# 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:

    num1 = float(input("Enter the numerator: "))

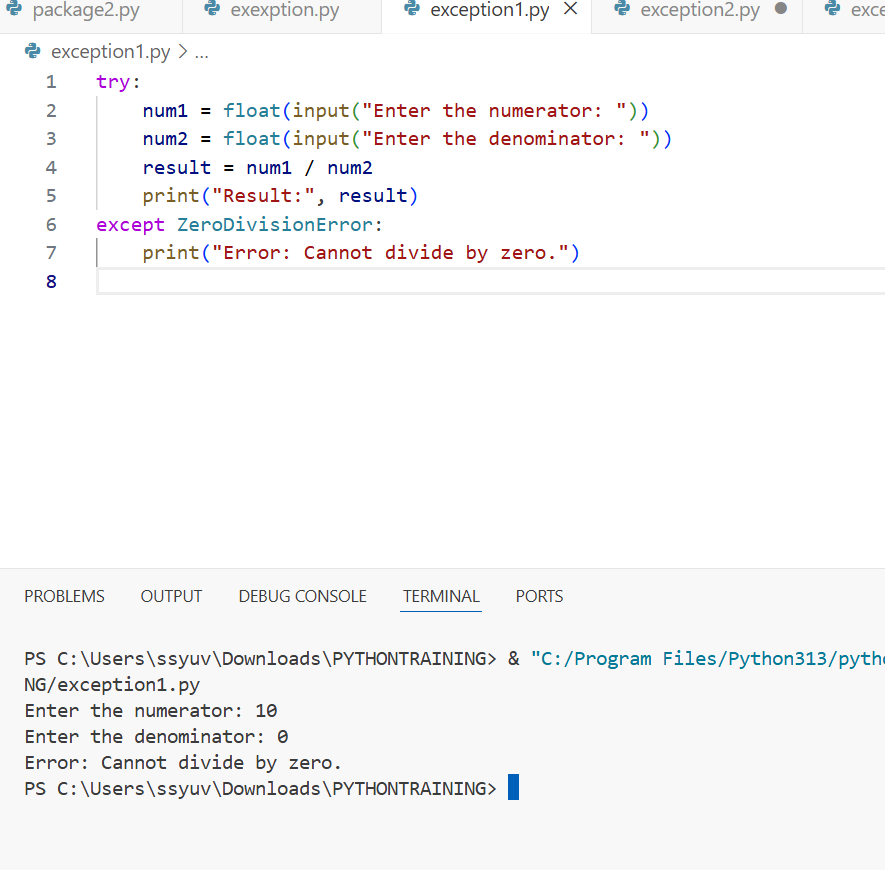
num2 = float(input("Enter the denominator: "))

result = num1 / num2

    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:

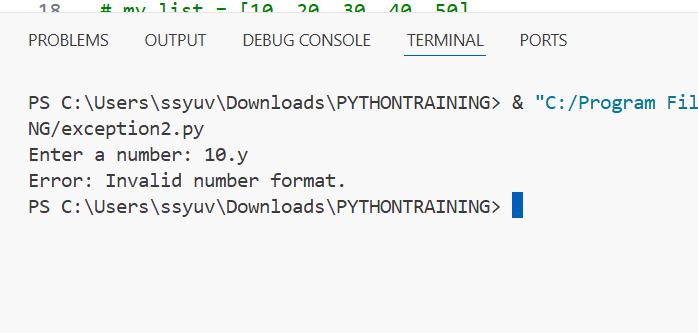
    user\_input = input("Enter a number: ")

    number = int(user\_input)

    print("Integer:", number)

except ValueError:

    print("Error: Invalid number format.")



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

try:

    num1 = float(input("Enter first number: "))

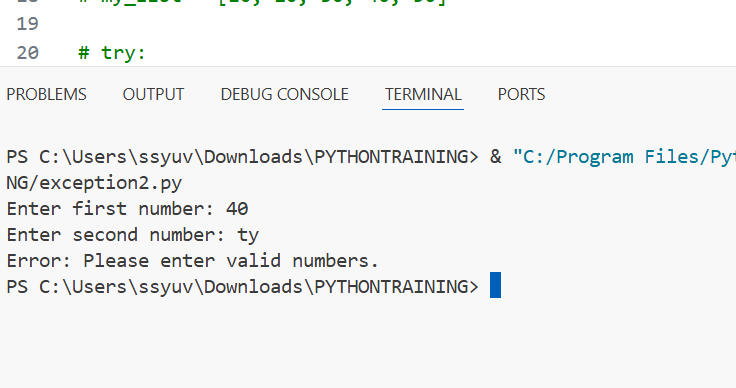
  num2 = float(input("Enter second number: "))

  total = num1 + num2

    print("Sum:", total)

except ValueError:

