# 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("Q1 - Enter numerator: "))

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

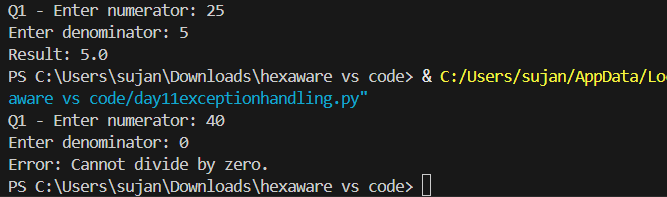
result = a / b

print("Result:", result)

except ZeroDivisionError:

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

OUTPUT



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

try:

s = input("\nQ2 - Enter a number string: ")

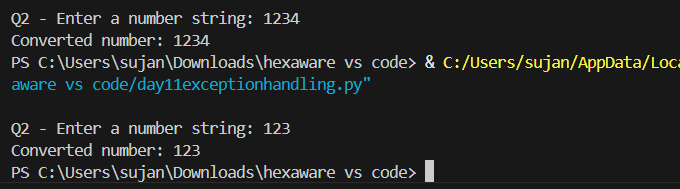
num = int(s)

print("Converted number:", num)

except ValueError:

print("Error: Invalid number format.")

OUTPUT:



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

try:

a = float(input("\nQ3 - Enter first number: "))

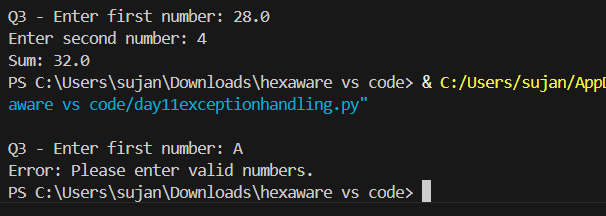
b = float(input("Enter second number: "))

print("Sum:", a + b)

except ValueError:

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

OUTPUT



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, 50]

try:

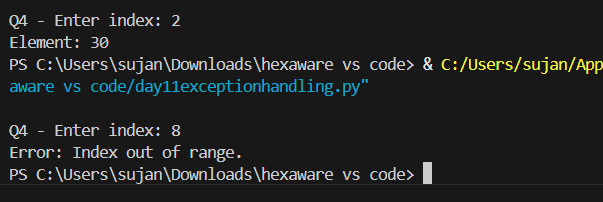
index = int(input("\nQ4 - Enter index: "))

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

except IndexError:

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

OUTPUT



## 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("\nQ5 - Enter a number: "))

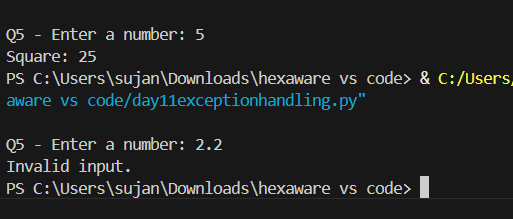
except ValueError:

print("Invalid input.")

else:

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

OUTPUT



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

try:

filename = input("\nQ6 - Enter filename to open: ")

file = open(filename, 'r')

except FileNotFoundError:

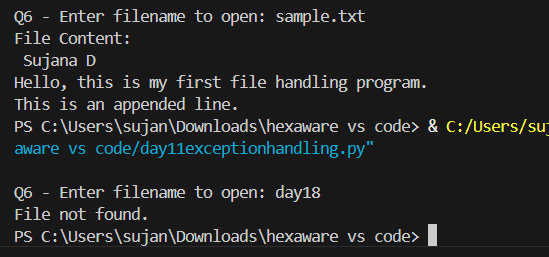
print("File not found.")

else:

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

file.close()

OUTPUT



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("\nQ7 - Enter an integer: "))

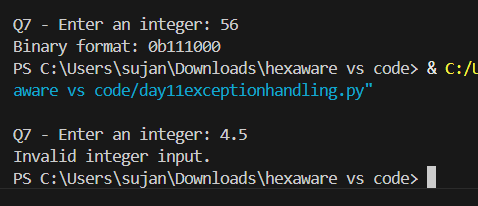
except ValueError:

print("Invalid integer input.")

else:

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

OUTPUT



## 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("sample.txt", "r")

content = file.read()

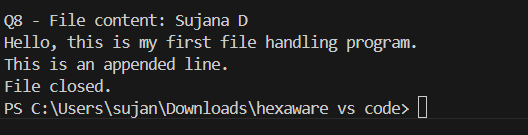
print("\nQ8 - File content:", content)

finally:

file.close()

print("File closed.")

