**KONGU ENGINEERING COLLEGE**

(AUTONOMOUS), PERUNDURAI – 638060



**DEPARTMENT OF ELECTRONICS AND INSTRUMENTATION ENGINEERING**

# PROGRAMMING IN PYTHON - 22ITC41

**MINI PROJECT – TEXT FILE WORD COUNTER**

**TEAM MEMBERS:** HARI PRASANNA S (23EIR036)

HARINI M (23EIR039)

MADHUMITHA S J (23EIR057)

**TEXT FILE WORD COUNTER**

**AIM:**

The aim of this project is to develop a simple yet efficient application that reads a text file and accurately counts the number of words it contains. The project is designed to demonstrate file handling, string manipulation, and basic algorithm implementation. It provides a foundational understanding of how text processing tools work, which are commonly used in data analysis, natural language processing, and other software applications.

**TECHNICAL SPECIFICATIONS:**

This project was developed using the following setup:

* **Software Used**: Python IDLE (Version 3.x)

Python IDLE provides a simple environment for writing and running Python code, ideal for beginners and small-scale applications.

* **Programming Language**: Python

Python was chosen for its simplicity, readability, and strong support for object-oriented programming.

* **Execution Environment**: Standard Python Shell

The program runs in the default Python shell without using any external libraries or GUI frameworks, ensuring clarity and focus on core programming concepts.

**SYSTEM OVERVIEW:**

The **Text File Word Counter** system is a lightweight application designed to process plain text files and compute the total number of words contained within them. The system takes a text file as input, reads its contents, and applies parsing techniques to identify and count individual words.

**KEY FUNCTIONALITIES:**

* **Input Module:** Accepts a text file from the user or a predefined location.
* **File Reader:** Opens and reads the contents of the file.
* **View Booking History:** Processes the content, identifies word boundaries, and counts the number of words accurately.
* **Output Module**: Displays the word count result to the user.

**SYSTEM ARCHITECTURE:**

The application is structured using a simple procedural approach with a single core function:

* **count\_words\_in\_file Function**:  
  This function handles the entire workflow of the application. It opens the specified text file, reads its contents, splits the text into individual words based on whitespace, and counts them. It also includes error handling to manage cases such as missing files or unexpected issues during reading.

**CODE:**

def count\_words\_in\_file(filename):

try:

with open(filename, 'r', encoding='utf-8') as file:

text = file.read()

words = text.split()

word\_count = len(words)

print(f"Total words in '{filename}': {word\_count}")

except FileNotFoundError:

print(f"File '{filename}' not found.")

except Exception as e:

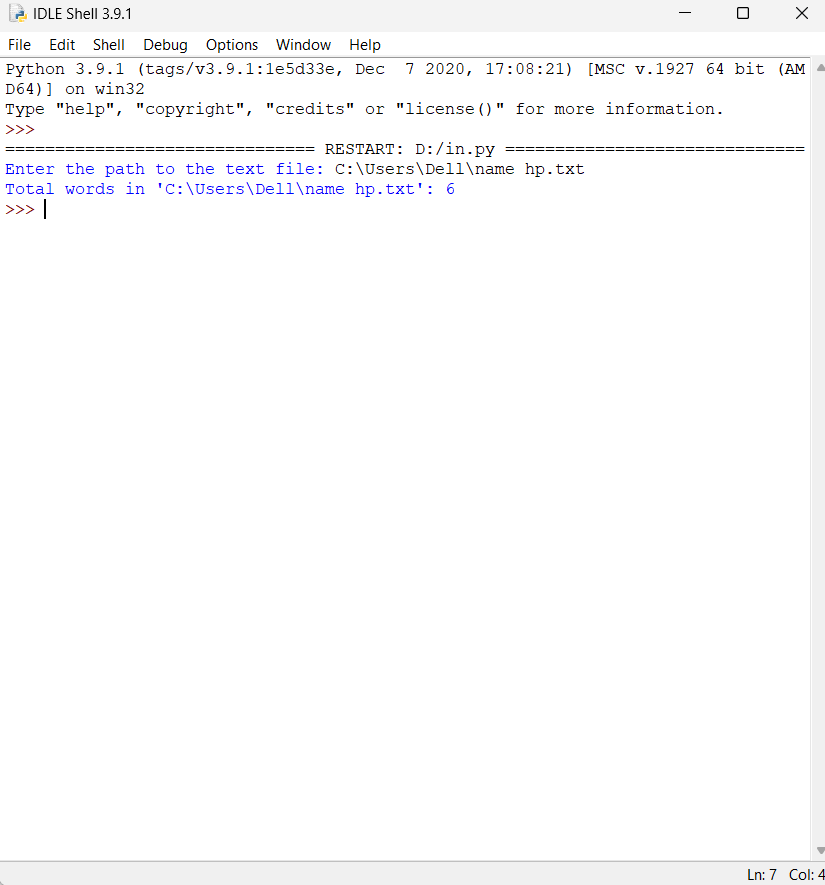
print(f"An error occurred: {e}")

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

filename = input("Enter the path to the text file: ")

count\_words\_in\_file(filename)

**OUTPUT:**



**CONCLUSION**

The **Text File Word Counter** project successfully demonstrates the use of file handling and basic string operations in Python to count the number of words in a text file. The project highlights how simple programming techniques can be used to perform useful text processing tasks. It is efficient, easy to use, and provides accurate results for basic word counting requirements. This project serves as a solid foundation for further development in the areas of text analysis and natural language processing.

**LIMITATIONS:**

* The program treats any whitespace-separated string as a word, which may lead to inaccuracies with punctuation or special characters.
* It does not differentiate between meaningful words and symbols or numbers.
* The tool lacks a graphical user interface (GUI), which may limit user accessibility.
* It processes only plain .txt files; other formats like .docx or .pdf are not supported.
* There is no support for advanced features like counting word frequency, ignoring stop words, or handling multiple files.

**SUGGESTIONS FOR IMPROVEMENT**

* Implement advanced text cleaning to exclude punctuation and non-alphabetic characters from word counts.
* Add functionality to count lines, characters, or word frequency.
* Incorporate a graphical user interface (GUI) to enhance user interaction.
* Extend support for additional file formats such as .pdf or .docx.
* Optimize the program for handling large files more efficiently.