CoGrammar

Lecture 6: File IO Operations

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

Lecture Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(FBV: Mutual Respect.)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Open Classes. You can submit these questions here: **Open Class Questions**

Lecture Housekeeping cont.

- For all **non-academic questions**, please submit a query: **www.hyperiondev.com/support**
- Report a **safeguarding** incident: **www.hyperiondev.com/safeguardreporting**
- We would love your **feedback** on lectures: **Feedback on Lectures**

Lecture Objectives

- Understand the way files are arranged in our PC
- Learn how to implement file read and write operations in Python

Introduction

- ★ Variables are a fine way to store data while your program is running, but if you want your data to persist even after your program has finished, you need to save it to a file.
- ★ You can think of a file's contents as a single string value, potentially gigabytes in size.

Files and File Paths

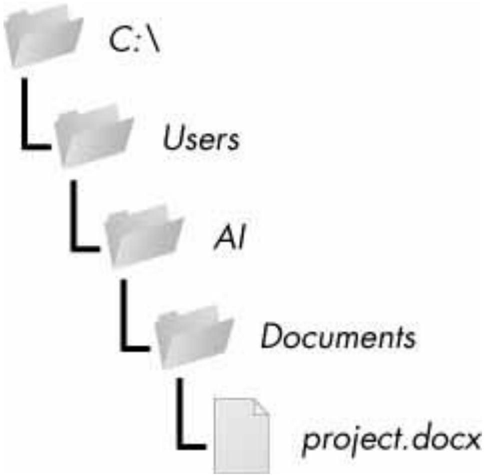
- A file has two key properties: a filename (usually written as one word) and a path.

- The path specifies the location of a file on the computer. For example, there is a file on my Windows laptop with the filename ***project.docx*** in the path ***C:\Users\Al\Documents***.

- The part of the filename after the **last period** is called the file's extension and tells you a file's type.

Files and File Paths

★ **Folders can contain files and other folders. For example, *project.docx* is in the *Documents* folder, which is inside the *AI* folder, which is inside the *Users* folder.**

