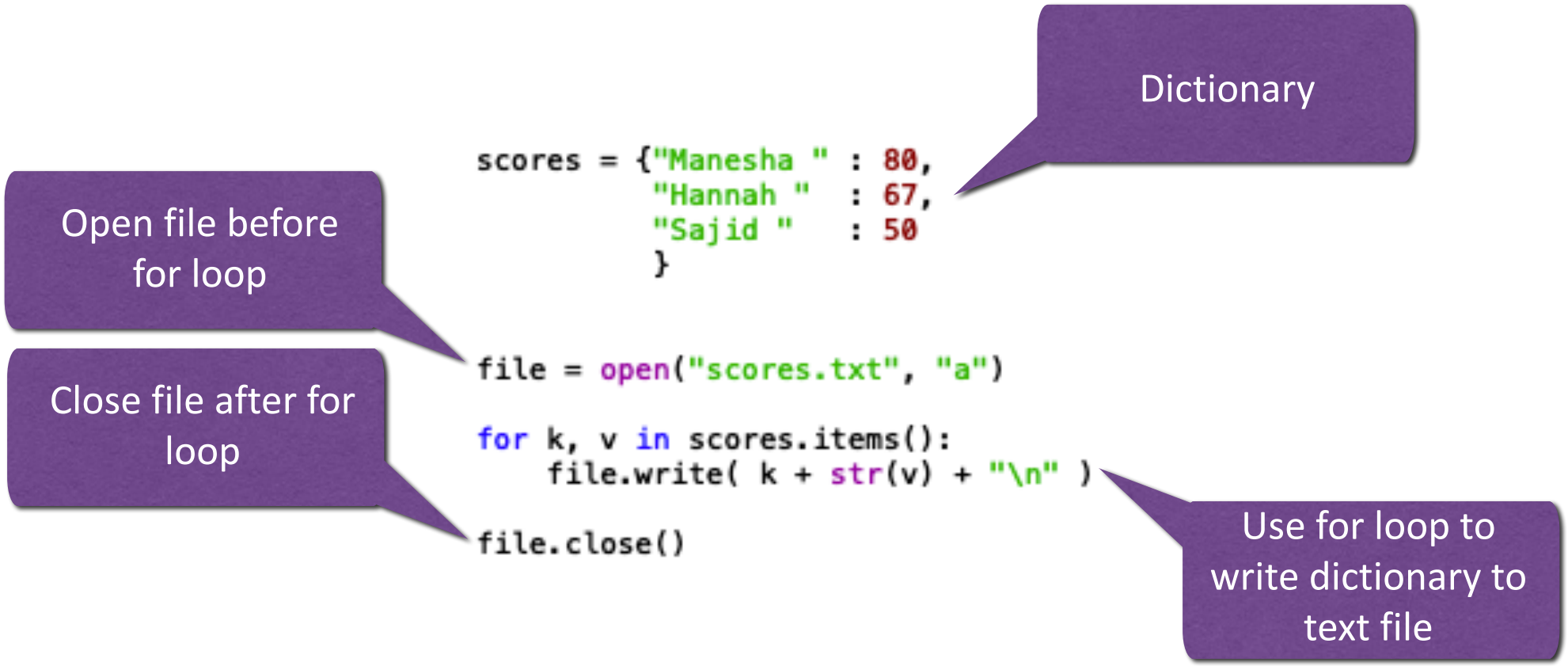
Mode specifier to write

```
text = "Farhad 44"

file = open("scores.txt", "a")

file.write(text + "\n")

file.close()
```

Mode specifier to append

Open the file using a text editor to confirm it's contents.

Writing Files



Open the file using a text editor to confirm it's contents.

Reading Files

Open with mode specifier “r”

Open returns an iterable object. Use for loop to iterate

```
file = open("scores.txt", "r")

for line in file:
    i = line.split()
    names.append( i[0] )
    scores.append( int(i[1]) )

file.close()
```

Remember to close the file!

