Python

Exceptions

Session Expectations:

- Cameras on 📷
- Mics muted 🔇
- Raise Hand ✋
- React 👍
- Respond "?"

Learning Objectives

- **Intent**: To explore the concepts of **Exceptions** in Python.
- **Implementation**: To add **exception handling** into our code.
- **Impact**: To contextualise **why** we would utilise **exception handling**.

What are Exceptions?

An exception in Python is an error that occurs during the execution of a program, which can disrupt the normal flow of the program's instructions.

Exceptions help manage errors in a controlled way, allowing the program to handle them gracefully without crashing.

Think of an exception like a special signal that indicates an issue, which you can catch and respond to in your code to maintain smooth operation.

When exceptions occur in our code they need to be handled, this is where Exception Handling comes in.

Exception handling is a mechanism to manage the errors, enabling the program to continue running or gracefully terminate.

Exception handling involves using "try", "except", "else", and "finally" blocks to catch and respond to exceptions, providing a way to execute alternative code when errors arise.

Think of exception handling like a safety net that catches errors, allowing us to address them and keep your program running smoothly.

Exception Handling - Division Example:

example.py

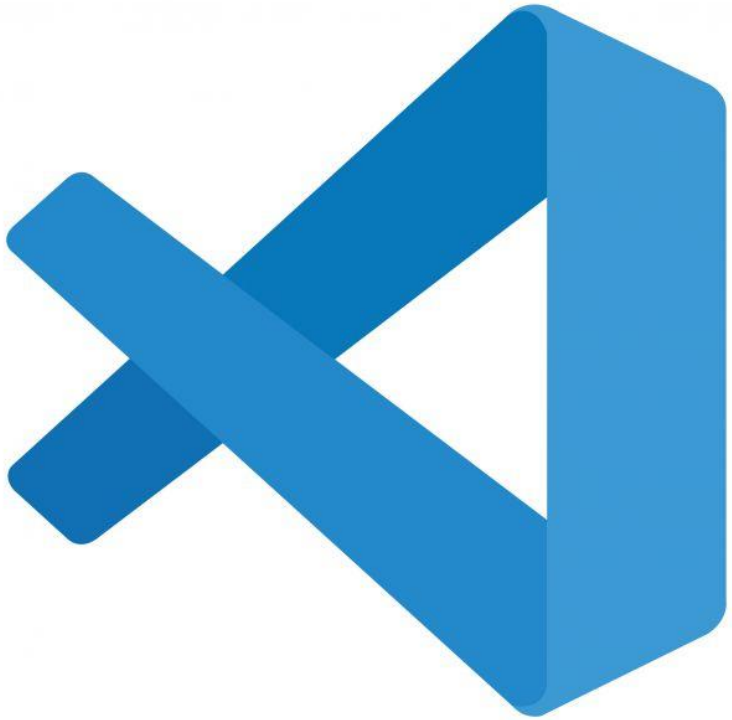
```python
try:
    # Code that may raise an exception
    num1 = int(input("Enter a number: "))  # Attempt to convert user input to an integer
    num2 = int(input("Enter another number: "))  # Attempt to convert user input to an integer

    result = num1 / num2  # Perform division operation

    # Print the result of the division
    print(f"The result of {num1} divided by {num2} is: {result}")

except ValueError:
    # Handle ValueError (if input cannot be converted to integer)
    print("Invalid input. Please enter valid integers.")

except ZeroDivisionError:
    # Handle ZeroDivisionError (if second number is zero)
    print("Error: Division by zero is not allowed.")

except Exception as e:
    # Handle any other exceptions not caught above
    print(f"An error occurred: {e}")

finally:
    # Execute this block of code regardless of whether an exception occurred or not
    print("Execution completed.")
```

“ValueError” and “ZeroDivisionError” are examples of Exceptions that are built into Python.

As developers we can also create custom exceptions to handle various types of errors.

A list of built-in exceptions can be found on the resources slide.

Let's move over to VS Code to work through some examples



Reference Links:

Built-In Exceptions:
https://www.programiz.com/python-programming/exceptions

Custom Exceptions:
https://www.programiz.com/python-programming/user-defined-exception