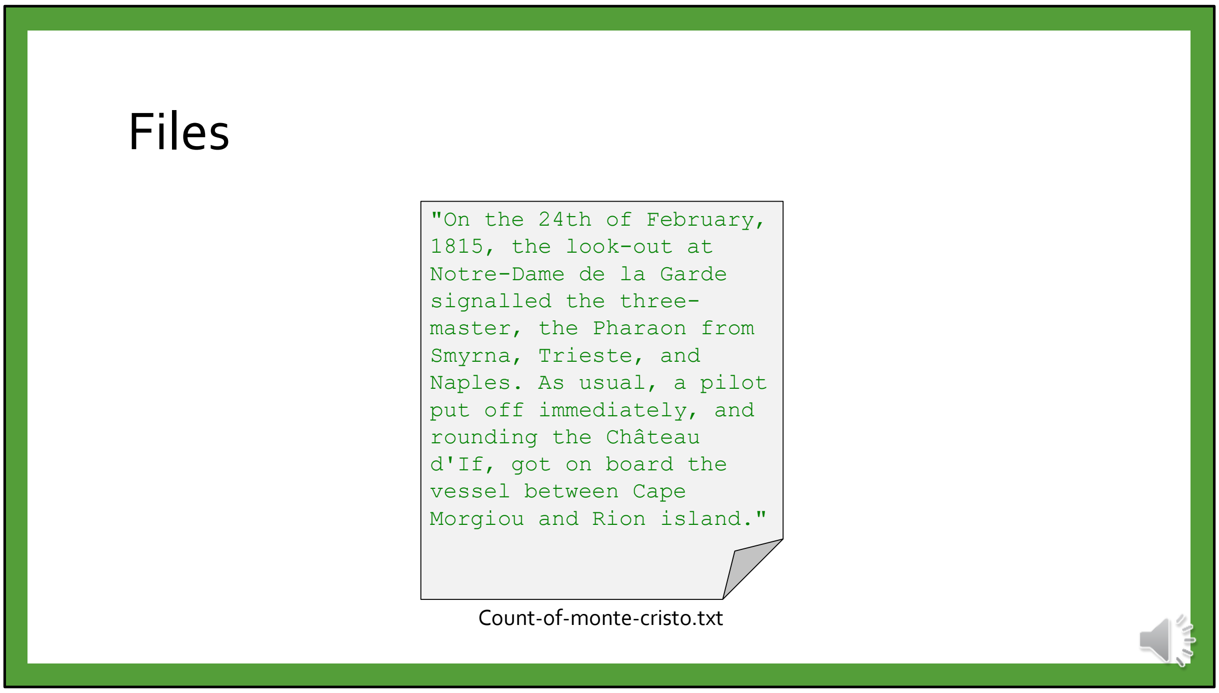
FILES

An Introduction to Computer Science

Let's learn about Files

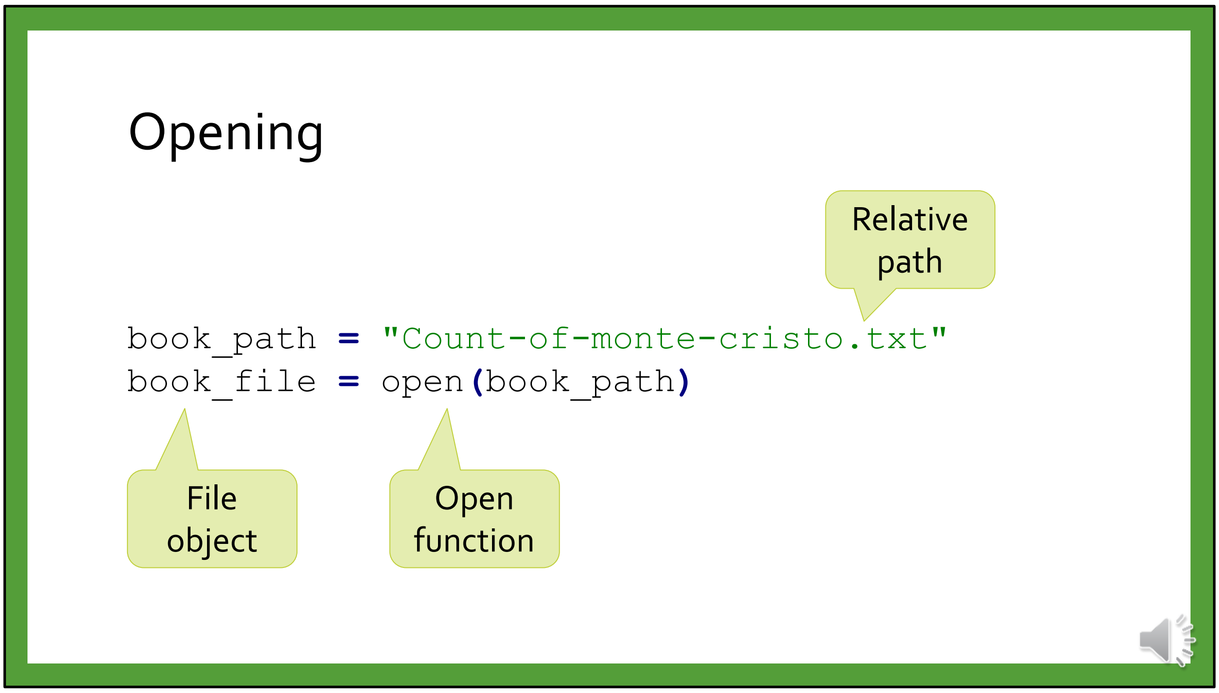


Files

"On the 24th of February, 1815, the look-out at Notre-Dame de la Garde signalled the three-master, the Pharaon from Smyrna, Trieste, and Naples. As usual, a pilot put off immediately, and rounding the Château d'If, got on board the vessel between Cape Morgiou and Rion island."

Count-of-monte-cristo.txt

You can think of a File as a string of data.
