**Lab Assignment 8: Error and Exception Handling**

You are working on a project to build a custom text processing tool that reads input from various sources, processes the text data, and stores the results in an output file. As part of this project, you need to implement a robust exception handling mechanism to handle potential errors that may arise during the text processing.

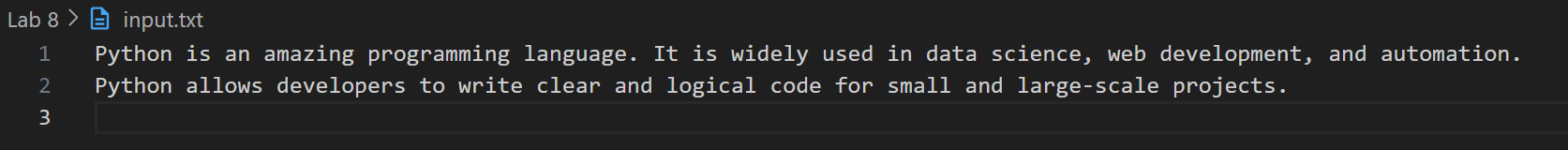
The tool needs to perform the following steps:

1. Read the input data from a file specified by the user.
2. Process the text data by performing various operations, such as counting words, calculating character frequencies, and generating word clouds.
3. Store the processed results in an output file.

Your task is to design a Python program that incorporates appropriate exception handling to handle the following situations:

1. File Not Found Error: If the user provides an invalid file path or the input file is not found, your program should raise a custom exception FileNotFoundError with a suitable error message.
2. Invalid Input Data: During text processing, if any unexpected input data is encountered (e.g., non-string values or missing data), your program should raise a custom exception InvalidInputDataError with relevant details.
3. Disk Space Full: If the output file cannot be written due to insufficient disk space, your program should raise a custom exception DiskSpaceFullError.

Data Set –



Code –

import os

# Custom exceptions

class FileNotFoundErrorCustom(Exception):

    pass

class InvalidInputDataError(Exception):

    pass

class DiskSpaceFullError(Exception):

    pass

def read\_input\_file(file\_path):

    """Reads the input file specified by the user."""

    if not os.path.exists(file\_path):

        raise FileNotFoundErrorCustom(f"File '{file\_path}' not found. Please provide a valid path.")

    try:

        with open(file\_path, 'r') as file:

            data = file.read()

    except Exception as e:

        raise InvalidInputDataError(f"An error occurred while reading the file: {e}")

    if not isinstance(data, str) or len(data.strip()) == 0:

        raise InvalidInputDataError("Input data is not valid. Expected non-empty text data.")

    return data

def process\_text(data):

    """Processes the text by counting words and calculating character frequencies."""

    if not isinstance(data, str):

        raise InvalidInputDataError("Invalid input data type. Expected a string.")

    # Word count

    words = data.split()

    word\_count = len(words)

    # Character frequency

    char\_freq = {}

    for char in data:

        if char.isalpha():

            char\_freq[char] = char\_freq.get(char, 0) + 1

    return word\_count, char\_freq

def save\_results(output\_file, word\_count, char\_freq):

    """Saves the processed results to an output file."""

    try:

        with open(output\_file, 'w') as file:

            file.write(f"Word Count: {word\_count}\n")

            file.write(f"Character Frequency:\n")

            for char, freq in char\_freq.items():

                file.write(f"{char}: {freq}\n")

    except OSError as e:

        if 'No space left on device' in str(e):

            raise DiskSpaceFullError("Failed to write output file due to insufficient disk space.")

        else:

            raise

def main():

    try:

        input\_file = "./Lab 8/input.txt"

        data = read\_input\_file(input\_file)

        # Process the text

        word\_count, char\_freq = process\_text(data)

        # Save results to output file

        output\_file = 'output\_results.txt'

        save\_results(output\_file, word\_count, char\_freq)

        print(f"Processing complete. Results saved to {output\_file}.")

    except FileNotFoundErrorCustom as e:

        print(f"Error: {e}")

    except InvalidInputDataError as e:

        print(f"Error: {e}")

    except DiskSpaceFullError as e:

        print(f"Error: {e}")

    except Exception as e:

        print(f"An unexpected error occurred: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Output –

