

**A MINI PROJECT REPORT ON**

**File Handling Application**

Institute Name: Besant Technologies

Submitted By: Asmitha S

Submitted To: Gowthami Shyam (Trainer)

Coordinator: Pavimithra

Submission Date: 25 - 09 - 2025

**ACKNOWLEDGEMENT**

I would like to express my sincere gratitude to **Besant Technologies** for giving me this opportunity to work on the mini project titled **“File Handling Application using Python”**.

I am especially thankful to **Mr. Mohamed Ismail, General Manager** of **Besant Technologies**, for his vision and valuable contribution during the program.

I am deeply thankful to **Mrs. Gowthami Shyam**, my **Python Trainer**, for her guidance and for delivering the training sessions with clarity and in an understandable manner.

I would also like to thank **Ms. Pavimithra**, **Coordinator** of Python Programming Training, for her efficient coordination and support throughout the program.

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **Table of contents** | **Page No** |
| Acknowledgement | 2 |
| Introduction to Python | 4 |
| Concept used in project | 5 |
| Source code | 7 |
| Explanation of the source code | 10 |
| Output | 12 |
| Conclusion | 16 |
| Bibliography | 17 |
| GitHub link | 18 |

**Introduction to Python**

Programming language is a set of instructions used to communicate with machines. Python is a high-level programming language. High-level refers to the human understandable format.

Python was created by Guido Van Rossum in 1989. But it was introduced in 1999, at that time it has the version 0.99. The latest version of Python is 3.13.5. Python 0.99 to 2.0 versions are mostly similar. But the Python 3 version is different from the old version.

**Why Python?**

* Syntax is simple.
* Python has a smaller number of codes.
* We can easily solve complex problems using Python code easily.

**Advantages**

* East to learn, understand and write the code.
* Python is an interpreted language (Do line by line execution).
* It is a cross-platform language (No need to worry about the Operating System).
* It is an extensible language (Can modified).
* Python is a platform independent language (No change in data transformation).
* Python has large standard libraries (200+).
* It is also a dynamically typed language (No need to declare the variable).

**Concept used in project**

This file handling application created using the core concepts in Python file handling.

* **Modules**

The “os” module in Python is a built-in standard library module that provides a portable way of interacting with the operating system. It offers functions for performing various operating system-dependent tasks, including file system operations.

* **Functions**
* Built-in Functions - These are pre-defined functions provided by Python's standard library, readily available for use like print(), int(), input().
* User-defined Functions - These are functions created by programmers to address specific requirements within their applications like crete\_file(), insert\_text(), append\_file().
* **Conditional statements**

Conditional statements in Python enable programs to make decisions and execute different blocks of code based on whether a specified condition is true or false. The primary conditional statements are if, elif, and else.

* **User input**

In Python, the input() function is used to obtain user input from the console. This function pauses the program's execution and waits for the user to type something and press Enter.

* **Loops**

Loops in Python are control flow statements used to repeatedly execute a block of code. Python primarily offers two types of loops, for loops and while loops.

