

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.")
```

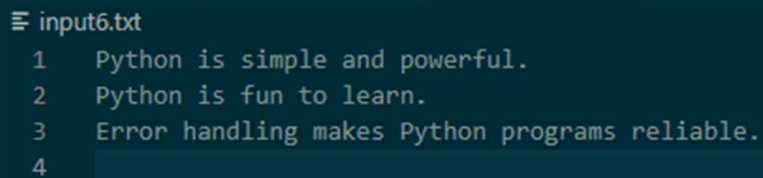
```
with open(file_path, "r") as f:
    content = f.read()
    if not content.strip():
        raise InvalidInputDataError("Error: Input file is empty or contains invalid data.")
    return content

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
```

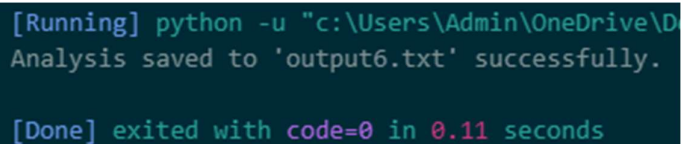
```
def save_output(output_file, word_count, char_count):  
    try:  
        total, used, free = shutil.disk_usage("/")  
        if free < 1024: # less than 1 KB free  
            raise DiskSpaceFullError()  
        with open(output_file, "w") as f:  
            f.write("=== Word Frequency ===\n")  
            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()
```



A screenshot of a text editor window titled 'input6.txt'. The window contains four lines of text: '1 Python is simple and powerful.', '2 Python is fun to learn.', '3 Error handling makes Python programs reliable.', and '4' followed by a blank line.

Output:



A screenshot of a terminal window showing the execution of a Python script. The first line shows '[Running] python -u "c:\Users\Admin\OneDrive\D' followed by 'Analysis saved to \'output6.txt\' successfully.' The second line shows '[Done] exited with code=0 in 0.11 seconds'.

```
≡ output6.txt
1  === Word Frequency ===
2  python: 3
3  is: 2
4  simple: 1
5  and: 1
6  powerful: 1
7  fun: 1
8  to: 1
9  learn: 1
10 error: 1
11 handling: 1
12 makes: 1
13 programs: 1
14 reliable: 1
15
16 === Character Frequency ===
17 p: 6
18 y: 3
19 t: 4
20 h: 4
21 o: 7
22 n: 8
23 i: 5
24 s: 5
25 m: 3
26 l: 6
27 e: 7
28 a: 6
29 d: 2
30 w: 1
31 r: 8
```