If we know the path and filename of the File, we can use Python to get access to it.



Opening

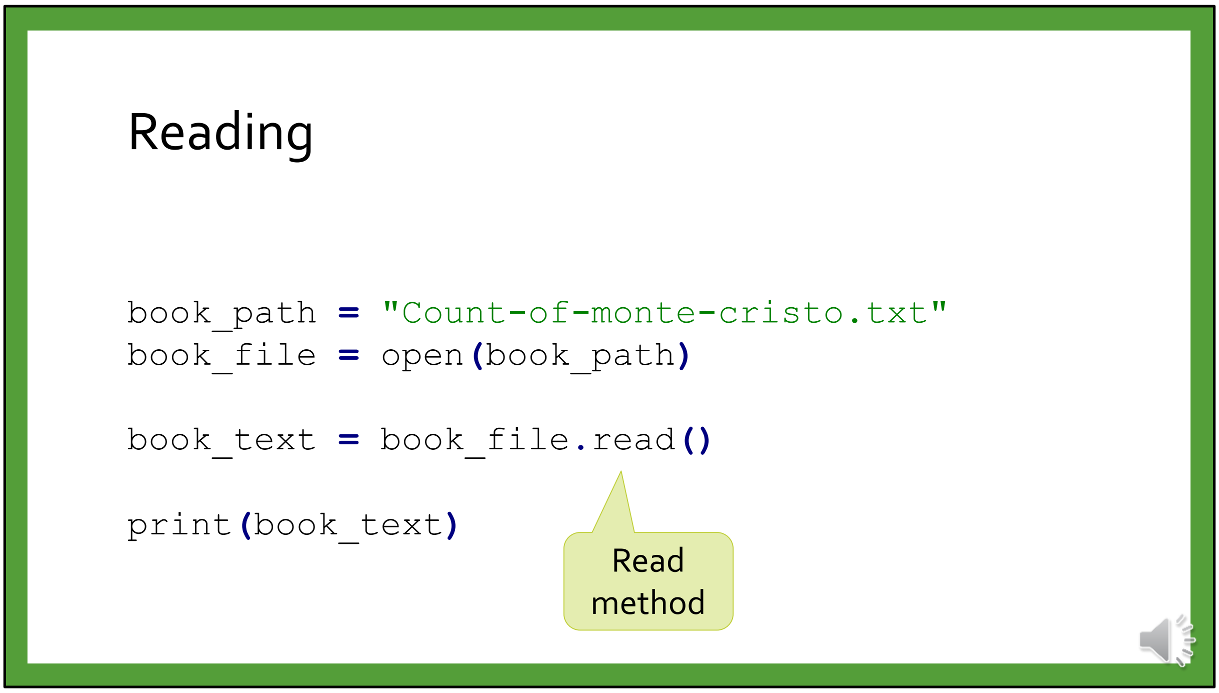
Relative path

```
book_path = "Count-of-monte-cristo.txt"
book_file = open(book_path)
```

File object

Open function

Before you can access a file, you must explicitly open the file using the "open" function. This function consumes the path to the file as a string and returns a file object. Typically, we store this file object into a variable.



