



Welcome to this **Co**Grammar Lecture: Text File I/O

The session will start shortly...

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



CoGrammar Text File 10

September 2024

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 Progression Overview

To be eligible for a certificate of completion, students must fulfil three specific criteria. These criteria ensure a high standard of achievement and alignment with the requirements for the successful completion of a Skills Bootcamp.

✓ **Criterion 1 - Meeting Initial Requirements**

Criterion 1 involves specific achievements within the first two weeks of the program. To meet this criterion, students need to:

- Attend a minimum of 7-8 hours per week of guided learning (lectures, workshops, or mentor calls) within the initial two-week period, for a total minimum of **15 guided learning hours (GLH)**, by no later than **15 September 2024**.
- Successfully complete the Initial Assessment by the end of the first 14 days, by no later than **15 September 2024**.



Skills Bootcamp Progression Overview


✓ Criterion 2 - Demonstrating Mid-Course Progress

Criterion 2 involves demonstrating meaningful progress through the successful completion of tasks **within the first half** of the bootcamp.

To meet this criterion, students should:

- Complete **42 guided learning hours** and the first half of the assigned tasks by the end of week 7, no later than **20 October 2024**.





Skills Bootcamp Progression Overview

✓ Criterion 3 - Demonstrating Post-Course Progress

Criterion 3 involves showcasing students' **progress after completing the course**. To meet this criterion, students should:

- Complete all mandatory tasks before the bootcamp's end date. This includes any necessary resubmissions, no later than **22 December 2024**.
- Achieve at least 84 guided learning hours by the end of the bootcamp, **22 December 2024**.



Poll

What is the output of the following code?

```
1 def modify_list(lst):  
2     lst.append(4)  
3     lst = [1, 2, 3]  
4     return lst  
5  
6 my_list = [5, 6, 7]  
7 result = modify_list(my_list)  
8 print(my_list)
```

- a. [5, 6, 7]
- b. [1, 2, 3]
- c. [5, 6, 7, 4, 1, 2, 3]
- d. [5, 6, 7, 4]

Poll

Given the following 2D list, what is the value of `matrix[1][2]`?

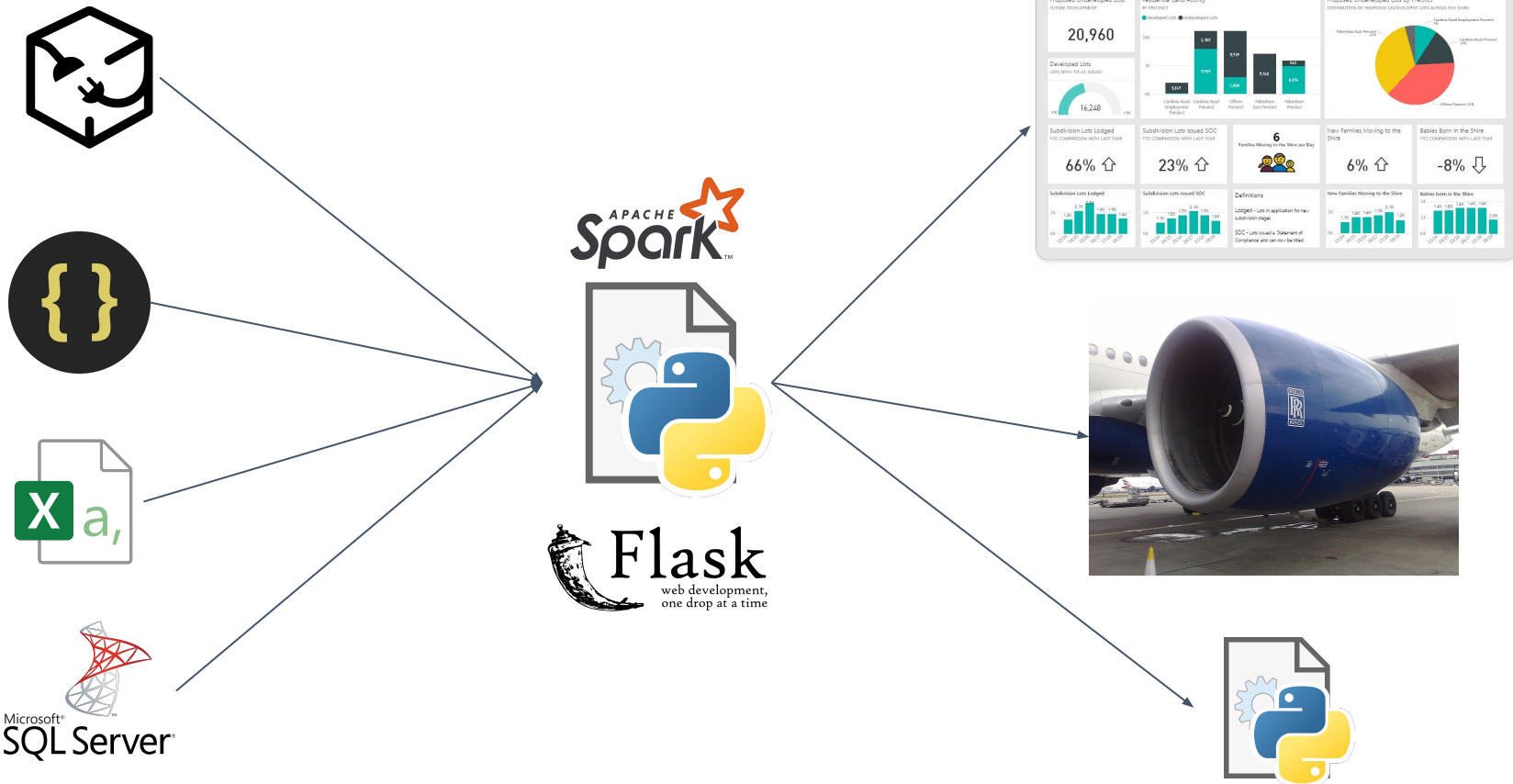
```
1 matrix = [  
2     [1, 2, 3],  
3     [4, 5, 6],  
4     [7, 8, 9]  
5 ]
```