Reading Files

```
file = open("scores.txt", "r")

names = []
scores = []

for line in file:
    i = line.split()
    names.append( i[0] )
    scores.append( int(i[1]) )

file.close()
```

Split divides items separated by spaces into string items of list. Alternative delimiter can be selected

```
index = scores.index( (max(scores)) )

print("highest score player: " +
      names[index] +
      "\n player scored " + str(scores[index])
      )
```

Finding the highest scoring player

Automatically close files using **with**

```
file = open("scores.txt", "a")

file.write("Lisa 50")

file.close()
```

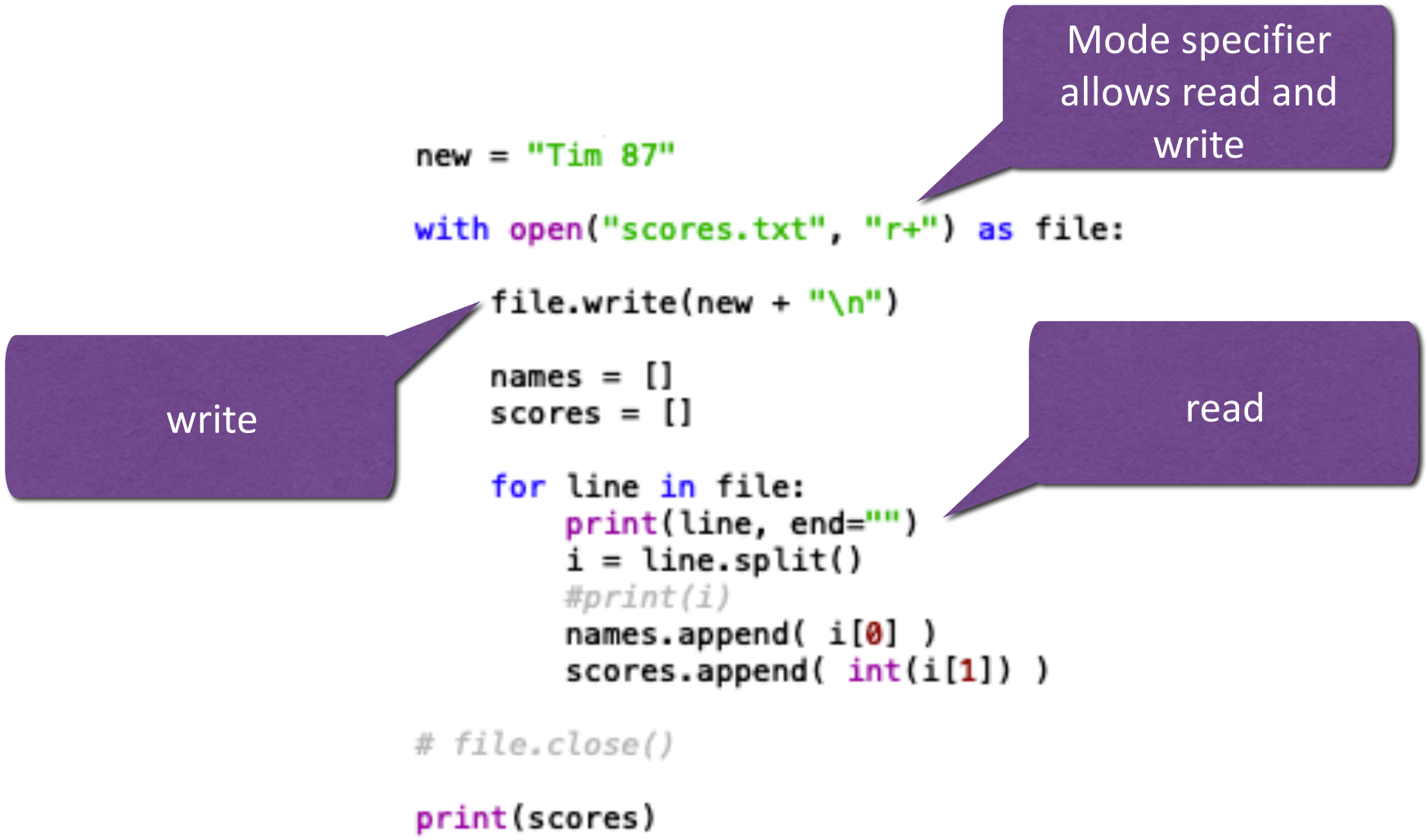
It can be difficult to remember to close the file

