CoGrammar

Error Handling

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

Software Engineering 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**

Software Engineering 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**

Prestigious Co-Certification Opportunities

New Partnerships!

- **University of Manchester & Imperial College London** join our circle along with The University of Nottingham Online.

Exclusive Opportunity:

- Co-certification spots awarded on a first-come basis.
- Meet the criteria early to gain eligibility for the co-certification.

New Deadlines:

- **11 March 2024**: 112 GLH & BYB tasks completion.
- **18 March 2024**: Record interview invitation or self-employment.
- **15 July 2024**: Submit verified job offer or new contract.

CoGrammar

Lecture Objectives

1. Identify common errors and exceptions in Python.

2. Implement raising and handling of exceptions to make your programs more robust.

Poll:

Assessment

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.

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.

Syntax Errors

```
print("Hello world!"
```

```
    print("Hello world!"
          ^
SyntaxError: '(' was never closed
```

Logical Errors

$$1 + 1 = 3$$

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.

Runtime Errors

```
print(100/0)
```

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

Defensive Programming

- Programmers anticipate errors.
  - User errors
  - Environment errors
  - Logical errors

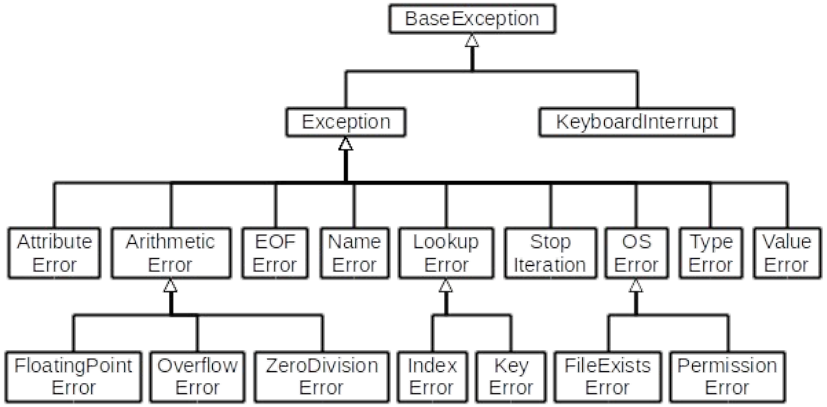
- Code is written to ensure that these errors don't crash the code base.

- Two ways - if statements and try-except blocks.

What are exceptions ?

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

Basic types of exceptions



try-except block structure

```
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

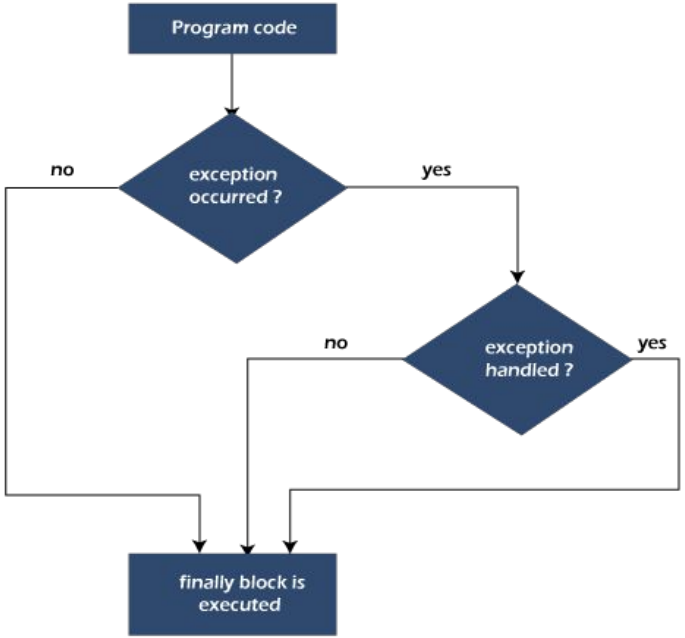
Execute this only if no exceptions are raised

Finally

Always execute this

https://www.learnbyexample.org/python-exceptions-try-except/

finally block



Using Try..Except

```python
num = input("Please enter a number: ")
try:
    num = int(num)
except ValueError:
    print("Please enter numbers only!")
```

```
Please enter a number: d
Please enter numbers only!
```

Using Try..Except

```python
num = input("Please enter a number: ")
try:
    num = int(num)
except ValueError:
    print("Please enter numbers only!")
else:
    print("Thank you for entering a number!")
```

```
Please enter a number: 5
Thank you for entering a number!
```

Using Try..Except

```
num = input("Please enter a number: ")
try:
    num = int(num)
except ValueError:
    message = "Please enter numbers only!"
    print(message)
else:
    message = "Thank you for entering a number!"
    print(message)
finally:
    # Let's pretend we have a funtion that logs data to a file
    log_data(message)
```

Using Conditionals Statements

```python
num = input("Please enter a number: ")

if num.isdigit():
    message = "Thank you for entering a number!"
    print(message)
else:
    message = "Please enter numbers only!"
    print(message)

# Let's pretend we have a funtion that logs data to a file
log_data(message)
```

Raising 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: In the next example 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.

Raising Exceptions

```
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}")
```

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

- ★ It may be tempting to wrap all code in a try-except block. However, you want to handle different errors differently.
- ★ Don't try to use try-except blocks to avoid writing code that properly validates inputs.
- ★ The correct usage for try except should only be for "exceptional" cases. Eg: The potential of Division by 0.

Wrapping Up

Error Types

There are 3 types of errors. Syntax , runtime and Logical.

Exception Handling

We can handle the errors that occur in our programs using try-except blocks or we can make use of conditional statements.

CoGrammar

Questions

CoGrammar

Thank you for joining