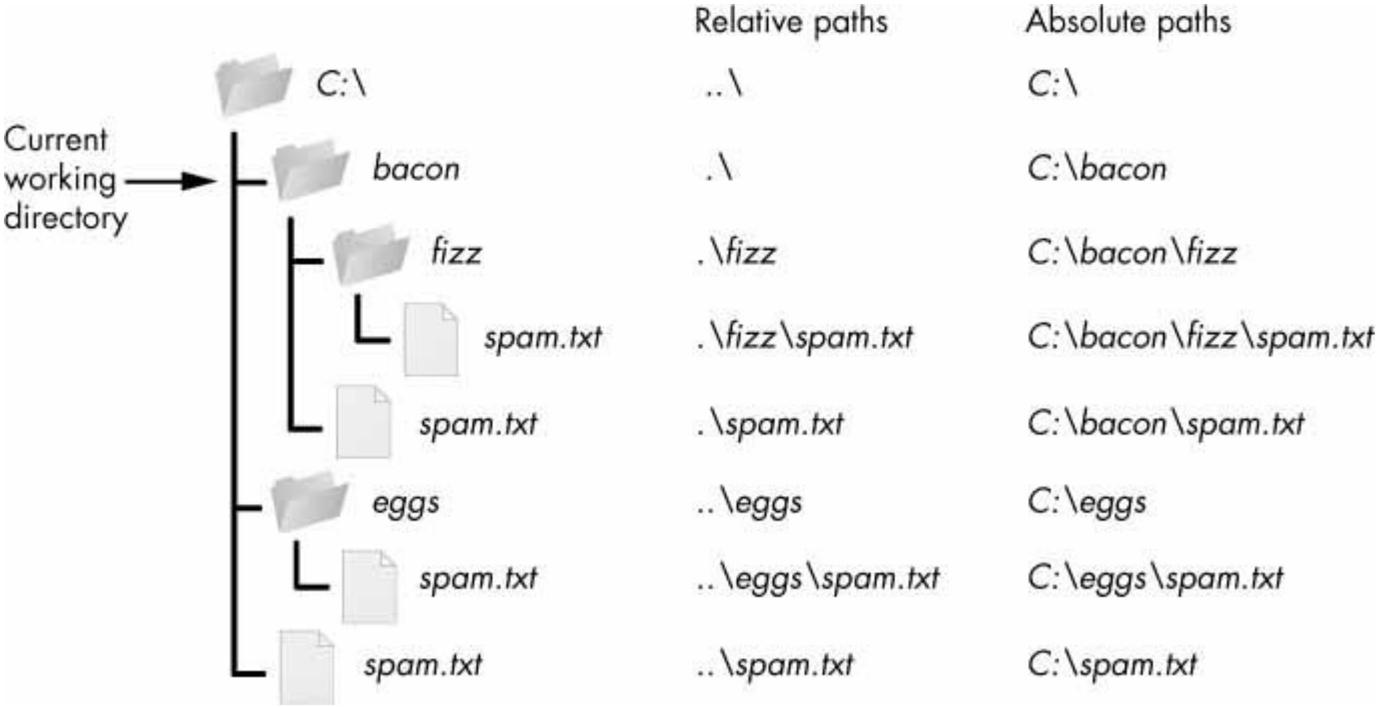


C:\
Users
AI
Documents
project.docx

Absolute vs. Relative Paths

- There are two ways to specify a file path:
  - An absolute path, which always begins with the root folder
  - A relative path, which is relative to the program’s current working directory

Absolute vs. Relative Paths



	Relative paths	Absolute paths
C:\	..\	C:\
bacon	.\	C:\bacon
fizz	.\fizz	C:\bacon\fizz
spam.txt	.\fizz\spam.txt	C:\bacon\fizz\spam.txt
spam.txt	.\spam.txt	C:\bacon\spam.txt
eggs	..\eggs	C:\eggs
spam.txt	..\eggs\spam.txt	C:\eggs\spam.txt
spam.txt	..\spam.txt	C:\spam.txt

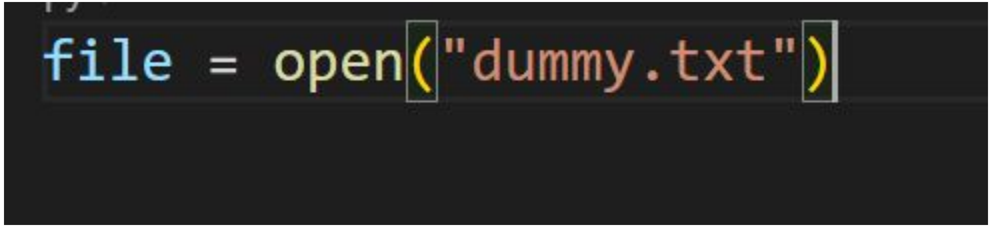
Opening Files in Python

- To open a file with the open() function, you pass it a string path indicating the file you want to open; it can be either an absolute or relative path. The open() function returns a File object.
- When a file is opened in read mode, Python lets you only read data from the file; you can’t write or modify it in any way. Read mode is the default mode for files you open in Python. But if you don’t want to rely on Python’s defaults, you can explicitly specify the mode by passing the string value 'r' as a second argument to open().

Opening Files in Python

★ In Python, we need to open a file first to perform any operations on it—we use the open() function to do so.

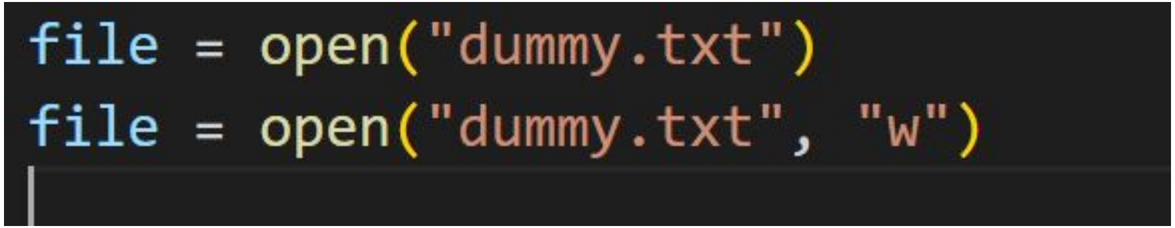
★ The call to open() returns a File object. A File object represents a file on your computer; it is simply another type of value in Python.