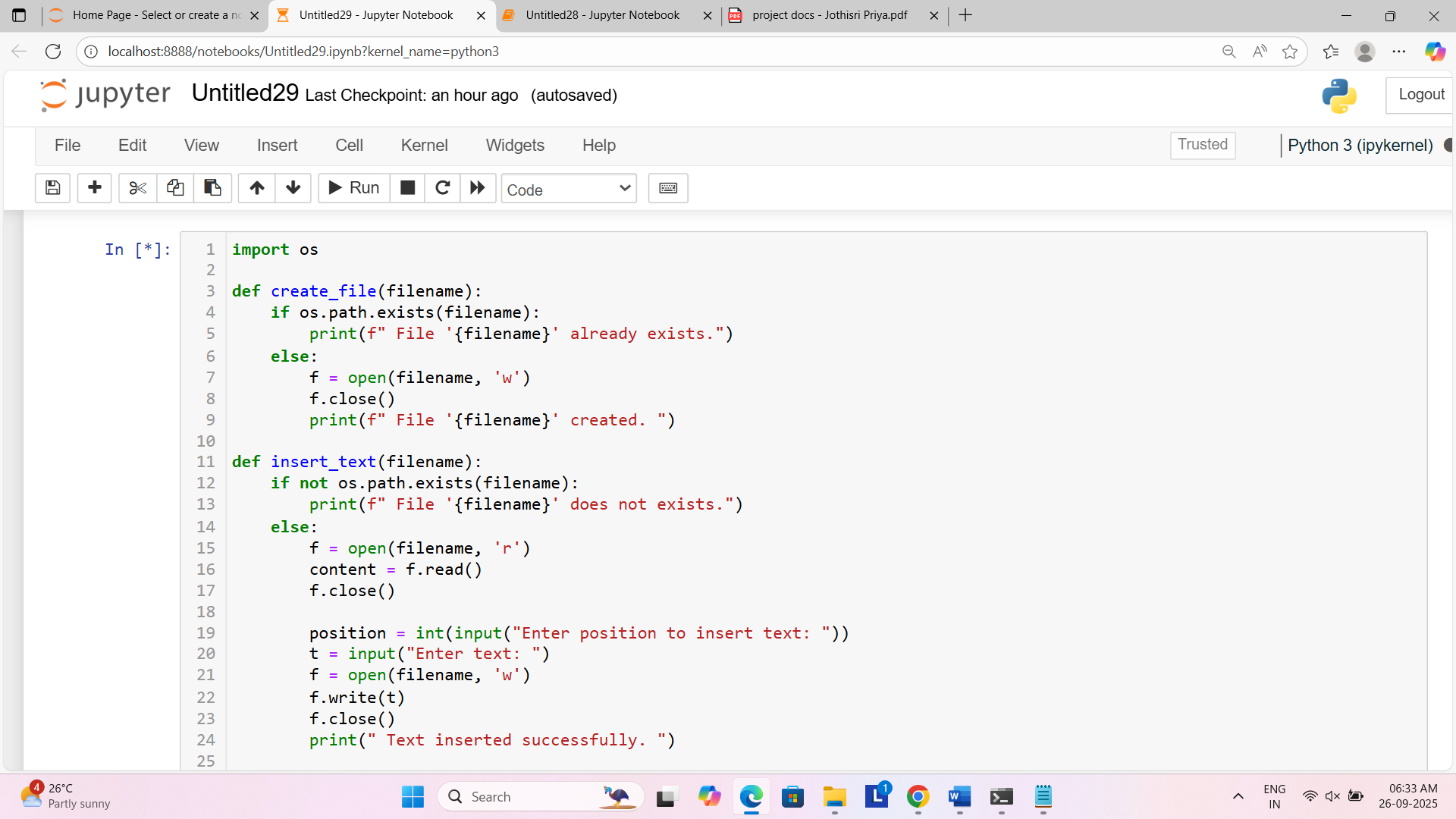
* **Exception handling**

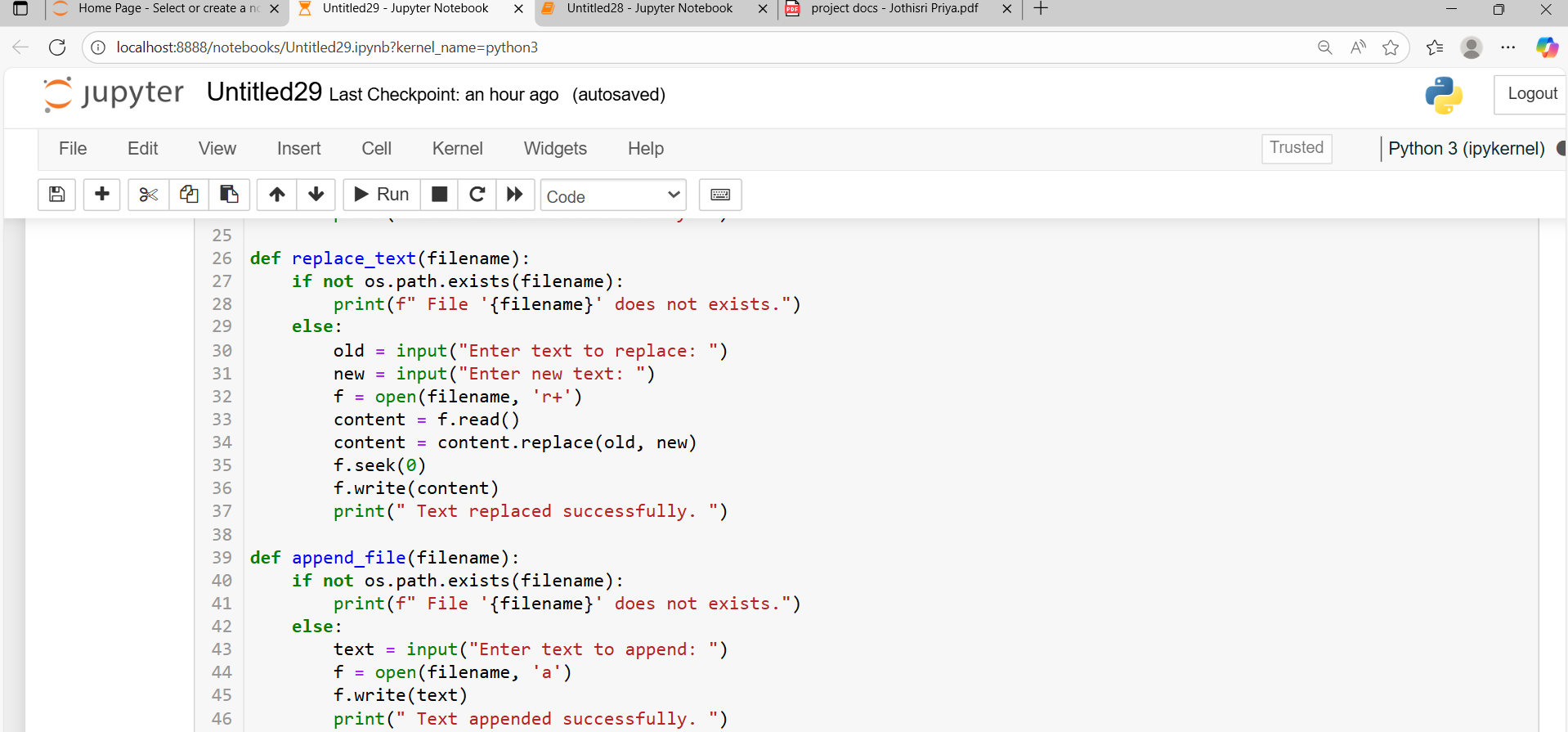
Exception handling in Python provides a structured way to manage runtime errors, preventing program crashes and allowing for graceful recovery. It utilizes a try-except block structure, optionally extended with else and finally clauses.

