**Project:** PDF\_CRACKER  
**Student / Intern:** Meet Pandit

**Internship:** [Inlighn Tech]

**Objective:**

The objective of this project is to develop a Python-based tool that attempts to crack hashed passwords using either a wordlist or brute-force techniques. This project will help students understand cryptographic hash functions, password security vulnerabilities, and multi-threading in Python.

**Project Overview:**

Passwords stored as hashes are commonly used in authentication systems. This project demonstrates how attackers attempt to break hashed passwords using dictionary attacks and brute-force methods. The script allows users to input a hash, specify a hash algorithm (e.g., MD5, SHA-256), and choose between using a wordlist or generating passwords dynamically

**How the Project Works:**

**1. Input Handling:** The user provides a hashed password, hash type, and optional parameters such as a wordlist or password length range.

**2. Dictionary Attack:** If a wordlist is provided, the script reads it and checks if any word matches the target hash.

**3. Brute Force Attack:** If no wordlist is used, the script generates password combinations using letters, digits, or custom character sets.

**4. Hash Matching:** The script hashes each password attempt and compares it with the target hash.

**5. Multi-threading:** The script utilizes multiple threads to speed up the cracking process.

**6. Output Result:** If a match is found, the cracked password is displayed; otherwise, the script reports failure.

**Key Concepts Covered:**

● Cryptographic hash functions (MD5, SHA-1, SHA-256, etc.)

● Dictionary attacks and brute-force techniques

● Multi-threading for performance optimization

● Handling command-line arguments in Python

● Using external libraries like hashlib and itertools

**Step-by-Step Implementation:**

1. Install required Python libraries if not already installed.

2. Create a Python script that accepts command-line inputs for hash, hash type, and optional parameters.

3. Implement a function to check if a generated password matches the target hash.

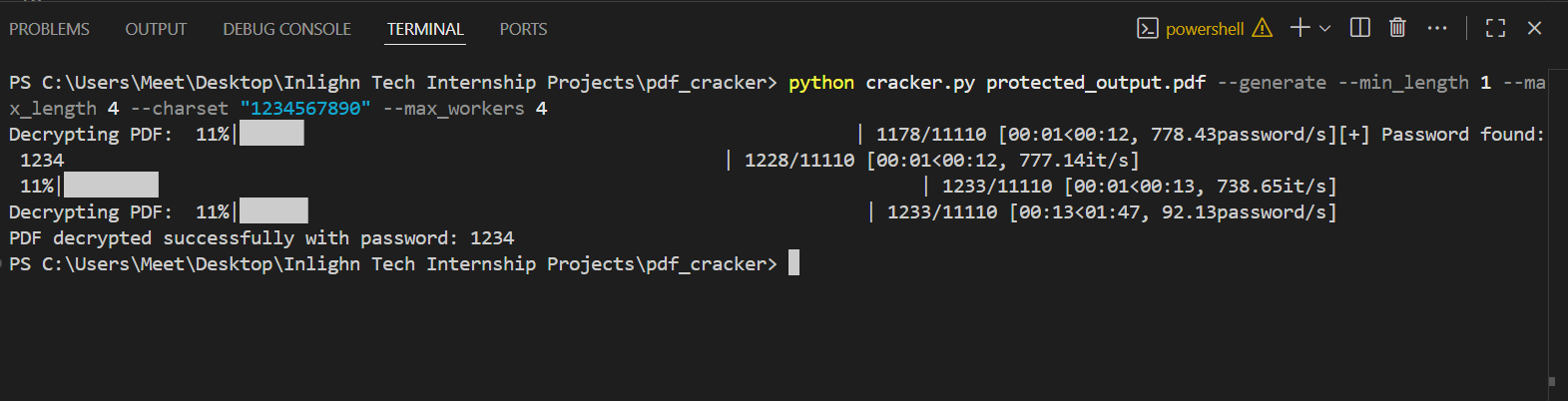
4. Implement the dictionary attack by reading words from a given wordlist.

5. Implement the brute-force attack to generate passwords within a specified length range.

6. Utilize multi-threading to accelerate password attempts.

7. Display the cracked password if found, or a failure message if not.

**Expected Output**

****