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

File Opening Modes

★ Python allows us to open files in different modes (read, write, append, etc.), based on which we can perform different file operations. By default, Python files are open in read mode.



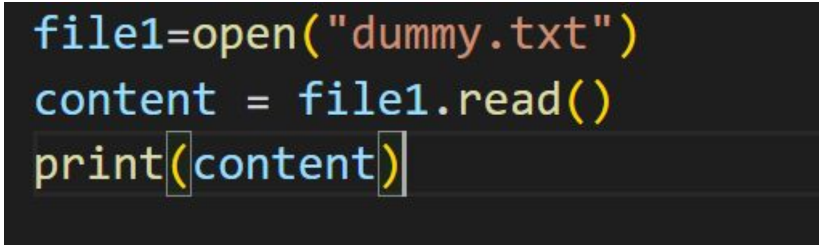
```python
file = open("dummy.txt")
file = open("dummy.txt", "w")
```

File Opening Modes

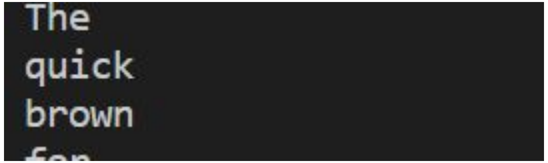
`r`	Open a file in reading mode (default)
`w`	Open a file in writing mode
`x`	Open a file for exclusive creation
`a`	Open a file in appending mode (adds content at the end of the file)
`t`	Open a file in text mode (default)
`b`	Open a file in binary mode
`+`	Open a file in both read and write mode

Reading Files

- ★ If you want to read the entire contents of a file as a string value, use the File object’s read() method.
- ★ If you think of the contents of a file as a single large string value, the read() method returns the string that is stored in the file.



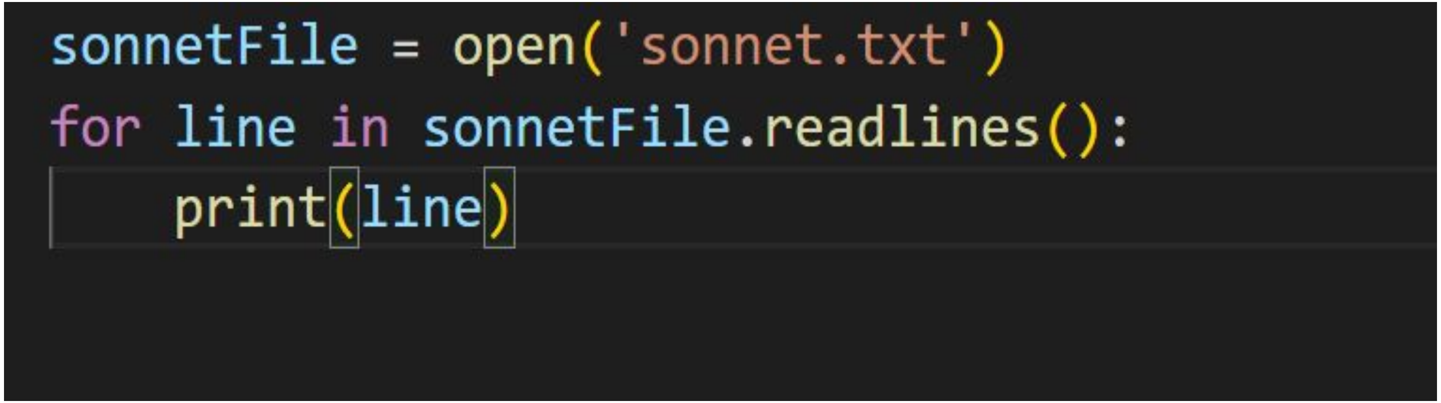
```
file1=open("dummy.txt")
content = file1.read()
print(content)
```



```
The
quick
brown
fox
```

Reading Files

★ Alternatively, you can use the readlines() method to get a list of string values from the file, one string for each line of text.



```python
sonnetFile = open('sonnet.txt')
for line in sonnetFile.readlines():
    print(line)
```

How do we open a file in Python?