OUTPUT



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("\nQ9 - Enter username: ")

password = input("Enter password: ")

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

print("Login successful.")

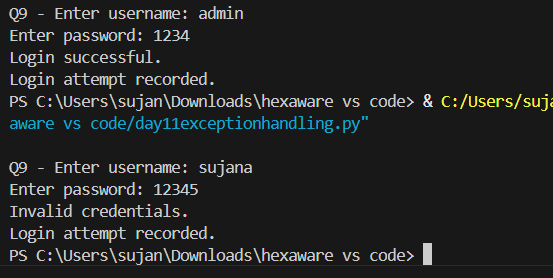
else:

print("Invalid credentials.")

finally:

print("Login attempt recorded.")

OUPUT



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("\nQ10 - Enter numerator: "))

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

result = a / b

print("Result:", result)

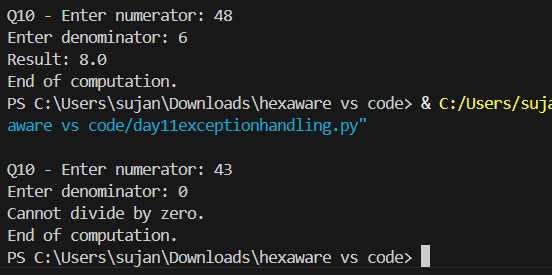
except ZeroDivisionError:

print("Cannot divide by zero.")

finally:

print("End of computation.")

OUTPUT



## 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("\nQ11 - Enter numerator: "))

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

result = a / b

print("Result:", result)

except ZeroDivisionError:

print("Division by zero error.")

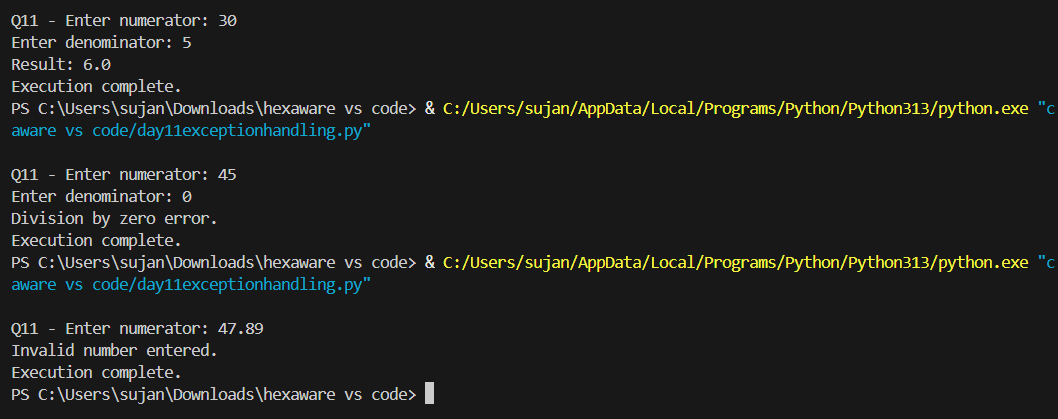
except ValueError:

print("Invalid number entered.")

finally:

print("Execution complete.")

OUTPUT



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 = 1000

try:

amount = float(input("\nQ12 - Enter withdrawal amount: "))

if amount > balance:

raise ValueError("Insufficient balance.")

except ValueError as e:

print("Transaction failed:", e)

else:

balance -= amount

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

finally:

print("Transaction process ended.")

OUTPUT

