Welcome to this CoGrammar Tutorial: Text File IO and Exception-Handling

The session will start shortly...

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

CoGrammar

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

- ***Guided Learning Hours (GLH):*** **60**
- ***Task Completion:*** 13 tasks

Due Date: 28 April 2024

CoGrammar

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

SKILLS FOR LIFE

SKILLS BOOTCAMPS

Department for Education

CoGrammar

Text File IO and Exception-Handling

March 2024

Agenda

- ❖ Text File IO
- ❖ Resource Management
- ❖ Try
- ❖ Except
- ❖ Finally
- ❖ Custom Exceptions

Text File IO

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 Modes

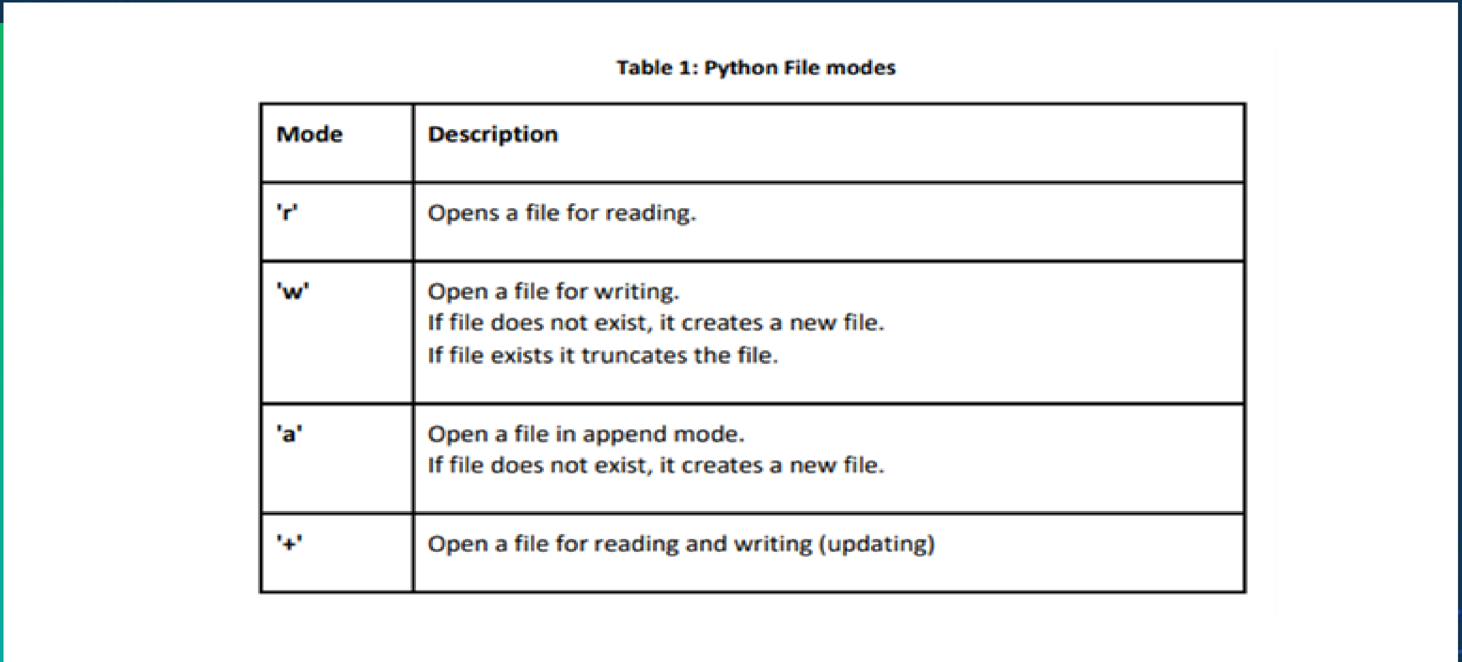
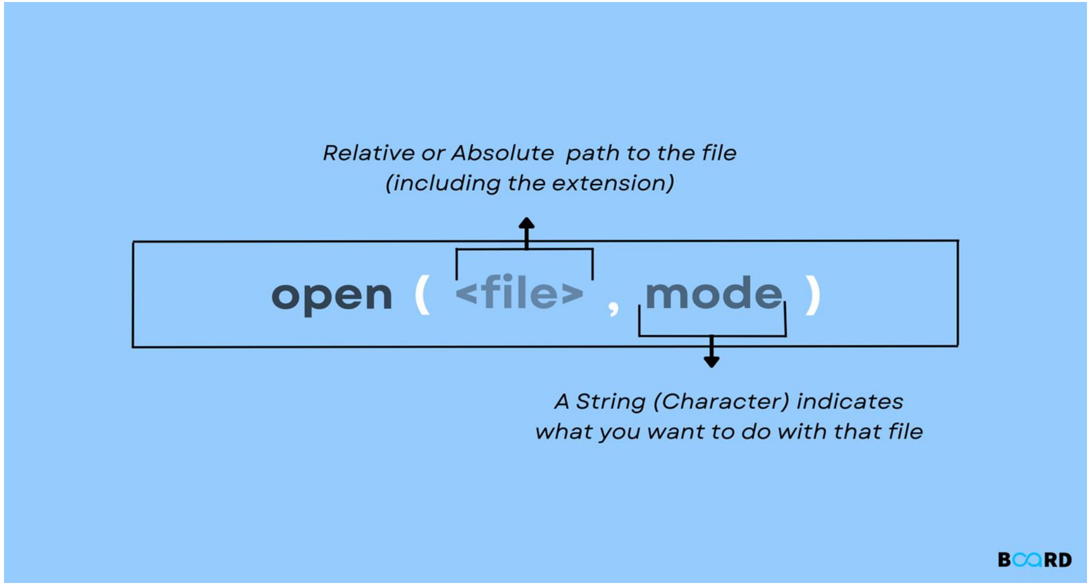