A. Right click -> Open
B. Double-click
C. open()
D. Ask batman

What is a relative path relative to?

A. Current working directory
B. You
C. Me

Which of these is NOT a mode argument that can be passed to the open() function?

A. w
B. r
C. a
D. p

Let’s Breathe

Let’s take a small break before moving on to the next topic.

Writing to Files

★ Python allows you to write content to a file in a way similar to how the print() function “writes” strings to the screen.

★ You can’t write to a file you’ve opened in read mode, though. Instead, you need to open it in “write plaintext” mode or “append plaintext” mode, or write mode and append mode for short.

★ Write mode will overwrite the existing file and start from scratch, just like when you overwrite a variable’s value with a new value.

Writing to Files

- Pass *'w'* as the second argument to open() to open the file in write mode.

- Append mode, on the other hand, will append text to the end of the existing file. You can think of this as appending to a list in a variable, rather than overwriting the variable altogether. Pass 'a' as the second argument to open() to open the file in append mode.

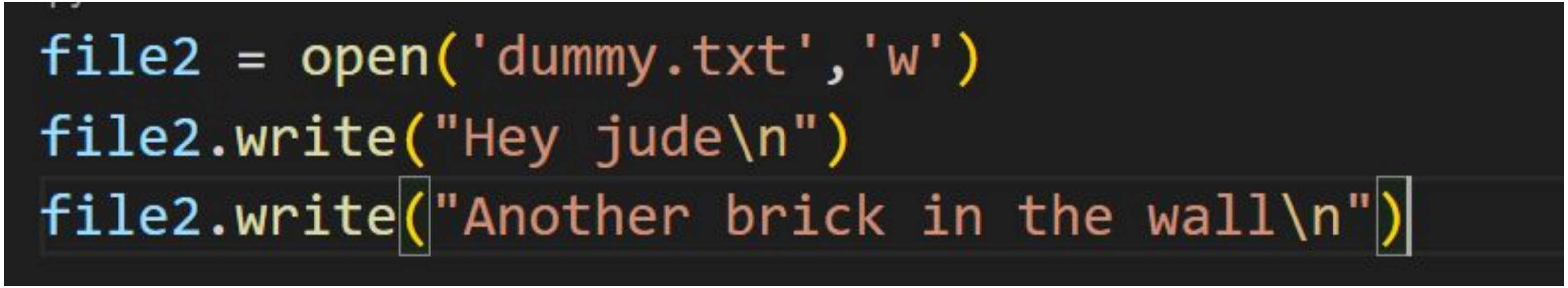
Writing to Files

★ If the filename passed to open() does not exist, both write and append mode will create a new, blank file.

★ Note that the write() method does not automatically add a newline character to the end of the string like the print() function does.

