# Exception Handling – Question Paper

## Section A: Basic Try-Except (2 marks each)

1. Write a program to divide two numbers entered by the user. Handle ZeroDivisionError using try-except.

try:

a = int(input("Enter numerator: "))

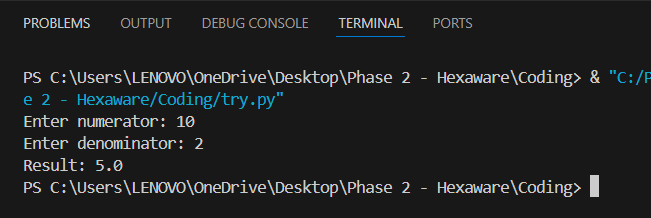
b = int(input("Enter denominator: "))

result = a / b

print("Result:", result)

except ZeroDivisionError:

print("Error: Cannot divide by zero.")



1. Write a program to convert a string to an integer. Handle ValueError if the input is not a valid number.

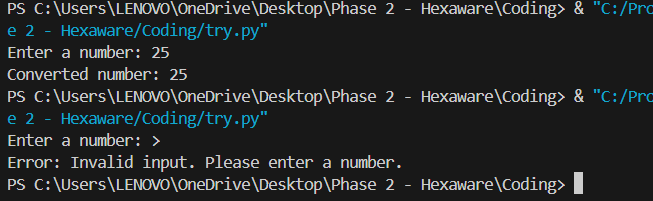
try:

num = int(input("Enter a number: "))

print("Converted number:", num)

except ValueError:

print("Error: Invalid input. Please enter a number.")



1. Accept two numbers from the user and perform addition. Use try-except to handle invalid input types.

try:

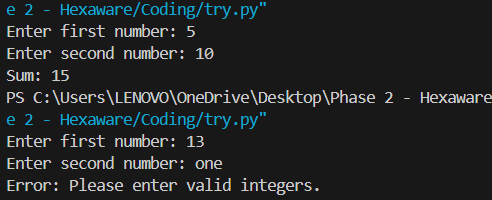
a = int(input("Enter first number: "))

b = int(input("Enter second number: "))

print("Sum:", a + b)

except ValueError:

print("Error: Please enter valid integers.")



1. Write a program to read an element from a list using an index entered by the user. Handle IndexError.

my\_list = [10, 20, 30, 40]

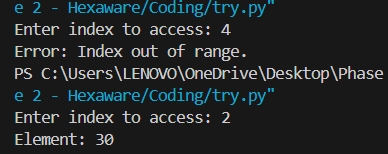
try:

index = int(input("Enter index to access: "))

print("Element:", my\_list[index])

except IndexError:

print("Error: Index out of range.")



## Section B: Try-Except-Else (4 marks each)

1. Create a program that accepts a number from the user and prints its square. Use try-except-else to handle ValueError and ensure successful computation is shown only if there's no error.

try:

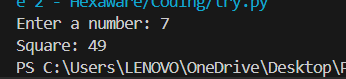
num = int(input("Enter a number: "))

except ValueError:

print("Error: Invalid input.")

else:

print("Square:", num \*\* 2)



1. Write a program to open a file and read contents. Use try-except-else to handle FileNotFoundError.

try:

file = open("data.txt", "r")

except FileNotFoundError:

print("Error: File not found.")

else:

print("File contents:\n", file.read())

file.close()



1. Write a Python program to convert a number to its binary format. Use try-except-else to handle any invalid input.

try:

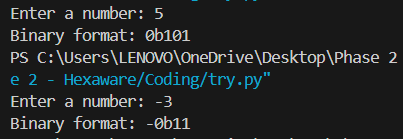
num = int(input("Enter a number: "))

except ValueError:

print("Error: Not a valid integer.")

else:

print("Binary format:", bin(num))



## Section C: Try-Finally (5 marks each)

1. Write a program that opens a file and ensures it gets closed, whether or not an exception occurs. Use try-finally.

try:

file = open("data.txt", "r")

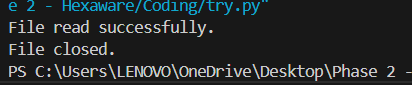
content = file.read()

print("File read successfully.")

finally:

file.close()

print("File closed.")



1. Simulate a login process where the user input is handled in a try block and a log message is printed in finally regardless of success or failure.

try:

username = input("Enter username: ")

password = input("Enter password: ")

if username == "admin" and password == "1234":

print("Login successful.")

else:

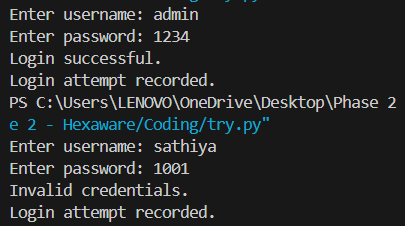
raise ValueError("Invalid credentials.")

except ValueError as e:

print(e)

finally:

print("Login attempt recorded.")



1. Write a program that divides two numbers, catching errors with try-except, and printing a clean-up message using finally.

try:

a = int(input("Enter numerator: "))

b = int(input("Enter denominator: "))

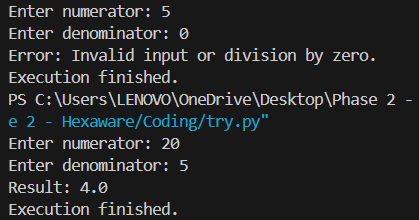
print("Result:", a / b)

except (ZeroDivisionError, ValueError):

print("Error: Invalid input or division by zero.")

finally:

print("Execution finished.")



## Section D: Combined Exception Handling (6 marks each)

1. Create a program that handles multiple exceptions: ZeroDivisionError, ValueError, and always prints "Execution complete" using finally.

try:

a = int(input("Enter first number: "))

b = int(input("Enter second number: "))

print("Result:", a / b)

except ZeroDivisionError:

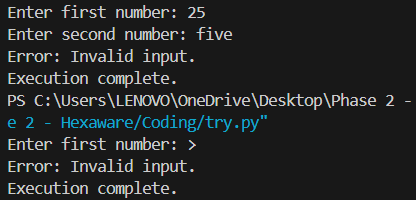
print("Error: Cannot divide by zero.")

except ValueError:

print("Error: Invalid input.")

finally:

print("Execution complete.")



1. Write a program to simulate bank withdrawal. Use try-except-else-finally to handle incorrect amount input, and always print a message whether the transaction succeeded or failed.

balance = 5000

try:

amount = int(input("Enter withdrawal amount: "))

if amount <= 0:

raise ValueError("Amount must be positive.")

if amount > balance:

raise ValueError("Insufficient balance.")

except ValueError as e:

print("Transaction failed:", e)

else:

balance -= amount

print(f"Withdrawal successful. Remaining balance: ₹{balance}")

finally:

print("Transaction process completed.")

