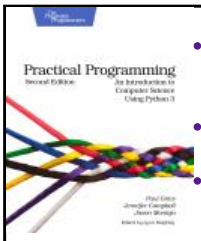
Reading Files



- Chapter 10.1 What Kinds of Files Are There?
- Chapter 10.2 Opening a File
- Chapter 10.3 Techniques for Reading Files

Optional reading

Information stored in files can be accessed by a Python program. To get access to the contents of a file, you need to *open* the file in your program. When you are done using a file, you should *close* it.

Opening and Closing A File

Python has a built-in function `open` that can open a file for reading.

The form of `open` is `open(filename, mode)`, where mode is `'r'` (to open for reading), `'w'` (to open for writing), or `'a'` (to open for appending to what is already in the file).

This opens a file called `In Flanders Fields.txt` for reading:

```
flanders_file = open('In Flanders Fields.txt', 'r')
```

Note that if the file is saved in the same directory as your program, you can simply write the name of the file, as what was done in the above example. However, if it is not saved in the same directory, you must provide the path to it.

To close a file, you write `flanders_file.close()` .

There are four standard ways to read from a file. Some use these methods:

`readline()`: read and return the next line from the file, including the newline character (if it exists). Return the empty string if there are no more lines in the file.

`readlines()`: read and return all lines in a file in a list. The lines include the newline character.

`read()`: read the whole file as a single string.

Approach	Code	When to use it

The `readline` approach	`file = open(filename, 'r')`
	
	`# Read lines until we reach the`
	`# place in the file that we want.`
	`line = file.readline()`
	`while` *`we are not at the place we want`*`:`
	`    line = file.readline()`
	
	`# Now we have reached the section`
	`# of the file we want to process.`
	
	`line = file.readline()`
	`while` *`we are not at the end of the section`*`:`
	`    process the line`
	`    line = file.readline()`
	
	`flanders_file.close()`
		When you want to process only part of a file.

The `for` *`line`* `in` *`file`* approach	`file = open(filename, 'r')`
	
	`for line in file:`
	`    process the line`
	
	`file.close()`
		When you want to process every line in the file one at a time.

The `read` approach	`file = open(filename, 'r')`
	
	`contents = file.read()`
	
	`now process` `contents`
	
	`file.close()`
		When you want to read the whole file at once and use it as a single string.

The `readlines` approach	`file = open(filename, 'r')`
	
	`# Get the contents as a list of strings.`
	`contents_list = file.readlines()`
	
	`process` `contents_list` *`using indexing to access particular lines from the file`*
	
	`file.close()`
		When you want to examine each line of a file by index.

Examples from the video

Here are the code examples that appeared in the video. All of them read the entire file into a string and print that string.

The `readline` approach

```
flanders_file = open(flanders_filename, 'r')
flanders_poem = ''

line = flanders_file.readline()
while line != "":
    flanders_poem = flanders_poem + line
    line = flanders_file.readline()

print(flanders_poem)
flanders_file.close()
```

The `for` *`line`* `in` *`file`* approach

```
flanders_file = open(flanders_filename, 'r')
flanders_poem = ''

for line in flanders_file:
    flanders_poem = flanders_poem + line

print(flanders_poem)
flanders_file.close()
```

The `read` approach

```
flanders_file = open(flanders_filename, 'r')
flanders_poem = flanders_file.read()

print(flanders_poem)
flanders_file.close()
```

The readlines approach

```
flanders_file = open(flanders_filename, 'r')
flanders_poem = ''

flanders_list = flanders_file.readlines()
for line in flanders_list:
    flanders_poem = flanders_poem + line

print(flanders_poem)
flanders_file.close()
```

Jennifer Campbell • Paul Gries
University of Toronto