Python programming Lab(23CP301P)

Name: Krishika Vansh **Semester: V**

Roll No: 23BCP448 **Faculty: Mr. Davinder Singh**

Division: VII Batch G13

Branch: Computer Engineering

School of Technology

November 2025

Experiment No: 6

Error and Exception Handling in Python Programs

Objective: To build a robust Python-based text processing tool that can read input from a user-specified file, process the content, and write results to an output file using custom exception handling mechanisms to ensure fault tolerance and graceful error reporting.

Code:

```
import os

import shutil


class InvalidInputDataError(Exception):

    def __init__(self, message="Invalid input data."):

        super().__init__(message)


class DiskSpaceFullError(Exception):

    def __init__(self, message="Not enough disk space to save the file."):

        super().__init__(message)


class CustomFileNotFoundError(Exception):

    def __init__(self, message="Input file not found."):

        super().__init__(message)


def read_input_file(file_path):

    if not os.path.exists(file_path):

        raise CustomFileNotFoundError(f"Error: File '{file_path}' not found.")
```

```
    with open(file_path, "r") as f:
        content = f.read()
    if not content.strip():
        raise InvalidInputDataError("Error: Input file is empty or contains invalid data.")
    return content


def process_word_counts(content):
    words = content.split()
    if not words:
        raise InvalidInputDataError("Error: No valid words found in the file.")
    word_count = {}
    for word in words:
        word = word.lower().strip(",.!?;:\"'()[]{}")
        if word:
            word_count[word] = word_count.get(word, 0) + 1
    return word_count


def process_char_counts(content):
    char_count = {}
    for char in content:
        if char.isalnum():
            char = char.lower()
            char_count[char] = char_count.get(char, 0) + 1
    return char_count
```

```
def save_output(output_file, word_count, char_count):
    try:
        total, used, free = shutil.disk_usage(".")
        if free < 1024:  # less than 1 KB free
            raise DiskSpaceFullError()
        with open(output_file, "w") as f:
            f.write("=== Word Frequency ===\n")
            for word, count in word_count.items():
                f.write(f"{word}: {count}\n")
            f.write("\n=== Character Frequency ===\n")
            for char, count in char_count.items():
                f.write(f"{char}: {count}\n")
    except OSError as e:
        raise DiskSpaceFullError(f"Disk write error: {e}")


def main():
    input_file = "input6.txt"
    output_file = "output6.txt"
    try:
        content = read_input_file(input_file)
        word_count = process_word_counts(content)
        char_count = process_char_counts(content)
        save_output(output_file, word_count, char_count)
        print(f"Analysis saved to '{output_file}' successfully.")
```

```
    except CustomFileNotFoundError as e:
        print(e)
    except InvalidInputDataError as e:
        print(e)
    except DiskSpaceFullError as e:
        print(e)
    except Exception as e:
        print(f"Unexpected Error: {e}")


if __name__ == "__main__":
    main()
```

input6.txt

```
Python is simple and powerful.
Python is fun to learn.
Error handling makes Python programs reliable.
```

Output:

```
[Running] python -u "c:\Users\Admin\OneDrive\D
Analysis saved to 'output6.txt' successfully.

[Done] exited with code=0 in 0.11 seconds
```

output6.txt

```
=== Word Frequency ===
python: 3
is: 2
simple: 1
and: 1
powerful: 1
fun: 1
to: 1
learn: 1
error: 1
handling: 1
makes: 1
programs: 1
reliable: 1

=== Character Frequency ===
p: 6
y: 3
t: 4
h: 4
o: 7
n: 8
i: 5
s: 5
m: 3
l: 6
e: 7
a: 6
d: 2
w: 1
r: 8
```