- a. 2
- b. 6
- c. 5
- d. 8

Learning Objectives & Outcomes

- Demonstrate the basics of file handling in Python, including opening, reading, writing, and closing files.
- Perform common file operations such as reading from and writing to text files.
- Handle file-related exceptions to ensure robust file operations.
- Understand the importance of resource management in programming.
- Use the 'with' statement to manage file resources efficiently, ensuring files are properly closed after operations.
- Apply best practices for managing resources to prevent memory leaks and other resource-related issues.

Introduction



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

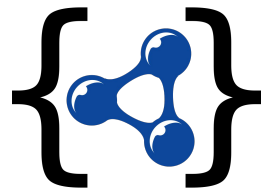
Understanding External Sources



They couldn't play soccer because they couldn't see the ball. They didn't want to go inside and play a game because it was a beautiful springtime night and they liked being ...[1]



```
id,name price
1,Laptop 1200
2,Smartphone 800
3,Headphones 200
4,Mouse 50
5,Monitor 1000
6,Desk 600
7,Speakers 400
```



```
{
  "track_id": "3"
  "name": "Inde"
  "artist": "Bucie"
  "album": "Inde"
  "duration_ms":
373000
  "popularity": 3
}
```

File Manipulations

- **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 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.).

Working with Files in Python: I/O

```
file = open("filename.txt", "access_mode")  
content = file.read()  
file.close()
```



opening

Read Only	r
Read and Write	r+
Write Only	w
Write and Read	w+
Append Only	a
Append and Read	a+

closing



Text file (.txt)

CoGrammar

File Access Modes

Must be closed
to avoid issues
like memory
leaks

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 Access 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 Access 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 Access 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

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

- An **Error/Exception** is an unexpected event that interrupts the normal execution of a computer program, preventing it from achieving its intended outcome.
- **Exception handling** in Python allows you to gracefully manage errors that may occur during program execution, including when working with files.

