**Lab Exercise: Exception Handling in Python**

**Objective:**

To understand and implement exception handling in Python by handling specific errors and using the sys library to display exception details.

**Part 1: Handling FileNotFoundError**

**Task:**

1. Write a Python function that tries to open and read a file.
2. If the file does not exist, catch the FileNotFoundError and display an appropriate message.

**Example Code:**

**def read\_file(filename):**

**try:**

**with open(filename, 'r') as file:**

**print(file.read())**

**except FileNotFoundError as e:**

**print(f"Error: {e}")**

**print("This file does not exist. Please check the filename and try again.")**

# Test the function

**read\_file("missing\_file.txt")**

**Expected Output:**

**Error: [Errno 2] No such file or directory: 'missing\_file.txt'**

**This file does not exist. Please check the filename and try again.**

**Part 2: Handling ZeroDivisionError**

**Task:**

1. Write a function that takes two numbers as input and divides them.
2. Handle the ZeroDivisionError exception when dividing by zero.

**Example Code:**

**def divide\_numbers(a, b):**

**try:**

**result = a / b**

**print(f"Result: {result}")**

**except ZeroDivisionError as e:**

**print(f"Error: {e}")**

**print("Cannot divide by zero!")**

# Test the function

**divide\_numbers(10, 0**)

**Expected Output:**

**Error: division by zero**

**Cannot divide by zero!**

**Part 3: Using sys.exc\_info() for Debugging**

**Task:**

1. Modify the functions to use sys.exc\_info() to print the exception type.

**import sys**

**def divide\_numbers(a, b):**

**try:**

**result = a / b**

**print(f"Result: {result}")**

**except ZeroDivisionError as e:**

**print(f"Error: {e}")**

**print("Exception Type:", sys.exc\_info()[0].\_\_name\_\_)**

**print("Cannot divide by zero!")**

# Test the function

**divide\_numbers(10, 0)**

**Expected Output:**

**Error: division by zero**

**Exception Type: ZeroDivisionError**

**Cannot divide by zero!**

**Part 4: Handling Multiple Exceptions**

**Task:**

1. Write a function that performs multiple operations and catches different types of exceptions.
2. Handle ValueError when converting a string to an integer.
3. Handle ZeroDivisionError for division errors.

**def safe\_operations():**

**try:**

**num1 = int(input("Enter a number: ")) # May raise ValueError**

**num2 = int(input("Enter another number: ")) # May raise ValueError**

**result = num1 / num2 # May raise ZeroDivisionError**

**print(f"Result: {result}")**

**except ValueError as e:**

**print(f"Error: {e}")**

**print("Invalid input! Please enter numeric values.")**

**except ZeroDivisionError as e:**

**print(f"Error: {e}")**

**print("Cannot divide by zero!")**

# Test the function

**safe\_operations()**

**Example Inputs & Expected Outputs:**

**Case 1 (Invalid Input):**

**Enter a number: ten**

**Error: invalid literal for int() with base 10: 'ten'**

**Invalid input! Please enter numeric values.**

**Case 2 (Zero Division):**

**Enter a number: 5**

**Enter another number: 0**

**Error: division by zero**

**Cannot divide by zero!**

**You can specify the encoding when opening a file in Python using the encoding parameter in the open() function. Here’s an example:**

**# Open a file with UTF-8 encoding**

**with open("example.txt", "r", encoding="utf-8") as file:**

**content = file.read()**

**print(content)**

**Other Common Encodings:**

* **UTF-8 (default on most systems) → "utf-8"**
* **UTF-16 → "utf-16"**
* **UTF-32 → "utf-32"**
* **ISO-8859-1 (Latin-1) → "iso-8859-1"**
* **Windows-1252 → "cp1252"**

**For writing to a file with a specific encoding:**

**with open("output.txt", "w", encoding="utf-8") as file:**

**file.write("Hello, world!")**

**You can use a try-except block with sys to catch and display any unexpected errors at the end of your script. Here’s how you can modify your code to include a last-chance exception handler:**

**import sys**

**try:**

**# Open a file with UTF-8 encoding**

**with open("example.txt", "r", encoding="utf-8") as file:**

**content = file.read()**

**print(content)**

**# Writing to a file with UTF-8 encoding**

**with open("output.txt", "w", encoding="utf-8") as file:**

**file.write("Hello, world!")**

**except Exception as e:**

**print(f"An unexpected error occurred: {e}", file=sys.stderr)**

**How This Works:**

* **The try block contains file operations.**
* **If any unanticipated error occurs, the except Exception as e block catches it.**
* **The error message is printed to sys.stderr (standard error output).**

**You can add a finally block to ensure that "Execution completed." is printed no matter what happens. Here’s how the updated code looks:**

**import sys**

**try:**

**# Open a file with UTF-8 encoding**

**with open("example.txt", "r", encoding="utf-8") as file:**

**content = file.read()**

**print(content)**

**# Writing to a file with UTF-8 encoding**

**with open("output.txt", "w", encoding="utf-8") as file:**

**file.write("Hello, world!")**

**except Exception as e:**

**print(f"An unexpected error occurred: {e}", file=sys.stderr)**

**finally:**

**print("Execution completed.")**

**Explanation:**

* **try block → Runs the main code.**
* **except Exception as e → Catches any unexpected errors and prints them.**
* **finally block → Always runs, whether or not an exception occurs.**

**This guarantees that "Execution completed." is printed at the end, ensuring a clean program exit.**