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



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("Enter the index (0-4): "))

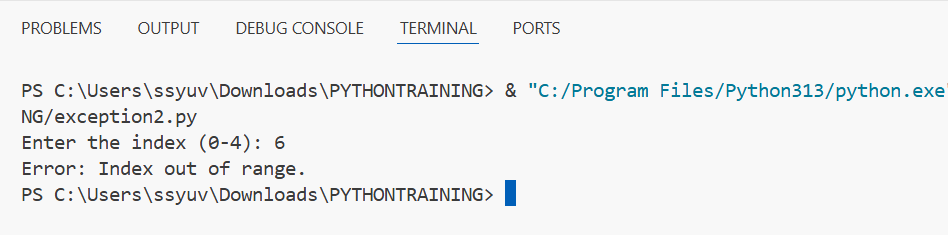
    print("Element at index", index, "is", my\_list[index])

except IndexError:

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

except ValueError:

    print("Error: Please enter a valid integer 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.

try:

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

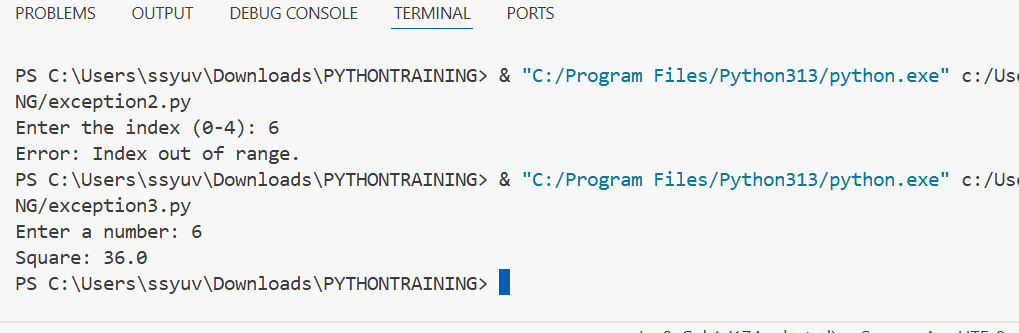
except ValueError:

    print("Error: Invalid number format.")

else:

    square = num \*\* 2

    print("Square:", square)



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

    file\_name = input("Enter the filename: ")

    f = open(file\_name, 'r')

except FileNotFoundError:

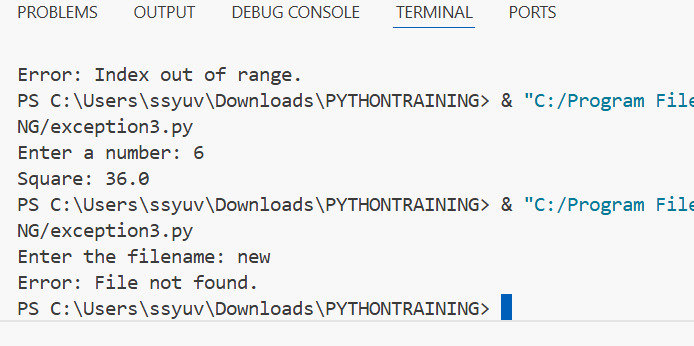
    print("Error: File not found.")

else:

    content = f.read()

    print("File content:\n", content)

  f.close()



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

try:

    number = int(input("Enter an integer: "))

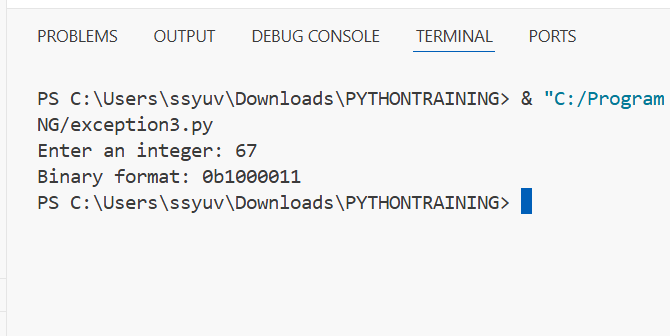
except ValueError:

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

else:

    binary = bin(number)

    print("Binary format:", 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.

try:

    file = open("example.txt", "r")