with open opens the file…

…it is closed again automatically when the code unindents

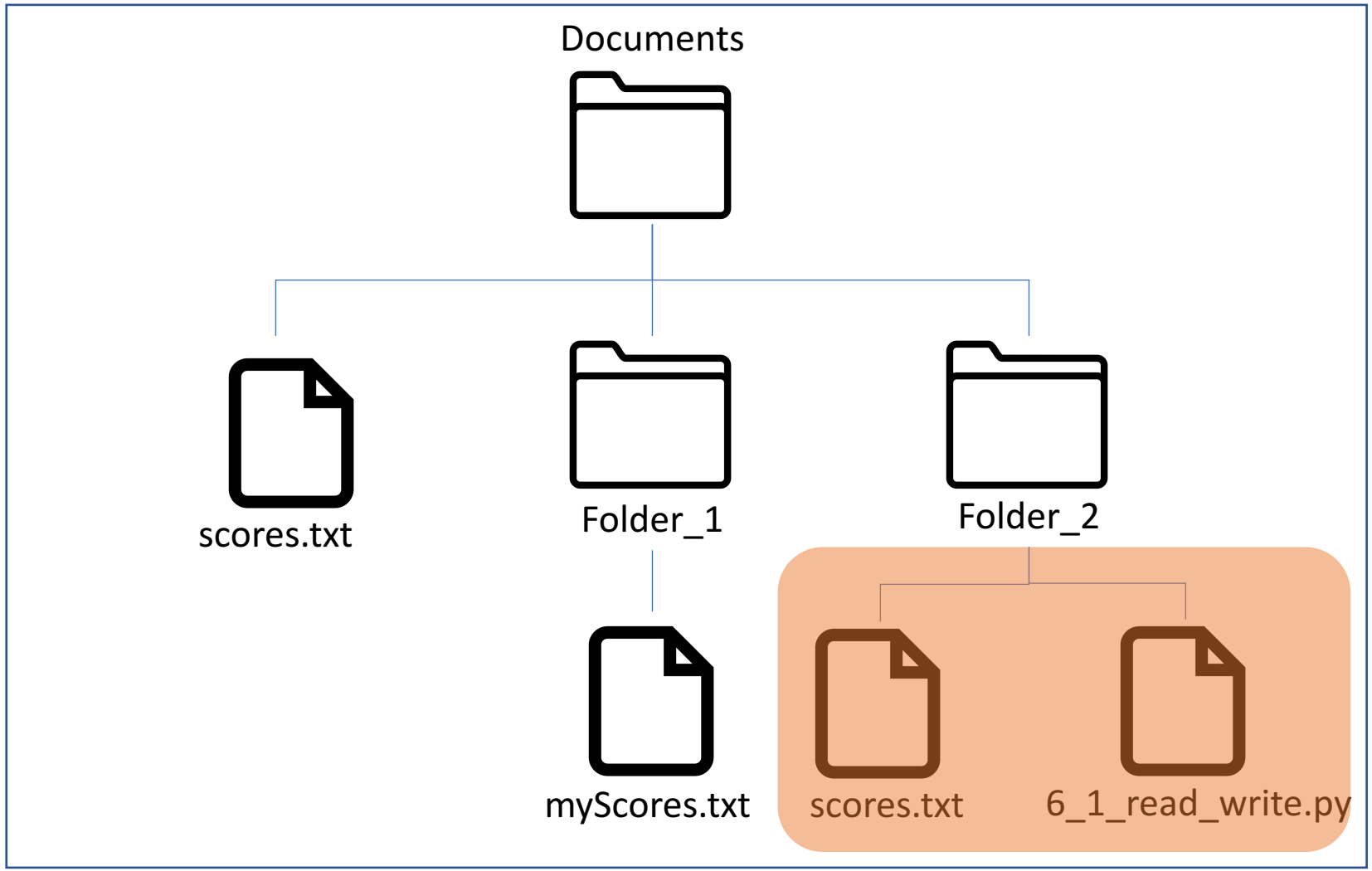
```
with open("scores.txt", "a") as file:
    file.write("Lisa 50" + "\n")
    file.write("Zoe 50" + "\n")
    file.write("Ben 50" + "\n")

a = 10
```