Writing to Files

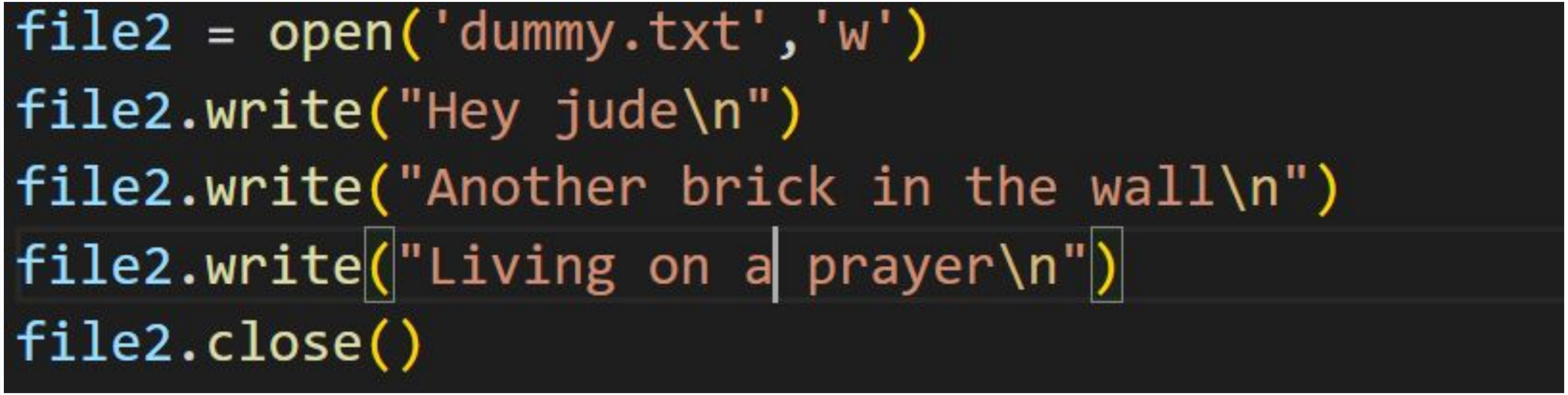
- To write to a Python file, we need to open it in write mode using the w parameter.
- Remember: If you try to perform the write operation to a file that already has some content, the new content will replace the existing ones.



```
file2 = open('dummy.txt','w')
file2.write("Hey jude\n")
file2.write("Another brick in the wall\n")
```

Closing Files

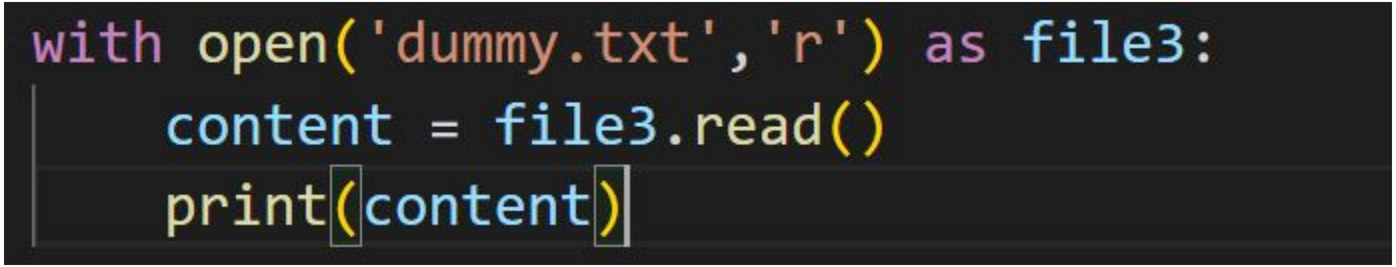
★ When we are done performing operations on the file, we need to close the file properly. We use the close() function to close a file in Python.



```
file2 = open('dummy.txt','w')
file2.write("Hey jude\n")
file2.write("Another brick in the wall\n")
file2.write("Living on a prayer\n")
file2.close()
```

With....open Syntax

- ★ In Python, there is a better way to open a file using with...open.
- ★ Automatically closes the file, so we don't have to use the close() function.



```python
with open('dummy.txt','r') as file3:
    content = file3.read()
    print(content)
```

What happens if an existing file is opened in write mode?

A. Previous content is wiped out
B. Nothing happens
C. A python comes to my house
D. New content is appended to the file

How do we close a file in Python?

A. Click X
B. .close()
C. Throw your pc in the trash bin
D. New content is appended to the file

Which mode argument can be passed to the open() function to open a file in append mode?

A. w
B. r
C. a
D. p

CoGrammar

Q & A SECTION

Please use this time to ask any questions relating to the topic, should you have any.

CoGrammar

Thank you for joining!

CoGrammar