Module 4: Error Handling

# Dealing with Syntax Errors:

In Python, syntax errors occur when the code violates the rules of the Python programming language. These errors are detected during the parsing phase before the code is executed. Common causes include missing parentheses, incorrect indentation, or invalid syntax in general.

* Syntax errors are detected by the Python interpreter during the parsing phase.
* Always carefully check the code for correct syntax, indentation, and matching parentheses.

Example:

# Syntax Error: Missing closing parenthesis

print("Hello, World!"

# Exceptions:

Exceptions, on the other hand, are runtime errors that occur during the execution of a program. These can include division by zero, accessing an index that doesn't exist in a list, or trying to open a file that doesn't exist.

* Exceptions are runtime errors that occur during the execution of the program.
* Common exceptions include ZeroDivisionError, IndexError, and FileNotFoundError.

Example:

# Exception: Division by zero

result = 10 / 0

# Handling Exceptions with try/except:

To gracefully handle exceptions and prevent the program from crashing, Python provides the try and except blocks. The code inside the try block is executed, and if an exception occurs, the corresponding except block is executed, allowing for controlled error handling.

* Use try and except blocks to handle exceptions gracefully.
* This prevents the program from terminating abruptly and allows for controlled error handling.
* Specify the type of exception to catch in the except block.
* Use multiple except blocks to handle different types of exceptions.

Example:

try:

result = 10 / 0

except ZeroDivisionError:

print("Cannot divide by zero!")

## Example 1: Syntax Error

# Syntax Error: Missing closing parenthesis

print("Hello, World!"

This program intentionally contains a syntax error by omitting the closing parenthesis. Running it will result in a SyntaxError.

## Example 2: Handling Division by Zero

# Handling Division by Zero

try:

result = 10 / 0

except ZeroDivisionError:

print("Cannot divide by zero!")

This program attempts to perform division by zero within a try block. The except block catches the ZeroDivisionError and prints a custom error message.

## Example 3: Handling IndexError

# Handling IndexError

try:

my\_list = [1, 2, 3]

value = my\_list[5]

except IndexError:

print("Index out of range!")

This program attempts to access an index that doesn't exist in the list. The except block catches the IndexError and prints a custom error message.

## Example 4: Using the else Block

# Using the else block in exception handling

try:

result = 10 / 2

except ZeroDivisionError:

print("Cannot divide by zero!")

else:

print("Division successful. Result:", result)

This program divides two numbers inside a try block. If no exception occurs, the code inside the else block is executed.

## Example 5: Finally Block

# Using the finally block in exception handling

try:

file = open("nonexistent\_file.txt", "r")

content = file.read()

except FileNotFoundError:

print("File not found!")

finally:

print("This block always executes, whether an exception occurred or not.")

file.close() # Ensure the file is closed, even if an exception occurred.

This program attempts to open a file that doesn't exist, resulting in a FileNotFoundError. The finally block ensures that certain code is executed regardless of whether an exception occurred.