Read and write



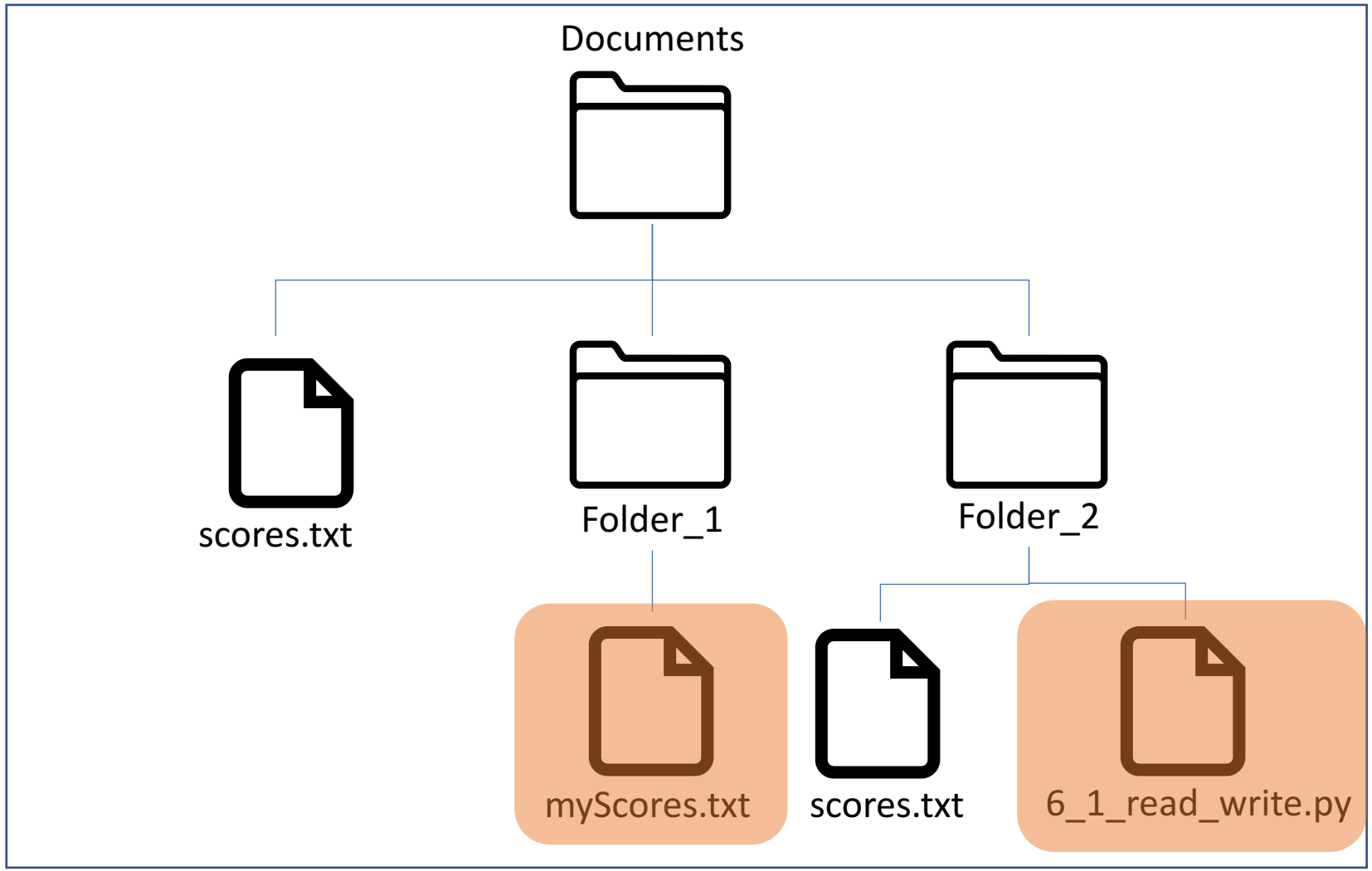
open(**file_path**, mode_specifier)

```
open("scores.txt", w)
```