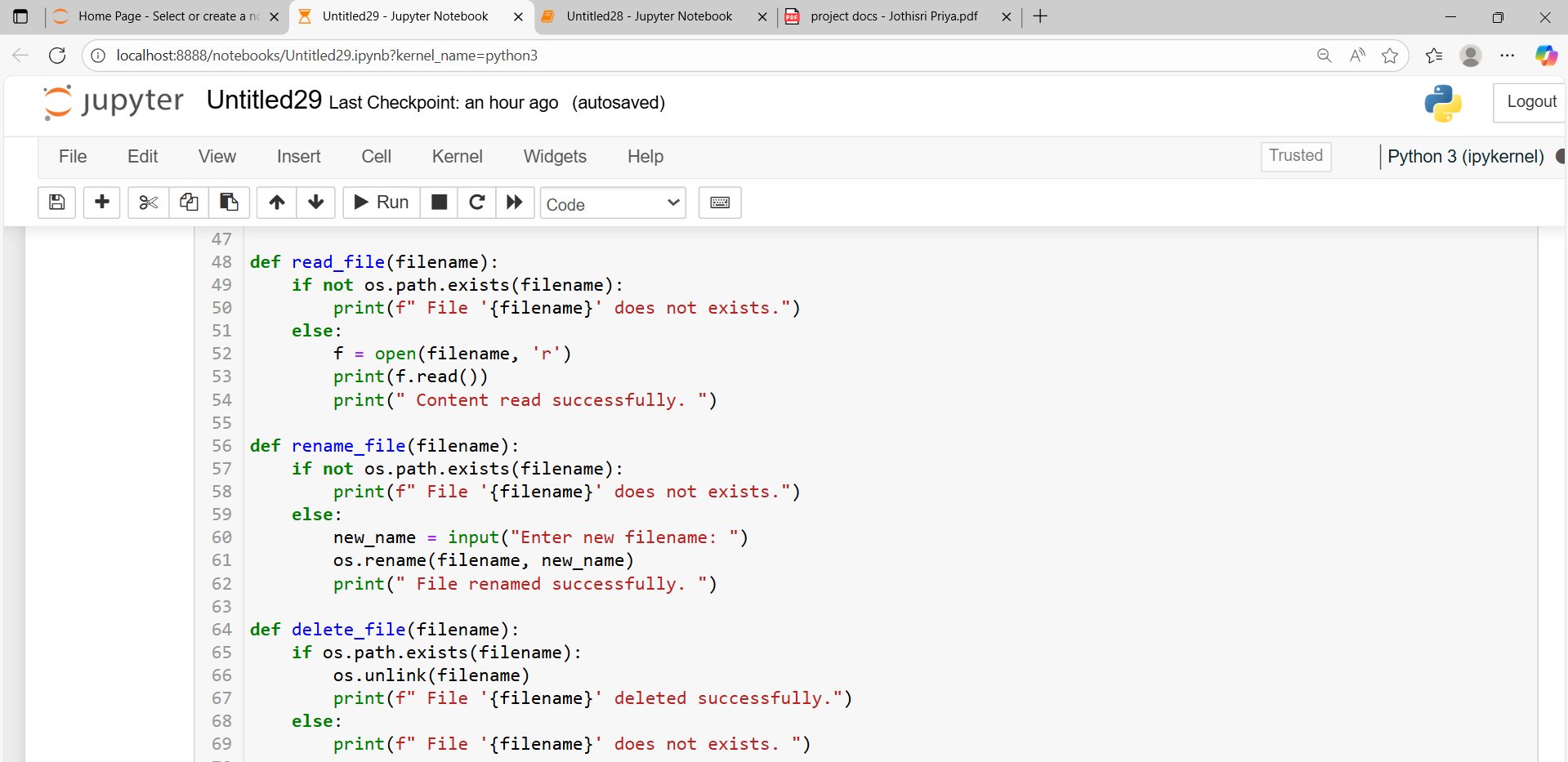
* **File handling**

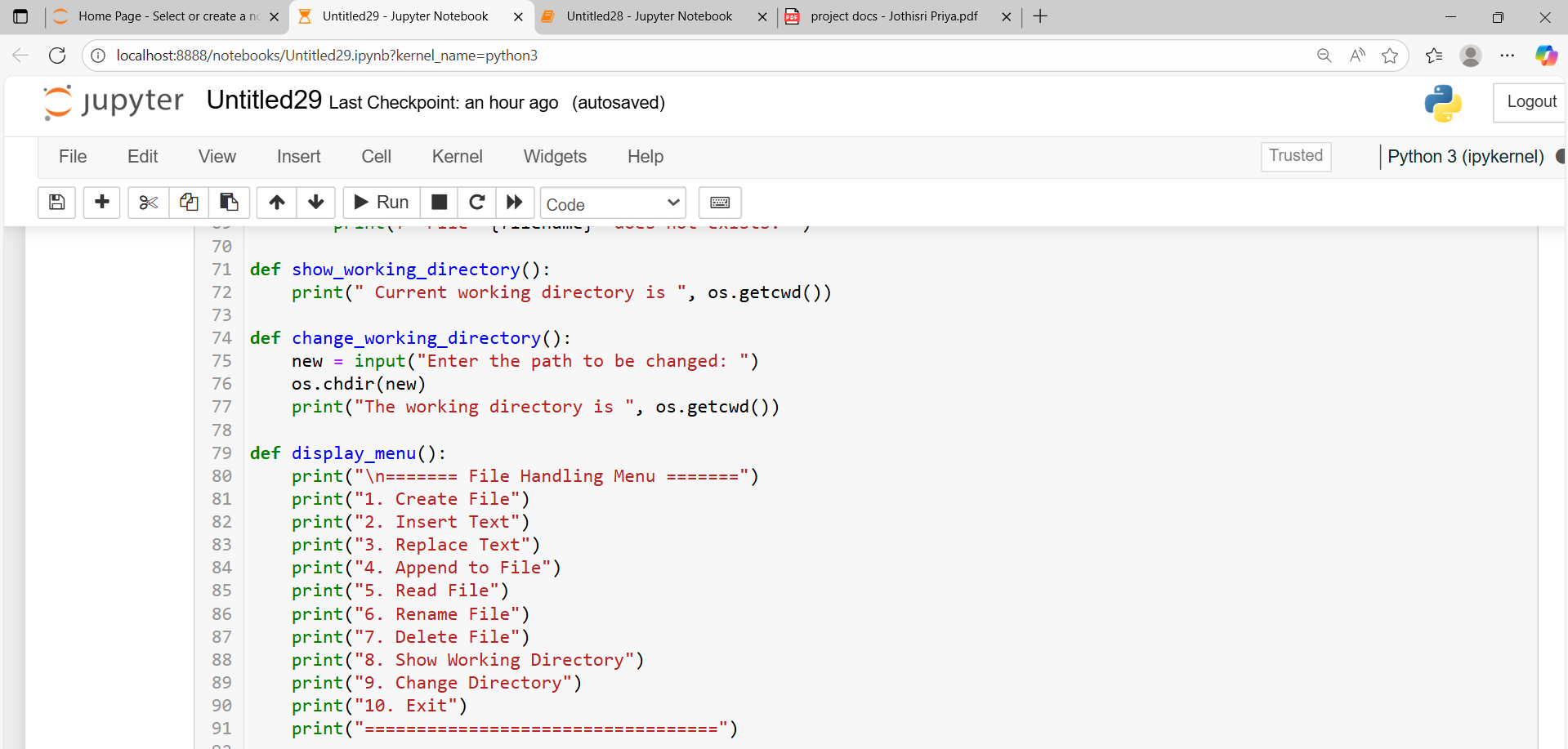
File handling in Python involves interacting with files stored on a computer's file system. This allows for operations such as reading data from a file, writing data to a file, and appending data to an existing file.

