Welcome to this CoGrammar Lecture: 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

60 Guided Learning Hours

Data Science - **13 tasks**
Software Engineering - **13 tasks**
Web Development - **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

Exception-Handling

March 2024

Agenda

- ❖ Errors
- ❖ Try
- ❖ Except
- ❖ Finally
- ❖ Resource Management
- ❖ Custom Exceptions

CoGrammar

We all make mistakes :)

❖ No programmer is perfect, and we’re going to make a lot of mistakes in our journey – and that is perfectly okay!

❖ What separates the good programmers from the rest is the ability to find and debug errors that they encounter.

*“A wise being once said **We fail upwards here…**” - Serge… or Einstein*

CoGrammar

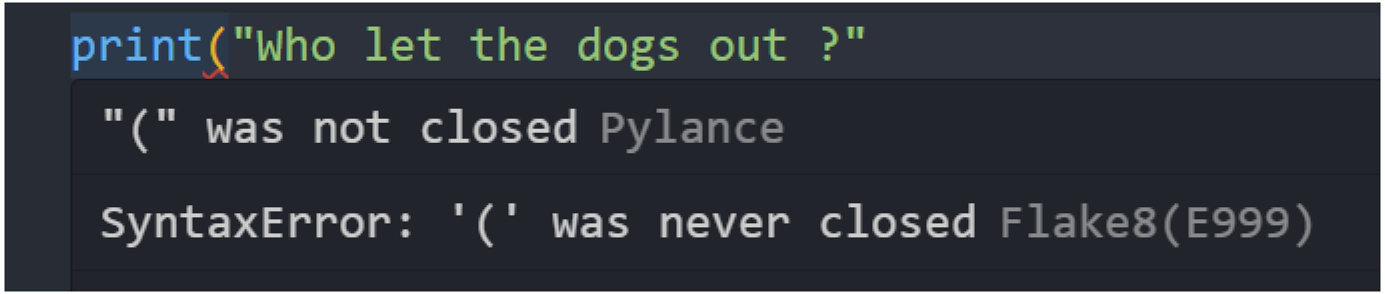
Syntax Errors

- ❖ Some of the easiest errors to fix… usually.
- ❖ Mainly caused by typos in code or Python specific keywords that were misspelled or rules that were not followed.
- ❖ When incorrect syntax is detected, Python will stop running and display an error message.

CoGrammar

Syntax Errors

Syntax Error Example



```
print("Who let the dogs out ?"
"(" was not closed Pylance
SyntaxError: '(' was never closed Flake8(E999)
```

CoGrammar

Logical Errors

- **Logical errors** occur when your program is running, but the output you are receiving is not what you are expecting.
- The code could be typed incorrectly, or perhaps an important line has been omitted, or the instructions given to the program have been coded in the wrong order.

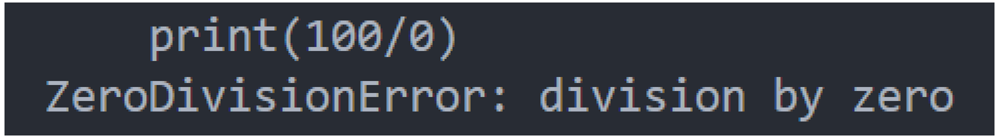
$$1 + 1 = 3$$

CoGrammar

Runtime Errors

- **Runtime errors** occur during the execution of a program, and they typically result from issues that manifest when the program is running rather than during the compilation or interpretation phase.

- **Runtime errors** are often detected when the program is running and can lead to the termination of the program if not handled properly.



```
    print(100/0)
ZeroDivisionError: division by zero
```

Defensive Programming

- Programmers anticipate errors:
    - User errors
    - Environment errors
    - Logical errors
- Code is written to ensure that these errors don't crash the code base.
- Two ways - if statements and try-except blocks.

