# Exception Handling – Question Paper DATE:23/06/25

## 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: "))

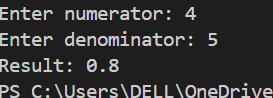
    b = int(input("Enter denominator: "))

    result = a / b

    print("Result:", result)

except ZeroDivisionError:

    print("Error: You 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:

    user\_input = input("Enter a number: ")

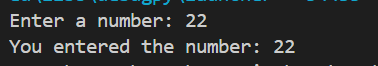
    number = int(user\_input)

    print("You entered the number:", number)

except ValueError:

    print("Error: That is not a valid number. Please enter digits only.")





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

try:

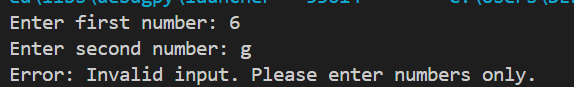
    a = int(input("Enter first number: "))

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

    print("Sum =", a + b)

except ValueError:

    print("Error: Invalid input. Please enter numbers only.")



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 (0 to 3): "))

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

except IndexError:

    print("Error: Index is 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: Please enter a valid integer.")

else:

    print("Square of the number is:", num \* num)



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

try:

    file = open("Sample.txt", "r")  # Try opening the file

except FileNotFoundError:

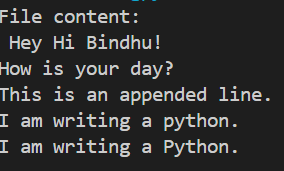
    print("Error: File not found.")

else:

    content = file.read()

    print("File content:\n", content)

    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: "))

except ValueError:

    print("Error: Please enter 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("Sample.txt", "r")

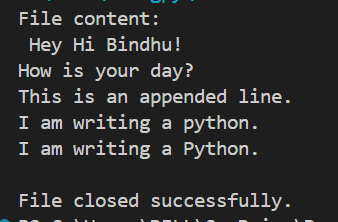
    content = file.read()

    print("File content:\n", content)

finally:

    file.close()

    print("File closed successfully.")



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:

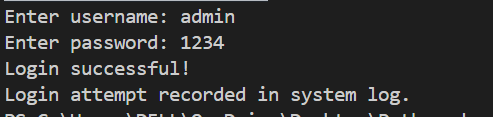
        raise ValueError("Invalid credentials")

except ValueError as e:

    print("Login failed:", e)

finally:

    print("Login attempt recorded in system log.")



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: "))

    result = a / b

    print("Result:", result)

except ZeroDivisionError:

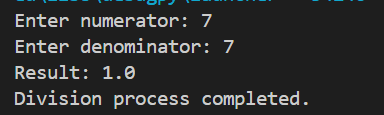
    print("Error: Cannot divide by zero.")

except ValueError:

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

finally:

    print("Division process completed.")



## 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 numerator: "))

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

    result = a / b

    print("Result:", result)

except ValueError:

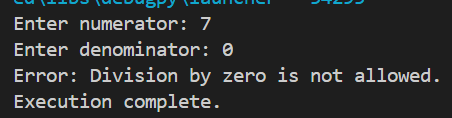
    print("Error: Please enter only numbers.")

except ZeroDivisionError:

    print("Error: Division by zero is not allowed.")

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.

try:

    balance = 1000

    amount = float(input("Enter amount to withdraw: "))

    if amount <= 0:

        raise ValueError("Amount must be greater than zero.")

    elif amount > balance:

        raise ValueError("Insufficient balance.")

except ValueError as e:

    print("Transaction failed:", e)

else:

    balance -= amount

    print(f"Withdrawal successful. Remaining balance: ₹{balance:.2f}")

finally:

    print("Transaction completed. Thank you for banking with us.")