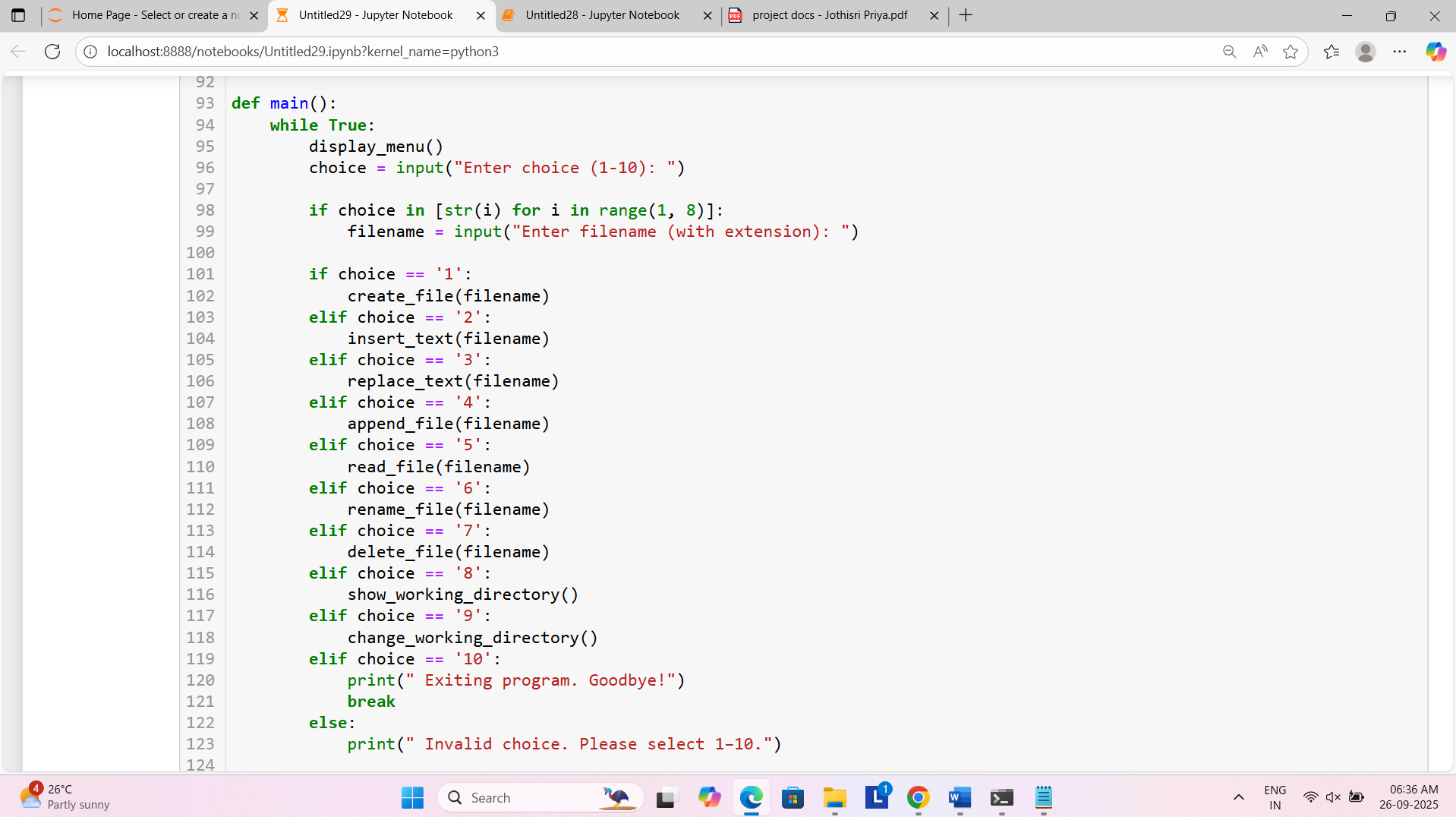
**Source code**

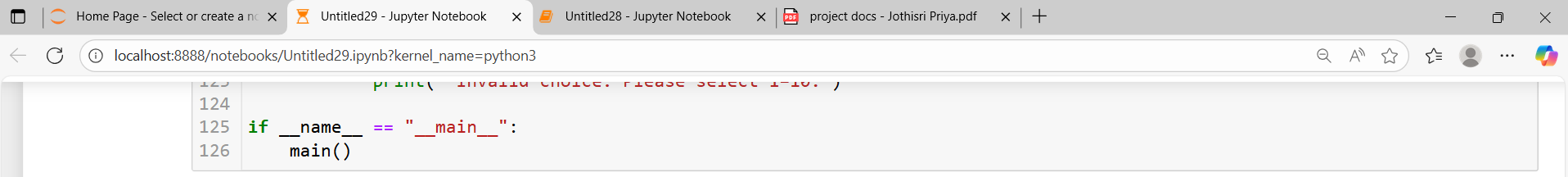


****

****

****

****

****

**Explanation of the source code**

* Imports the os module, which provides functions to interact with the operating system.
* In create\_file(filename), Checks if the file exists using os.path.exists(filename).If it exists, informs the user. If not, it creates a new empty file in write mode 'w'. File is closed explicitly here.
* In insert\_text(filename), Checks if the file exists. Reads the existing content of the file. Asks the user where to insert the new text. Asks for the text to insert. The new text is combining with the original content.
* In replace\_text(filename), Checks if the file exists. Asks for old and new text. Reads the content using 'r+' mode (read + write). Replaces all instances of old text with the new one. seek(0) moves the file pointer back to the beginning before writing.
* In append\_file(filename), Checks if file exists. Gets text input from the user. Opens file in append mode ('a'), which adds content to the end without erasing existing content.