Table 1: Python File modes

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)

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

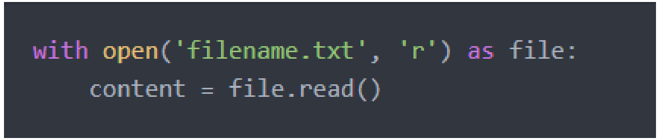
Resource Management

CoGrammar

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.

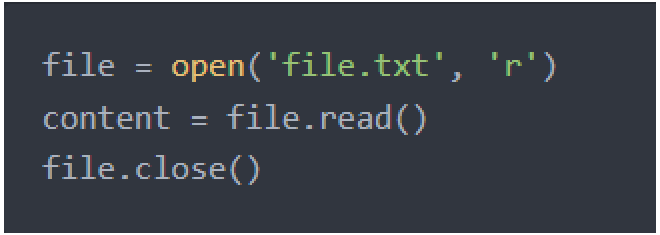


```
with open('filename.txt', 'r') as file:
    content = file.read()
```

Resource Management

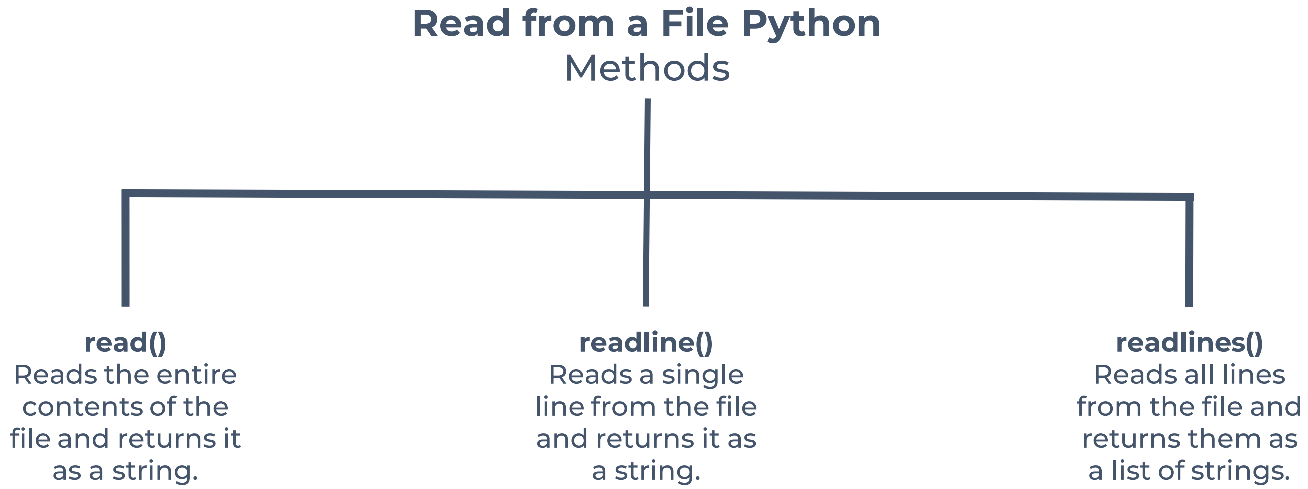
Explicit Method

❖ 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 Handling (Reading)



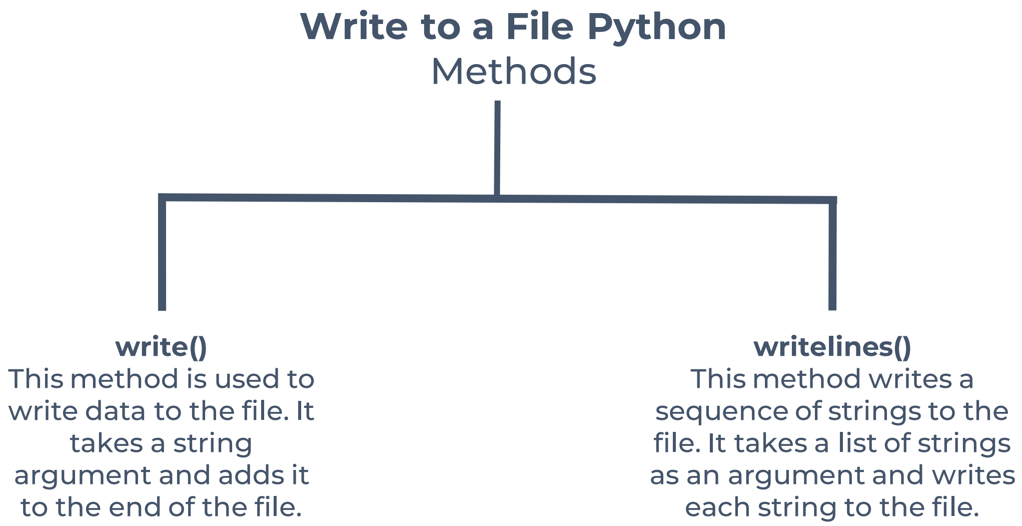
Read from a File Python
Methods

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.