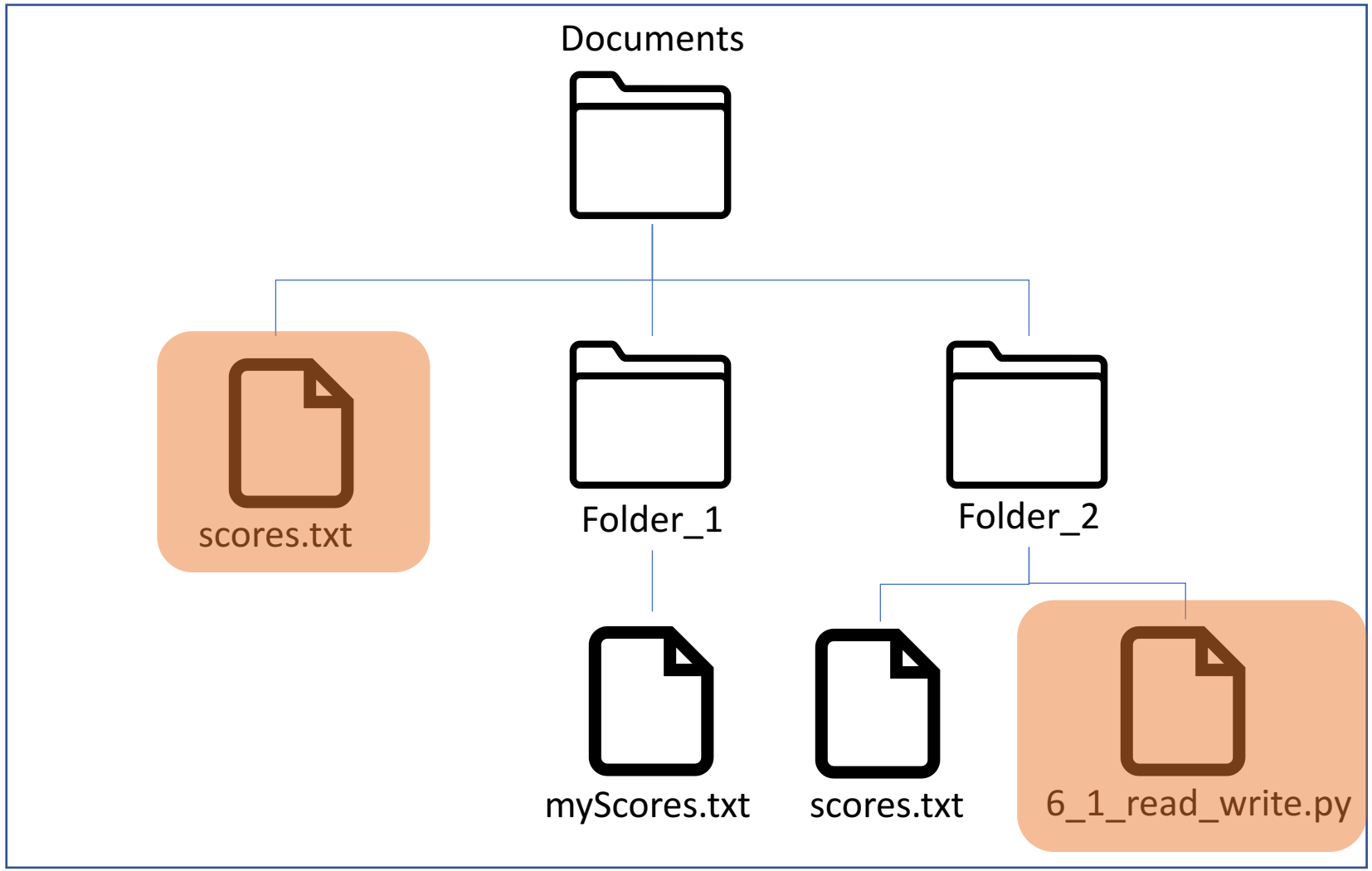
open(**file_path**, mode_specifier)

```
open( "../Folder_1/myScores.txt", w)
```



open(**file_path**, mode_specifier)

```
open("../scores.txt", w)
```



Input

Variable to store the user input

Prompt shown to the user

```
1  score = 15
2  name = input("Enter player name: ")
3
4  with open("scores.txt", "a") as file:
5      file.write(name + " " + str(score) + "\n")
```

```
1  user_input = input("enter 3 words : ")
2  split_input = user_input.split()
3  split_input
```

```
enter 3 words : I am here

['I', 'am', 'here']
```

Delimited values returned as a list