* In read\_file(filename), Opens the file in read mode ('r'). Prints its contents to the screen.
* In rename\_file(filename), Checks if the file exists. Gets the new name from the user. Renames it using os.rename().
* In delete\_file(filename), If file exists, deletes it using os.unlink(). If it doesn’t exist, shows a message.
* show\_working\_directory(), Shows the current directory using os.getcwd().
* In change\_working\_directory(), Takes a new path from the user. Uses os.chdir() to change the current working directory. Prints the updated directory.
* display\_menu(), Displays the available options in a menu format.
* In the main(), Infinite loop to keep the program running. Displays menu and asks for user choice. If the choice is between 1–7, it prompts for a filename. Executes the corresponding file operation. If the choice is '10', it exits the loop and ends the program.
* The entry point check,

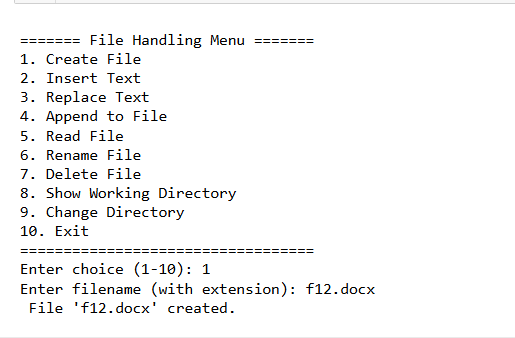
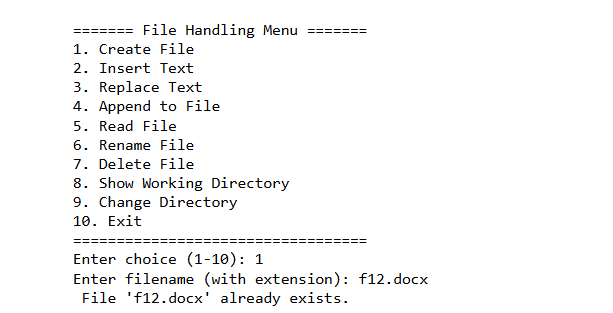
if \_\_name\_\_ == "\_\_main\_\_":

main()

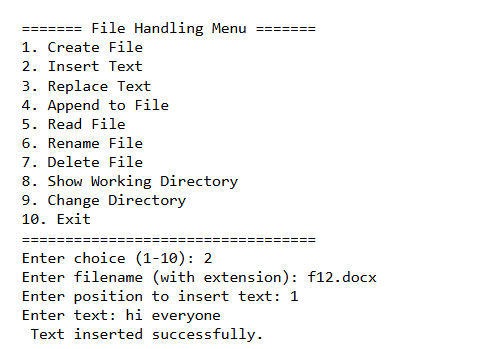
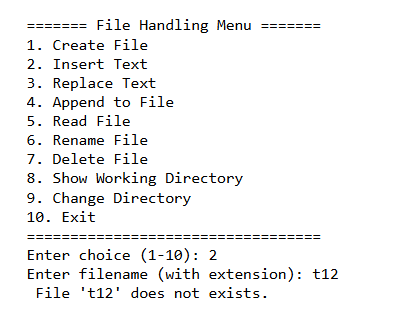
Ensures main() runs only when the script is run directly, not when imported as a module.