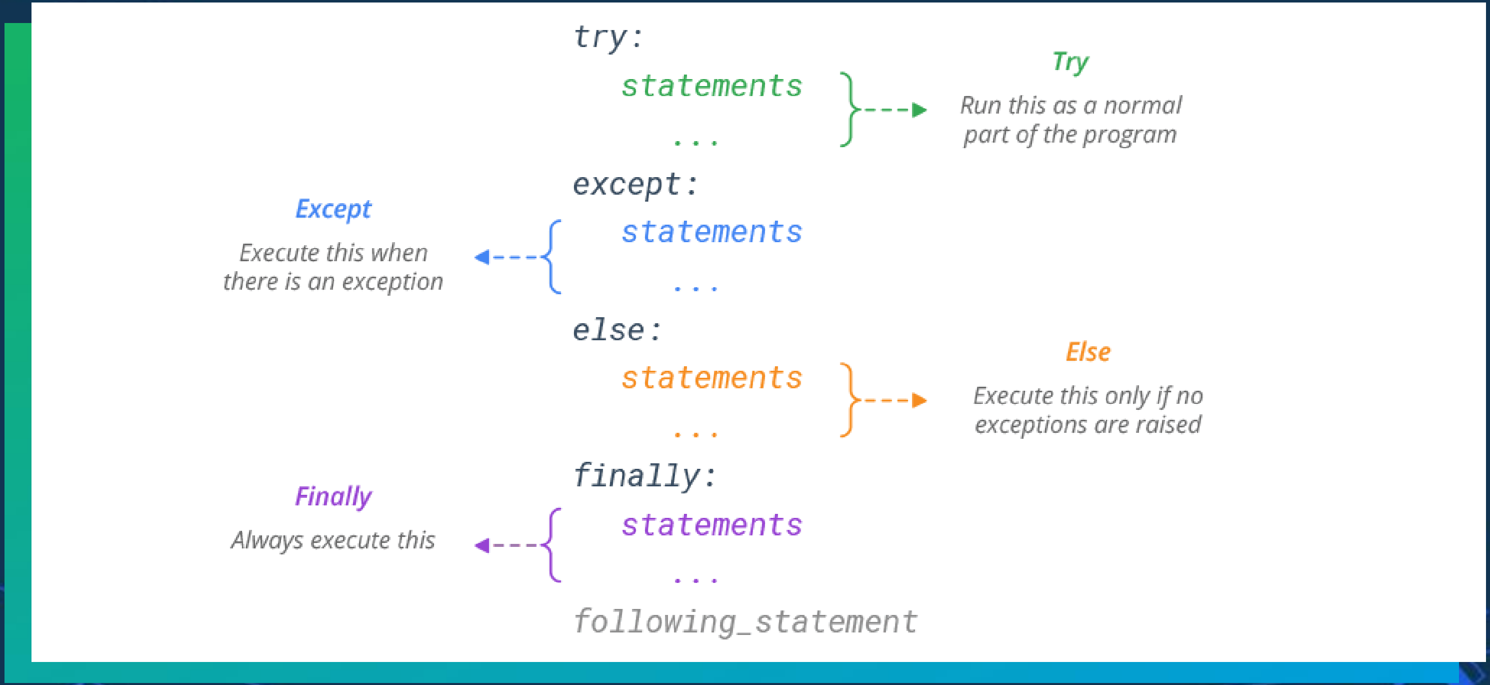
Let’s take a short break

CoGrammar

Try / Except Finally

CoGrammar

Try / Except / Finally



```
try:
    statements
    ...
except:
    statements
    ...
else:
    statements
    ...
finally:
    statements
    ...
following_statement
```