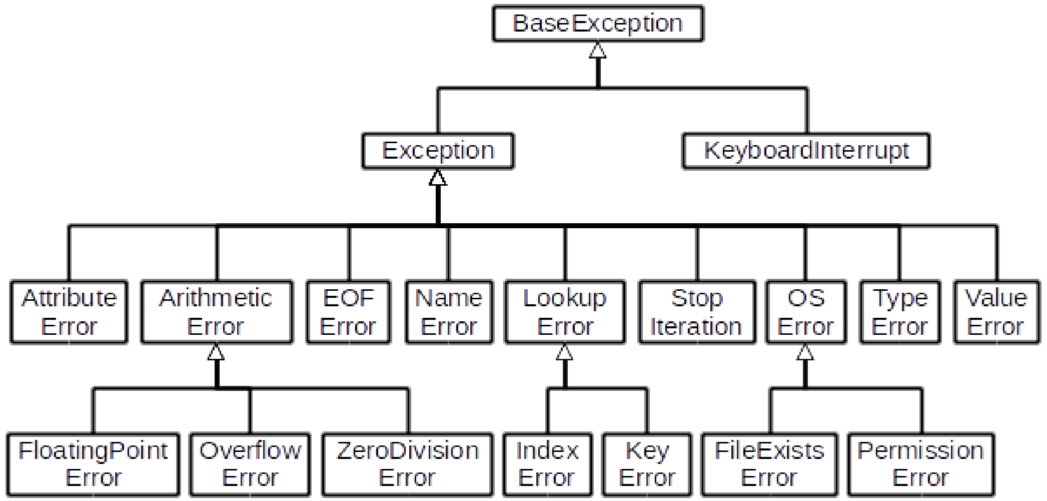
CoGrammar

What are Exceptions ?

❖ An **exception** is an event that occurs during the execution of a program, disrupting the normal flow of its initial instructions.

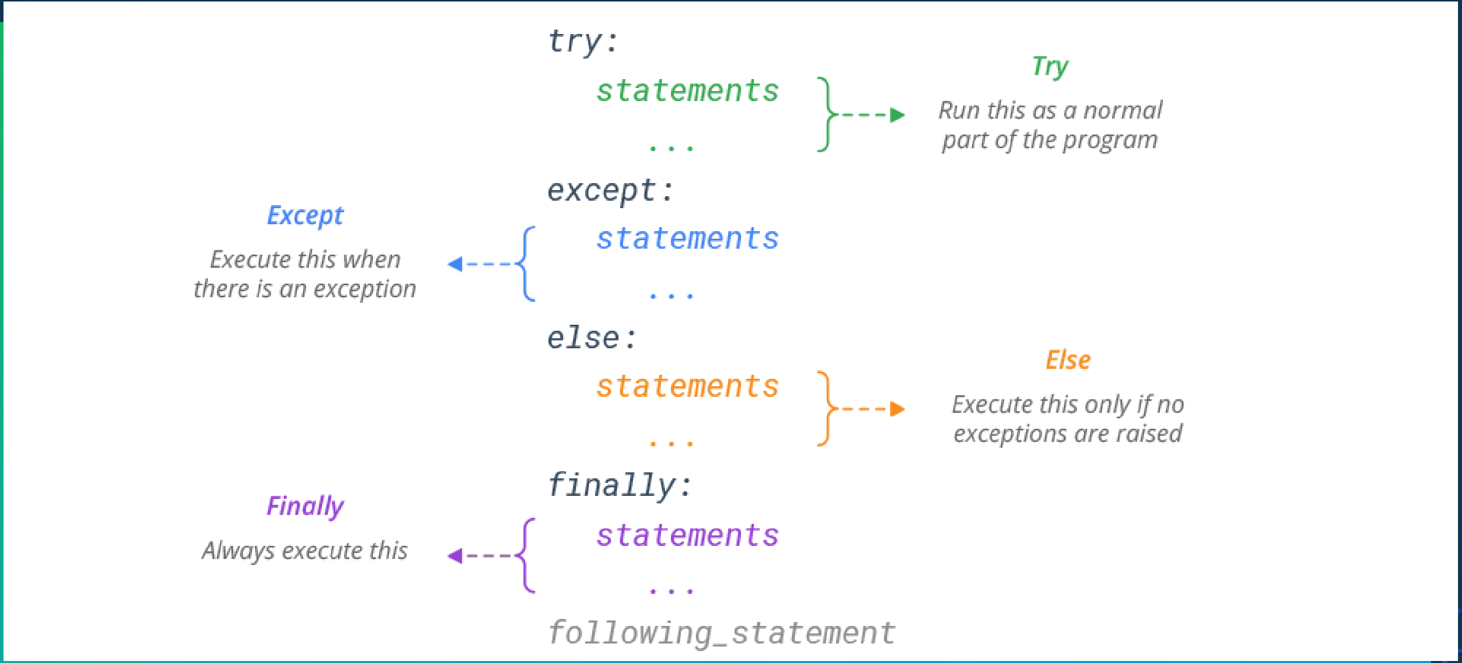
CoGrammar

Basic types of exceptions



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

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.