**Output**

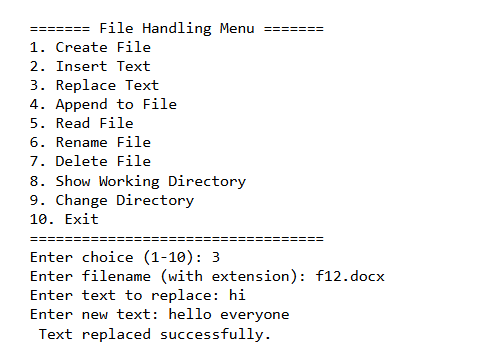
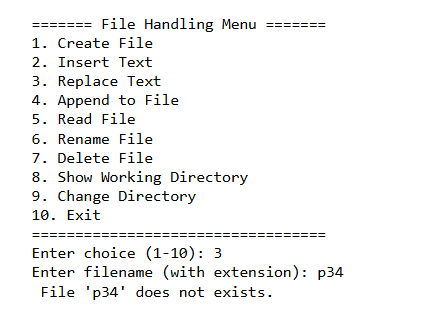
1. **Creating a file**

** **

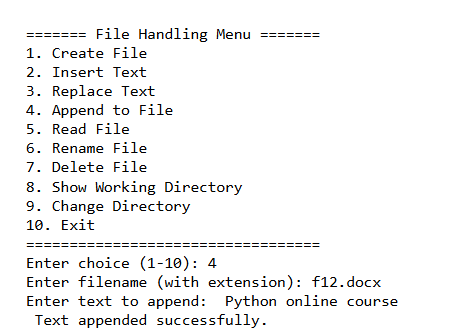
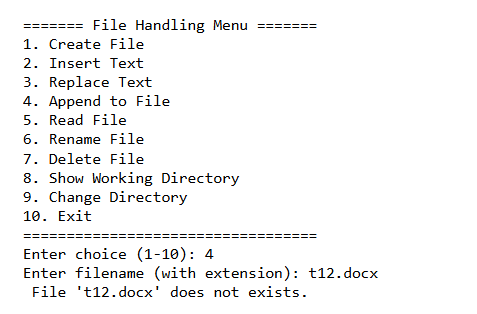
1. **Insert text**

** **

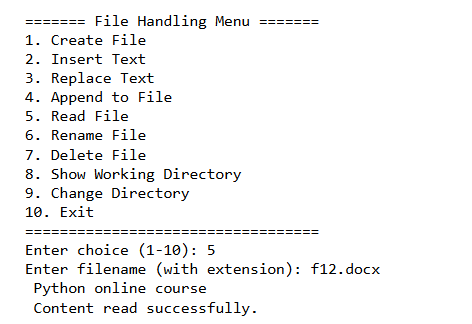
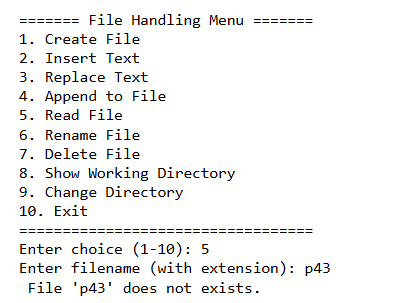
1. **Replace text**

** **

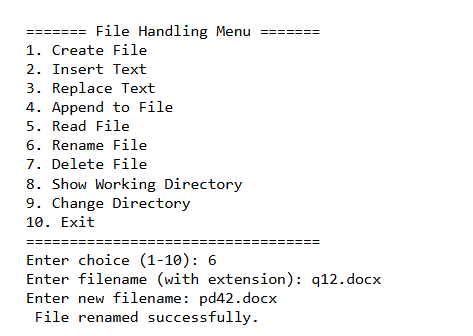
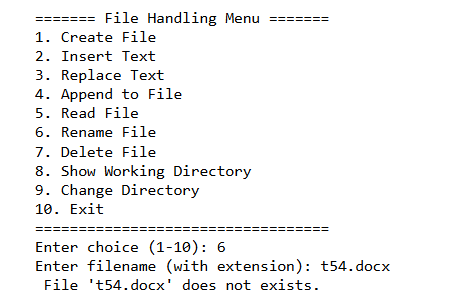
1. **Append text**