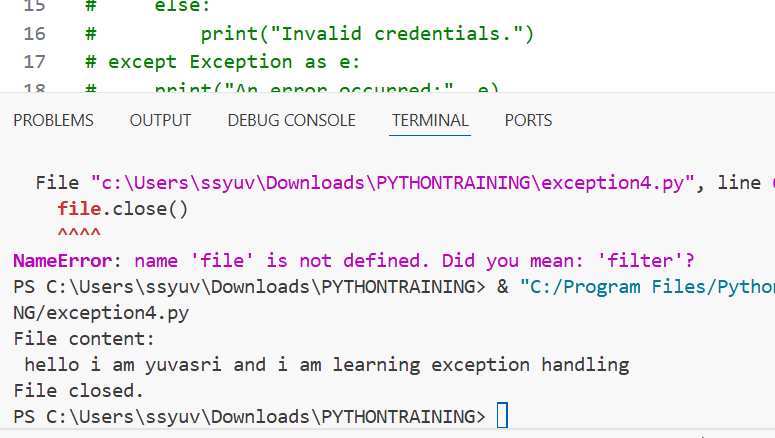
    content = file.read()

    print("File content:\n", content)

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:

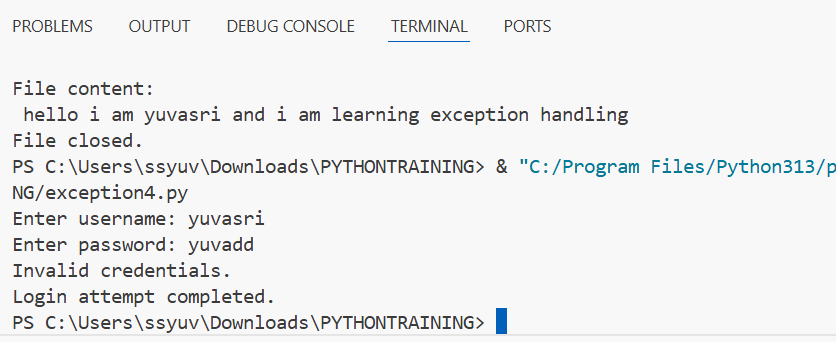
        print("Invalid credentials.")

except Exception as e:

    print("An error occurred:", e)

finally:

    print("Login attempt completed.")



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

try:

    num1 = float(input("Enter numerator: "))

    num2 = float(input("Enter denominator: "))

    result = num1 / num2

    print("Result:", result)

except ZeroDivisionError:

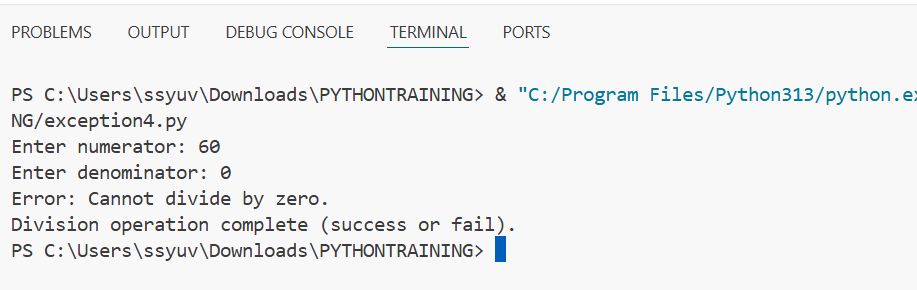
    print("Error: Cannot divide by zero.")

except ValueError:

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

finally:

    print("Division operation complete (success or fail).")



## 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:

  num1 = float(input("Enter numerator: "))

  num2 = float(input("Enter denominator: "))

    result = num1 / num2

    print("Result:", result)

except ZeroDivisionError:

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

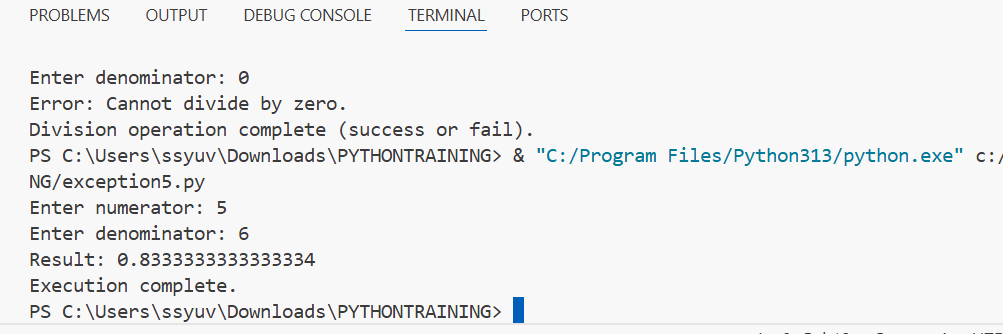
except ValueError:

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

finally:

    print("Execution complete.")

balance = 1000.0



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 = 1000.0

try:

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

    if amount <= 0:

        raise ValueError("Amount must be positive.")

    if amount > balance:

        raise ValueError("Insufficient balance.")

except ValueError as ve:

    print("Transaction failed:", ve)

else:

    balance -= amount

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

finally:

    print("Transaction process completed.")