Try

Run this as a normal part of the program

Except

Execute this when there is an exception

Else

Execute this only if no exceptions are raised

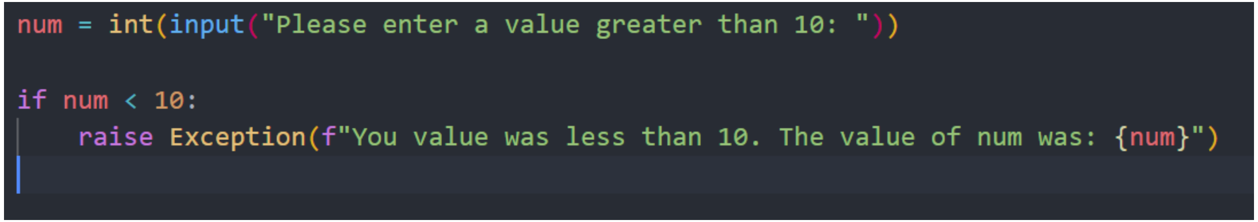
Finally

Always execute this

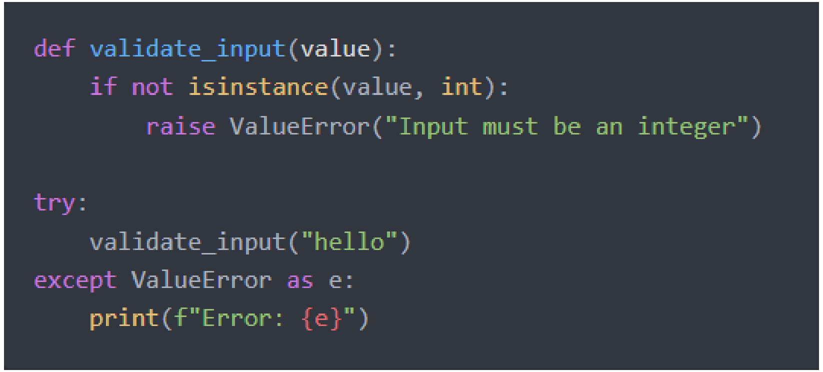
Custom Exceptions

CoGrammar

Custom Exceptions

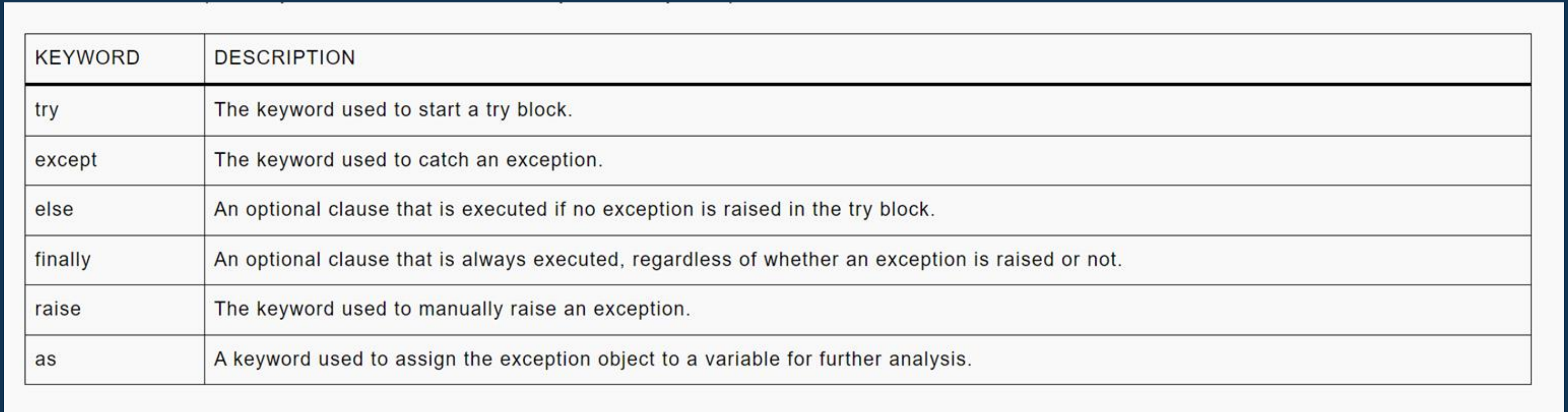


```
num = int(input("Please enter a value greater than 10: "))

if num < 10:
    raise Exception(f"You value was less than 10. The value of num was: {num}")
```



```
def validate_input(value):
    if not isinstance(value, int):
        raise ValueError("Input must be an integer")

try:
    validate_input("hello")
except ValueError as e:
    print(f"Error: {e}")
```

Terminology



KEYWORD	DESCRIPTION
try	The keyword used to start a try block.
except	The keyword used to catch an exception.
else	An optional clause that is executed if no exception is raised in the try block.
finally	An optional clause that is always executed, regardless of whether an exception is raised or not.
raise	The keyword used to manually raise an exception.
as	A keyword used to assign the exception object to a variable for further analysis.

A Note on try-except

1. It may be tempting to wrap all code in a try-except block. However, you want to handle different errors differently.
2. Don’t try to use try-except blocks to avoid writing code that properly validates inputs.
3. The correct usage for try except should only be for “exceptional” cases. Eg: The potential of Division by 0.
4. Raise Exceptions When Necessary; If your code encounters an exceptional condition that it cannot handle, consider raising an exception using the raise statement.

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar