PDF Protection Tool Using Python

By Inlighn Tech

Objective:

The objective of this project is to create a Python-based tool that allows users to add password protection to PDF files. This project will help students understand file handling, encryption, and command-line arguments in Python.

Project Overview:

PDF files often contain sensitive information, and protecting them with a password adds an extra layer of security. This project focuses on building a tool that encrypts PDF files using Python's PyPDF2 library. The script takes an input PDF file, applies password protection, and saves the encrypted version as a new file.

How the Project Works:

1. **Input Handling:** The script accepts three command-line arguments: the input PDF file, the output (protected) PDF file, and the password.
2. **Reading the PDF:** The script opens the input PDF in read mode.
3. **Creating a New PDF:** A new PDF file is created, and each page from the original PDF is added to it.
4. **Applying Encryption:** The `encrypt()` function is used to apply password protection to the new PDF file.
5. **Saving the Encrypted File:** The password-protected PDF is saved with the specified output file name.
6. **Error Handling:** The script handles cases where the input file is missing, invalid, or unreadable.

Key Concepts Covered:

- File handling in Python
- Working with PDFs using PyPDF2
- Implementing encryption for security

- Using command-line arguments in Python scripts
- Exception handling for robust code execution

Step-by-Step Implementation:

1. Install the PyPDF2 library if not already installed.
2. Create a Python script that accepts command-line inputs.
3. Open and read the input PDF file.
4. Copy the content of the original PDF to a new `PdfWriter` object.
5. Apply encryption using the `encrypt()` method.
6. Save the encrypted file and provide user feedback.
7. Implement error handling to manage missing or corrupt PDF files.

Expected Outcomes:

By completing this project, students will:

- Learn how to work with PDFs programmatically.
- Understand encryption techniques for securing documents.
- Gain experience with command-line argument handling in Python.
- Develop a useful tool for securing confidential PDF files.

Next Steps:

Students should implement their own version of the PDF protection tool using the outlined concepts. A video lecture will be provided later to demonstrate the correct implementation and solution. This project builds foundational skills for document security and automation tasks in Python.