Reading

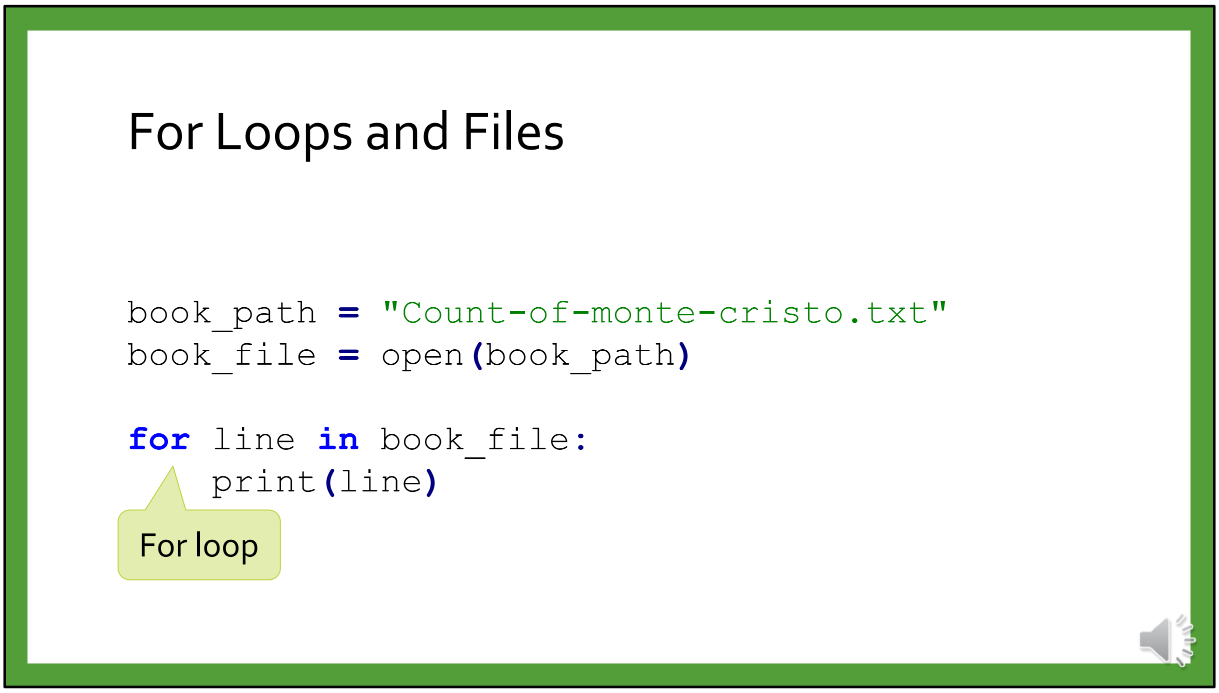
```
book_path = "Count-of-monte-cristo.txt"
book_file = open(book_path)

book_text = book_file.read()

print(book_text)
```

Read method

The primary way to get data from a file is to use the ".read()" method.
This simply returns the contents of the file as a single string.
Check out this example, where we open the file, read the file handle, and then print the text.
Notice how this is a multi-step process: we use the path to open the file, and then we read from that open file.



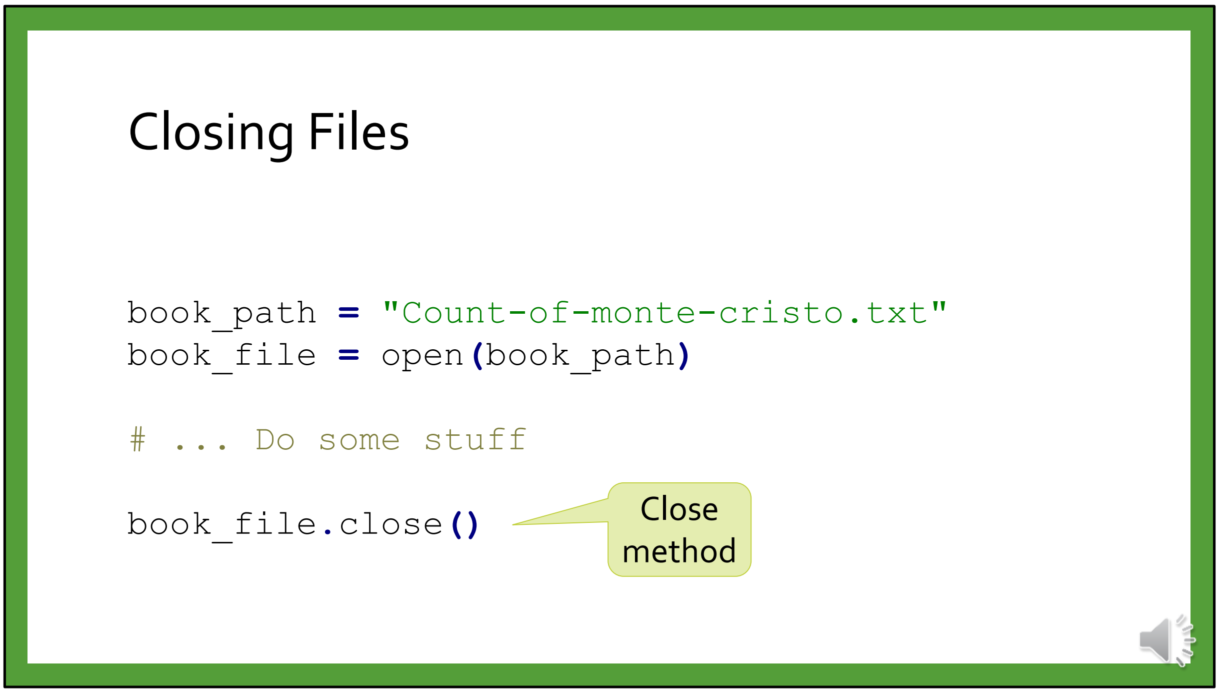
For Loops and Files

```
book_path = "Count-of-monte-cristo.txt"
book_file = open(book_path)

for line in book_file:
    print(line)
```

For loop

Often, we want to process a file line-by-line.
Because a File is actually a sequence of strings (each separated by a new line), we can process it using a For loop quite easily.
The example shown will process the file line-by-line, which would be perfect if we wanted to manipulate each line of the file, perhaps to adjust capitalization or convert it to a number.
Notice that with the for loop, we no longer need to use the read method.



Closing Files

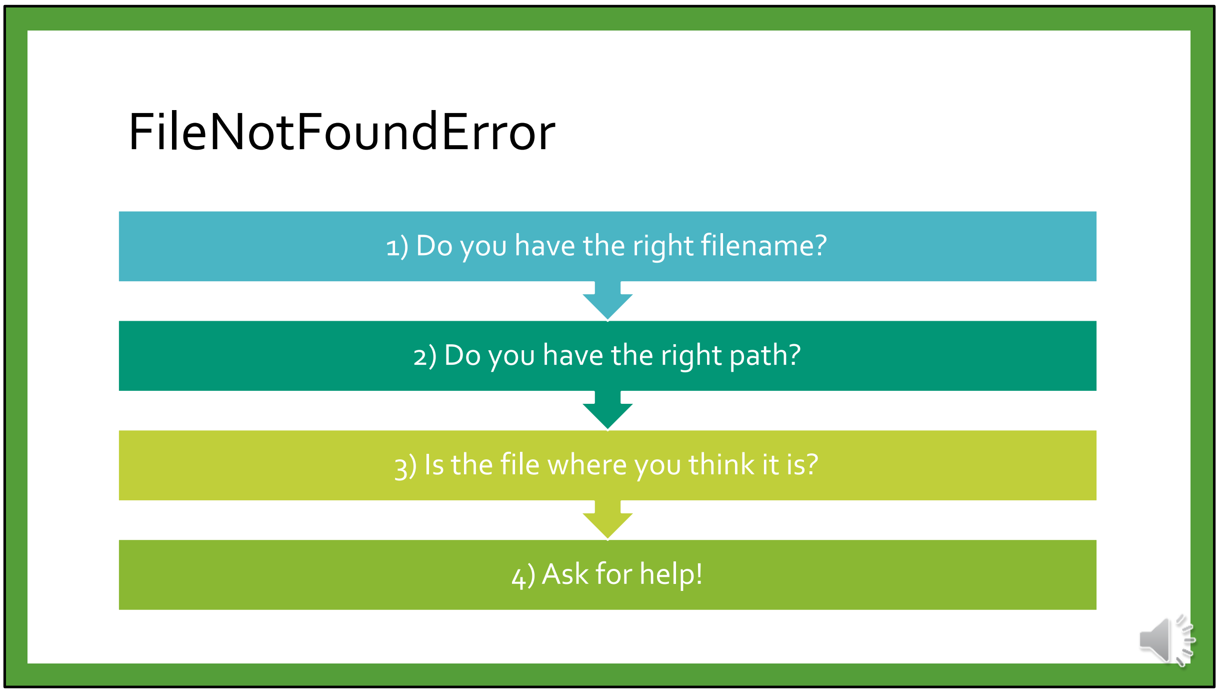
```
book_path = "Count-of-monte-cristo.txt"
book_file = open(book_path)

# ... Do some stuff

book_file.close()
```

Close method

When you are done with a file, you should always remember to close it using the "close" method.
It's like leaving a house: if you open a door, you need to close the door.