File Exception Handling

IsADirectoryError

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

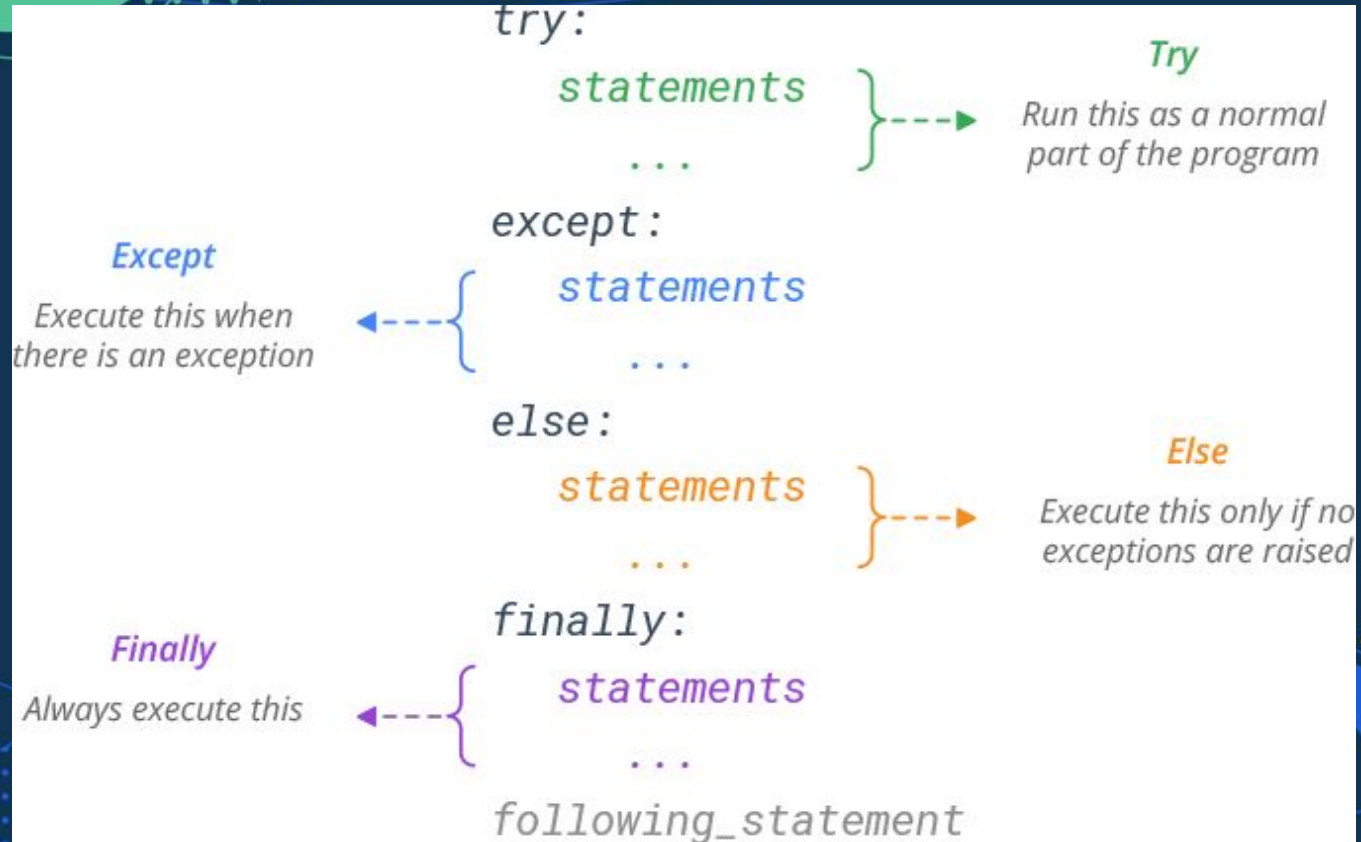
FileNotFoundError

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

PermissionError

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

Try / Except / Finally Structure



Try / Except Example

- 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
```

Poll

Given the following code, which lines ensure the file is properly closed after reading?

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

- a. with open('sample.txt', 'r') as file:
- b. content = file.read()
- c. file.close()
- d. The file is automatically closed when using 'with'

Poll

What is the output of the following code?

```
1  try:
2      with open('output.txt', 'a') as file:
3          file.write('Adding some text.')
4  except IOError:
5      print("An error occurred while performing file operations.")
```

- a. Creates a new file and writes 'Adding some text.' to it.
- b. Always raises an IOError exception.
- c. Appends 'Adding some text.' to output.txt.
- d. Prints a message if there's an issue with file operations.

Lesson Conclusion and Recap

- **File Operations**
 - Open/close, read/write files (text, CSV, JSON)
 - File modes: read, write, append
 - open(), read(), write(), with statement
- **Working with External Data Formats**
 - Handling text
- **Error Handling with Exceptions**
 - Importance of error handling
 - Common exceptions: FileNotFoundError, PermissionError, IsADirectory
 - try, except, else, finally blocks

Follow-up Activity

Task Overview:

- You will create a Python program that processes student grades stored in a text file. The program should read the grades, calculate the average, identify the highest and lowest grades, and write these results to a new text file.

Steps:

1. Create a Function to Read Data:
 - Write a function `read_grades(filename)` that reads grades from a text file, where each line contains a grade. Store these grades in a list.
2. Process the Data:
 - Write a function `process_grades(grades)` that:
 - Calculates the average grade from the provided ones
 - Identifies the highest and lowest grades from the provided ones
 - Returns these values.

Follow-up Activity

3. Write the Results to a New File:
 - Write a function `write_results(filename, average, highest, lowest)` that writes the average, highest, and lowest grades to a new text file.
4. Main Program:
 - Combine the functions in a main block:
 - Use `read_grades` to read the grades from a file named `grades.txt`.
 - Use `process_grades` to calculate the necessary statistics.
 - Use `write_results` to save the results in a file named `results.txt`.
 - Example Files:
 - Input File (`grades.txt`):
85
90
78
92
88
 - Output File (`results.txt`):
 - Average Grade: 86.6
 - Highest Grade: 92
 - Lowest Grade: 78

Follow-up Activity

1. **Submission:** Just make sure that you have the output provided above. This is not tied to any of your tasks.
2. Use any method available to you. As long as you understand the process.

Questions and Answers



Thank you for attending



Department
for Education

CoGrammar

