# Exception Handling – Question Paper Aarthi P

23/06/2025

## 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.
2. **try:**
3. **num = int(input("Enter the numerator: "))**
4. **deno = int(input("Enter the denominator: "))**
5. **result = num / deno**
6. **print("Result:", result)**
7. **except ZeroDivisionError:**
8. **print("Error: Cannot divide by zero.")**
9. Write a program to convert a string to an integer. Handle ValueError if the input is not a valid number.
10. **input = input("Enter a number: ")**
11. **try:**
12. **number = int(input)**
13. **print("The integer value is:", number)**
14. **except ValueError:**
15. **print("Error: Invalid input. Please enter a valid integer.")**
16. Accept two numbers from the user and perform addition. Use try-except to handle invalid input types.
17. **try:**
18. **n1 = float(input("Enter the first number: "))**
19. **n2 = float(input("Enter the second number: "))**
20. **result = n1 + n2**
21. **print("The sum is:", result)**
22. **except ValueError:**
23. **print("Error: Invalid input. Please enter numeric values.")**
24. Write a program to read an element from a list using an index entered by the user. Handle IndexError.
25. **list = [10, 20, 30, 40, 50]**
26. **try:**
27. **index = int(input("Enter an index (0 to 4): "))**
28. **element = list[index]**
29. **print("Element at index", index, "is:", element)**
30. **except IndexError:**
31. **print("Error: Index out of range. Please enter a valid index.")**

## 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.
2. **try:**
3. **num = int(input("Enter a number: "))**
4. **except ValueError:**
5. **print("Error: Please enter a valid number.")**
6. **else:**
7. **print("The square of", num, "is:", num\*\*2)**
8. Write a program to open a file and read contents. Use try-except-else to handle FileNotFoundError.
9. **try:**
10. **file = open(“one.txt”, 'r')**
11. **except FileNotFoundError:**
12. **print("Error: File not found. Please check the file name.")**
13. **else:**
14. **content = file.read()**
15. **print("File contents:\n")**
16. **print(content)**
17. **file.close()**
18. Write a Python program to convert a number to its binary format. Use try-except-else to handle any invalid input.
19. **try:**
20. **num = int(input("Enter an integer: "))**
21. **except ValueError:**
22. **print("Error: Please enter a valid integer.")**
23. **else:**
24. **binary = bin(num)**
25. **print("Binary representation of", num, "is:", binary)**

## 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.
2. **try:**
3. **file = open(“one.txt”, 'r')**
4. **content = file.read()**
5. **print("File contents:\n")**
6. **print(content)**
7. **except FileNotFoundError:**
8. **print("Error: File not found.")**
9. **finally:**
10. **file.close()**
11. **print("File closed successfully.")**
12. 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.
13. **crt\_user = "admin"**
14. **crt\_pass = "1234"**
15. **try:**
16. **user = input("Enter username: ")**
17. **pass = input("Enter password: ")**
18. **if user == crt\_user and pass == crt\_pass:**
19. **print("Login successful!")**
20. **else:**
21. **raise ValueError("Invalid username or password.")**
22. **except ValueError as e:**
23. **print("Login failed:", e)**
24. **finally:**
25. **print("Login attempt finished. Logging event...")**
26. Write a program that divides two numbers, catching errors with try-except, and printing a clean-up message using finally.

**try:**

**num = int(input("Enter the numerator: "))**

**deno = int(input("Enter the denominator: "))**

**result = num / deno**

**print("Result:", result)**

**except ZeroDivisionError:**

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

**finally:**

**print(“Operation complete. Cleaning up…”)**

## 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:**

**num = int(input("Enter the numerator: "))**

**deno = int(input("Enter the denominator: "))**

**result = num / deno**

**print("Result:", result)**

**except ZeroDivisionError:**

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

**except ValueError:**

**print("Error: Please enter valid numeric values.")**

**finally:**

1. **print(“Execution complete”)**
2. 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.
3. **bal = 5000**
4. **try:**
5. **amt = float(input("Enter amount to withdraw: "))**
6. **if amt <= 0:**
7. **raise ValueError("Withdrawal amount must be positive.")**
8. **if amt > bal:**
9. **raise ValueError("Insufficient balance.")**
10. **except ValueError as e:**
11. **print("Transaction failed:", e)**
12. **else:**
13. **bal -= amt**
14. **print(f"Withdrawal of ₹{amt} successful. balance: ₹{bal}")**
15. **finally:**
16. **print("Transaction process complete.")**