Experiment No. 14: Basic Exception Handling

AIM: Write a python program that takes two numbers as input and performs division. Implement exception handling to manage division by zero and invalid input error gracefully

THEORY:

1. Exception Handling (try-except-finally)

- try block: Contains code that might raise an exception.
- except block: Catches specific errors like ZeroDivisionError and ValueError.
- else block: Executes if no exception occurs.
- finally block: Runs regardless of whether an error occurs.

2. Error Handling in Python

- ZeroDivisionError: Occurs when dividing by zero.
- ValueError: Raised when a user enters non-numeric input.

ALGORITHM

1. **Start**
2. **Prompt the user** to enter two numbers.
3. **Try**:

   Convert inputs to float.

   Perform division: num1 / num2.

4. **Catch errors:**

   If division by zero occurs, print an error message.

   If input is not a valid number, print an error message.

5. **If no errors** occur, print the result.
6. **Finally**, display a completion message.
7. **End**

PROGRAM

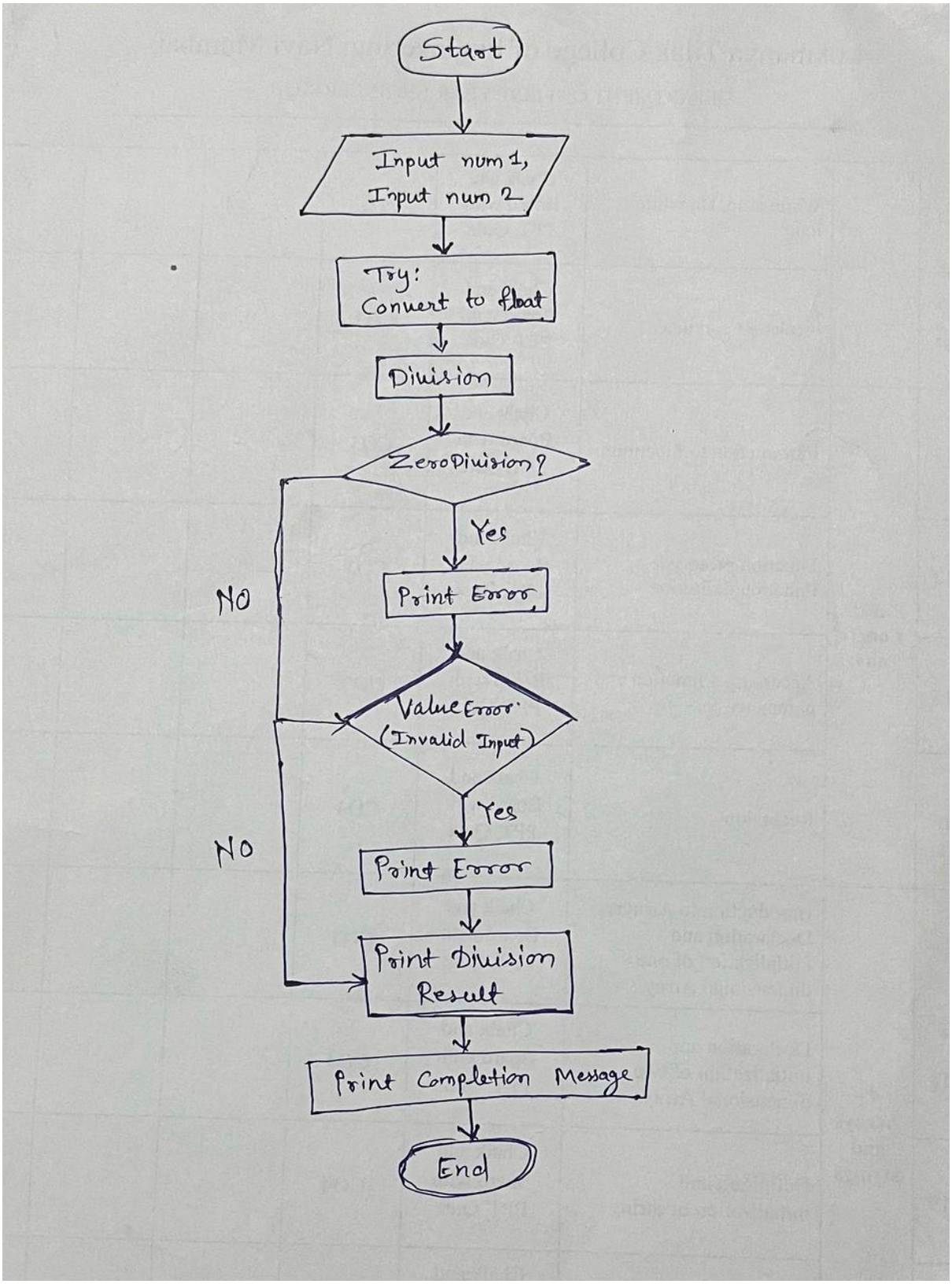
```
def safe_division():
    try:
        # Taking user input
        num1 = float(input("Enter the first number: "))
        num2 = float(input("Enter the second number: "))

        # Performing division
        result = num1 / num2

    except ZeroDivisionError:
        print("Error: Cannot divide by zero! Please enter a non-zero denominator.")
    except ValueError:
        print("Error: Invalid input! Please enter numeric values only.")
    else:
        print(f"Result: {result}")  # Only runs if no exception occurs
    finally:
        print("Program execution completed.")  # Runs always

# Calling the function
safe_division()
```

FLOWCHART:


Start
Input num 1,
Input num 2
Try:
Convert to float
Division
ZeroDivision?
Yes
NO
Print Error
ValueError
(Invalid Input)
Yes
NO
Print Error
Print Division
Result
Print Completion Message
End