



Welcome to this CoGrammar Lecture: Text File I/O

The session will start shortly...

Questions? Drop them in the chat.
We'll have dedicated moderators
answering questions.



Software Engineering Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
(Fundamental British Values: Mutual Respect and Tolerance)
- 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 Academic Sessions. You can submit these questions here: [Questions](#)

Software Engineering Session 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](#)

Skills Bootcamp

8-Week Progression Overview

Fulfil 4 Criteria to Graduation

✓ Criterion 1: Initial Requirements

Timeframe: First 2 Weeks

Guided Learning Hours (GLH):

Minimum of 15 hours

Task Completion: First four tasks

Due Date: 24 March 2024

✓ Criterion 2: Mid-Course Progress

60 Guided Learning Hours

Data Science - **13 tasks**

Software Engineering - **13 tasks**

Web Development - **13 tasks**

Due Date: 28 April 2024

Skills Bootcamp Progression Overview

✓ Criterion 3: Course Progress

Completion: All mandatory tasks,
including Build Your Brand and
resubmissions by study period end
Interview Invitation: Within 4 weeks
post-course
Guided Learning Hours: Minimum of
112 hours by support end date
(10.5 hours average, each week)

✓ Criterion 4: Demonstrating Employability

Final Job or Apprenticeship
Outcome: Document within 12
weeks post-graduation
Relevance: Progression to
employment or related
opportunity

CoGrammar Text File 10

March 2024

Agenda

- ❖ File I/O
 - Resource Management
 - Implicit File Handling
 - Explicit File Handling
 - Reading from Files
 - Writing to Files

What is File I/O ?

- ❖ **File I/O** stands for **Input/Output** operations involving files.
- ❖ It refers to the process of reading data from files (**input**) or writing data to files (**output**) using a computer program.


In simpler terms, file I/O is all about your program interacting with files: either taking in information from them or putting information into them. It's like the communication link between your program and the outside world of files.

File Handling

- ❖ **File modes** in Python are specifications used when opening a file to indicate the intended operation that will be performed on the file. These modes determine whether the file will be opened for reading, writing, appending, or for a combination of reading and writing
- ❖ **File handling** in Python refers to the process of working with files on your computer's storage. It allows you to read from, write to, and manipulate files. Python provides built-in functions and methods for performing various file operations.

File Modes

Table 1: Python File modes

Mode	Description
'r'	Opens a file for reading.
'w'	Open a file for writing. If file does not exist, it creates a new file. If file exists it truncates the file.
'a'	Open a file in append mode. If file does not exist, it creates a new file.
'+' 	Open a file for reading and writing (updating)

File Handling

- ❖ In Python, file I/O operations are performed using the **open()** function and various methods associated with file objects.
- ❖ **Opening Files:** The **open()** function is used to open a file and returns a file object. You specify the file name/path and the mode in which you want to open the file ('r' for reading, 'w' for writing, 'a' for appending, etc.).

Opening Files

*Relative or Absolute path to the file
(including the extension)*

```
open ( <file> , mode )
```

*A String (Character) indicates
what you want to do with that file*

BOARD

Resource Management

Implicit Method

- ❖ The **with** statement is used for resource management in Python.
- ❖ It ensures that resources are properly cleaned up after use, even if an error occurs.

```
with open('filename.txt', 'r') as file:  
    content = file.read()
```

Resource Management

Explicit Method

- ❖ The explicit way involves manually opening and closing files using the **open()** function for opening and the **close()** method for closing.

```
file = open('file.txt', 'r')  
content = file.read()  
file.close()
```


File Modes (r)

- ❖ **Reading from Text Files:** You can read text from a file using the **open()** function with the mode **'r'**

```
with open('filename.txt', 'r') as file:  
    content = file.read()
```

File Modes (w)

- ❖ **Writing to Text Files:** You can write text to a file using the **open()** function with the mode 'w'

```
with open('filename.txt', 'w') as file:  
    file.write("Hello, world!")
```

File Modes (a)

- ❖ **Appending to Text Files:** You can append text to an existing file using the **open()** function with the mode **'a'**

```
with open('filename.txt', 'a') as file:  
    file.write("\nThis is a new line.")
```

File Handling (Reading)

Read from a File Python Methods

read()
Reads the entire
contents of the
file and returns it
as a string.

readline()
Reads a single
line from the file
and returns it as
a string.

readlines()
Reads all lines
from the file and
returns them as
a list of strings.

File Handling (Writing)

Write to a File Python Methods

```
graph TD; A[Write to a File Python Methods] --> B[write()]; A --> C[writelines()];
```

write()

This method is used to write data to the file. It takes a string argument and adds it to the end of the file.

writelines()

This method writes a sequence of strings to the file. It takes a list of strings as an argument and writes each string to the file.

Let's take a short Break



Exception Handling



Exception Handling

- ❖ **Exception handling** in Python allows you to gracefully manage errors that may occur during program execution, including when working with files.
- ❖ It enables you to anticipate and respond to errors without crashing the program.

Try / Except

- ❖ You can wrap file I/O operations inside a try block and catch specific exceptions using except blocks.

```
try:
    with open('file.txt', 'r') as file:
        content = file.read()
    print(content)
except FileNotFoundError:
    print("File not found!")
except PermissionError:
    print("Permission denied to open the file!")
except IOError as e:
    print(f"An I/O error occurred: {e}")
except Exception as e:
    print(f"An unexpected error occurred: {e}")
```

Try / Except / Finally

- ❖ We can also use a **finally** block to ensure that certain cleanup actions are always performed, regardless of whether an exception occurred or not.

```
try:
    file = open('file.txt', 'r')
    content = file.read()
    print(content)
except FileNotFoundError:
    print("File not found!")
finally:
    file.close() # Ensure file is closed even if an exception occurs
```

Questions and Answers



Thank you for attending



Department
for Education