FileNotFoundError

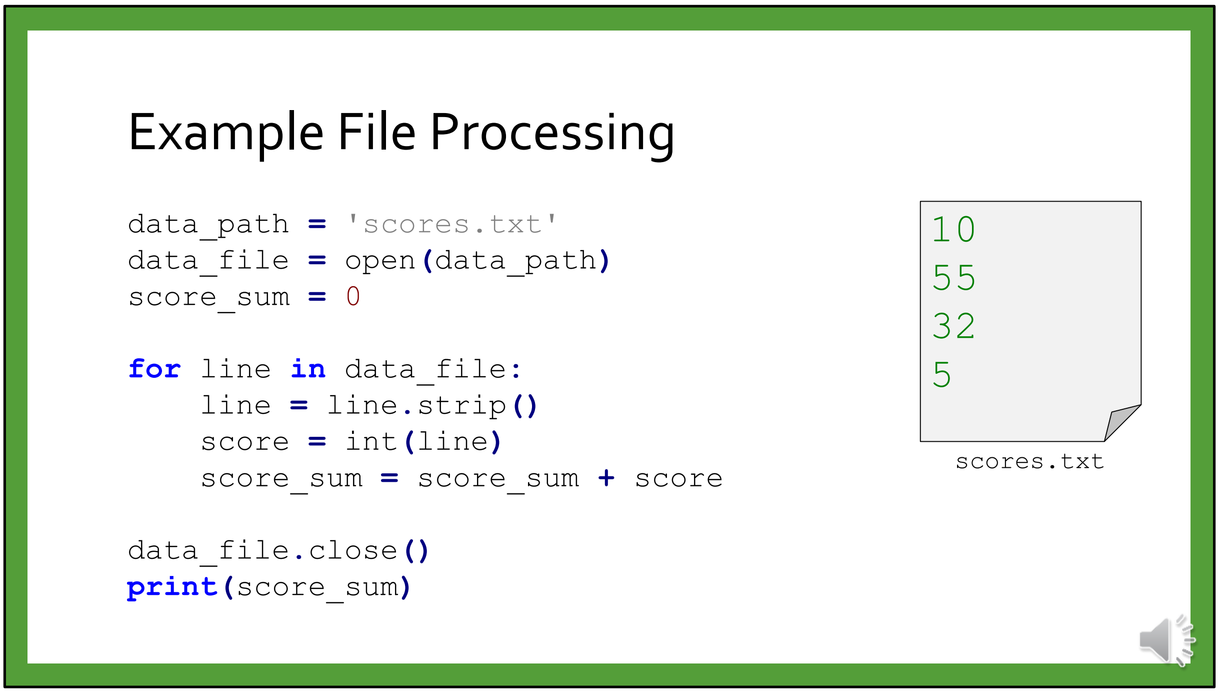
1) Do you have the right filename?

2) Do you have the right path?

3) Is the file where you think it is?

4) Ask for help!

Filesystems are tricky, because everyone has a different setup.
Often, we misplace files.
When we try to open a file that does not exist, Python raises an FileNotFoundError and will suggest the file does not exist.
Typically, you should check that you know the exact filename, that you are using the path, and that you know where your Python source file is relative to the file you are opening.



Example File Processing

```
data_path = 'scores.txt'
data_file = open(data_path)
score_sum = 0

for line in data_file:
    line = line.strip()
    score = int(line)
    score_sum = score_sum + score

data_file.close()
print(score_sum)
```

```
10
55
32
5
```

scores.txt

Processing books is one possible application for files, but often we want to process a file full of numeric data.
Here, we see some example code that will process a list of numbers in a file, each of which represents a score.
We are also using the Sum pattern we learned about before to add each of those scores together.
Notice how we must strip off the new lines at the end of each line, and then convert that line to a number; when we read data from a file, it comes in as a string.