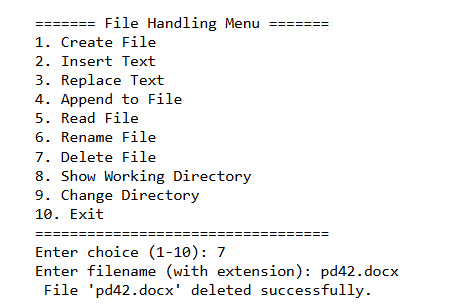
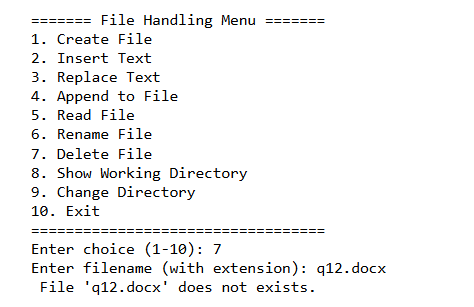
1. **Read file**

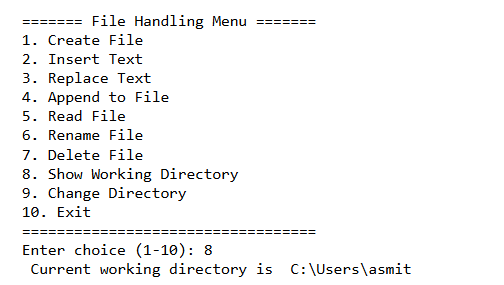
1. **Rename file**

** **

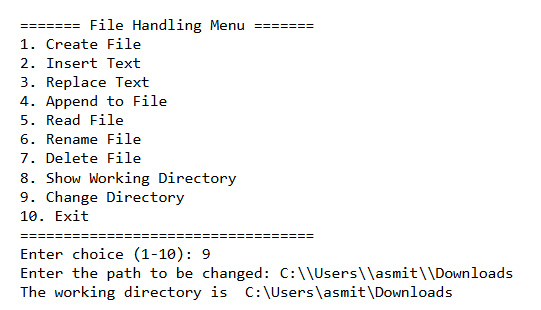
1. **Delete file**

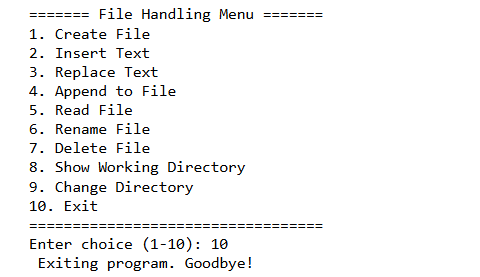
1. **Get current working directory**



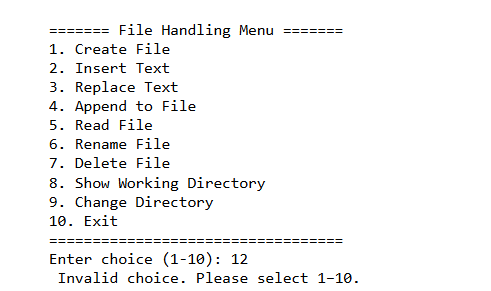
1. **Change the working directory**



1. **Exit**

****

1. **if provide invalid input**



**Conclusion**

This Python program serves as an efficient and user-friendly file handling system that allows users to interact with files and directories through a structured, menu-driven interface. It provides a practical demonstration of how core file operations can be managed using simple programming constructs and the built-in os module in Python.

The program supports a variety of essential file operations, including creating files, reading contents, inserting and appending text, replacing existing content, renaming files, and deleting files. It also includes functionality for displaying and changing the current working directory, giving users full control over their file management tasks from within the program itself.

Each feature is logically organized into separate functions, which promotes readability, reusability, and ease of maintenance. The use of conditional statements to navigate through the menu makes the interface intuitive and interactive, catering well to users who may not be familiar with command-line file operations.

The insert\_text function effectively demonstrates how content can be dynamically modified by inserting new text at a user-defined position. Similarly, the replace\_text function highlights how file content can be edited in-place, showcasing the flexibility and power of Python’s string handling capabilities.

Overall, this program demonstrates a strong grasp of Python’s file handling and OS interaction capabilities. It efficiently accomplishes its goal of providing a basic but comprehensive file management system, making it a valuable tool for educational purposes and for users who wish to automate routine file tasks in a controlled environment. The clear structure and practical features make this program a robust example of how Python can be used to build real-world utilities with straightforward logic and design.

**Bibliography**

* **W3Schools.** Python File Handling.  
  <https://www.w3schools.com/python/python_file_handling.asp>
* **GeeksforGeeks.** File Handling in Python.  
  <https://www.geeksforgeeks.org/file-handling-python/>
* **Programiz.** Python File Handling.  
  <https://www.programiz.com/python-programming/file-operation>
* **TutorialsPoint.** Python - File Handling.  
  <https://www.tutorialspoint.com/python/python_files_io.htm>
* **Python Software Foundation.** Reading and Writing Files in Python.  
  <https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files>

**GitHub link**

<https://github.com/AsmithaAsmitha>