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

CoGrammar

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()
```

CoGrammar

Custom Exceptions

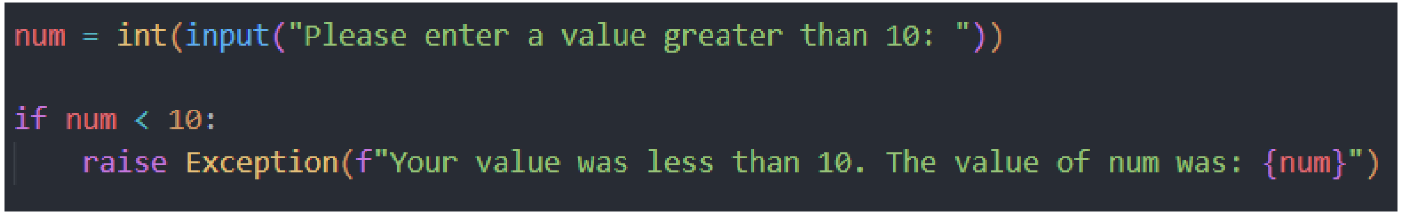
❖ There will be occasions when you want your program to raise a custom exception whenever a certain condition is met.

❖ In Python we can do this by using the **“raise”** keyword and adding a custom message to the exception

❖ The raise statement allows you to handle exceptional conditions in your program explicitly, providing better control over error handling and making your code more robust and predictable.

Custom Exceptions

❖ We’re prompting the user to enter a value> 10. If the user enters a number that does not meet that condition, an exception is raised with a custom error message.



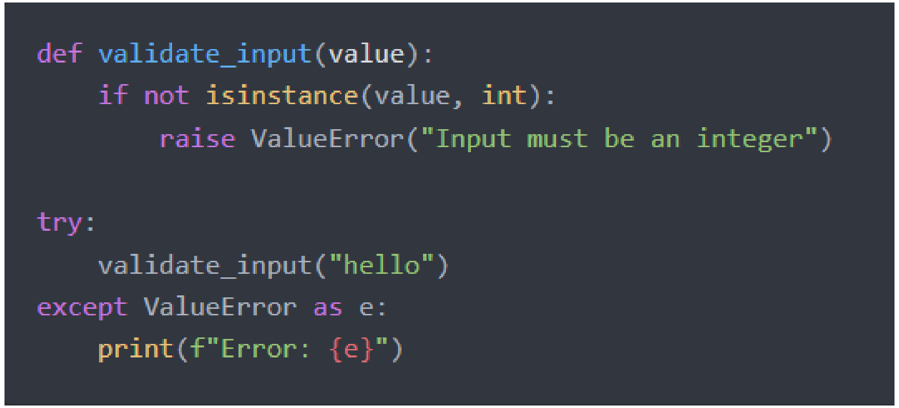
```python
num = int(input("Please enter a value greater than 10: "))

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

CoGrammar

Custom Exceptions cont.

- You can include additional information when raising exceptions by passing arguments to the exception constructor. This can be useful for providing context about the error:

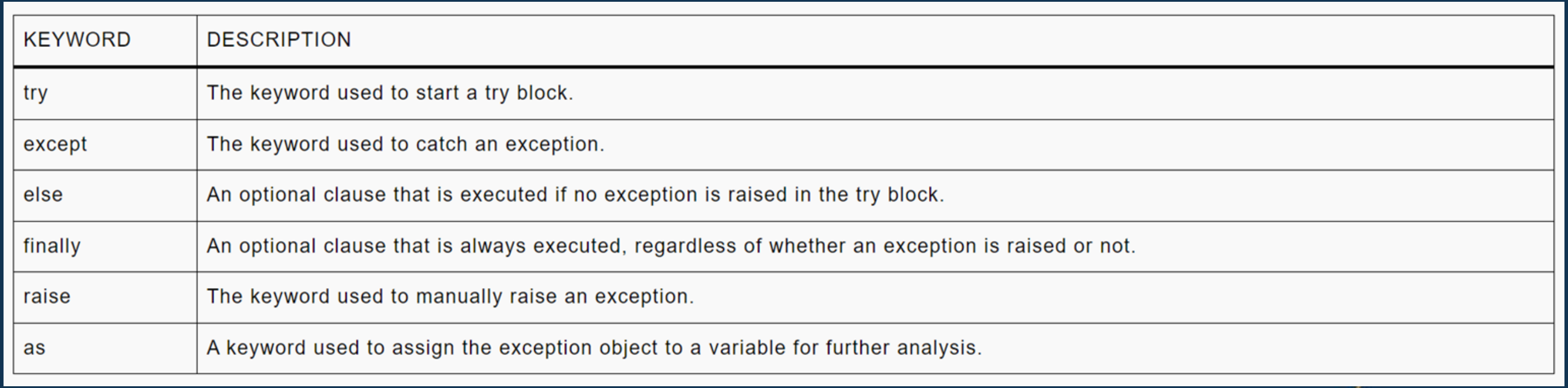


```python
def validate_input(value):
    if not isinstance(value, int):
        raise ValueError("Input must be an integer")

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

CoGrammar

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