Week 1 Lab: Extracting Email Addresses Using Python and Regex

Objective:

The goal of this lab is to give students hands-on experience with Python and regular expressions (regex) in the context of digital forensics. Specifically, students will write a Python script to extract all valid email addresses from a given text file using regex.

Lab Instructions:

1. **Introduction**:
    - In digital forensics, extracting contact information such as email addresses is often a critical task during investigations. Email addresses have a well-defined structure, making them an ideal target for regular expression searches.
    - In this lab, you will develop a Python script that uses regex to search and extract valid email addresses from the provided text file.
2. **Required Libraries**:
    - **Python** comes with the re module, which supports regular expressions.
    - You don't need to install any additional packages.
3. **What You'll Do**:
    - Write a Python script that reads the provided **lab1example.txt** file.
    - Develop a regex pattern to identify valid email addresses in the text.
    - Extract and print the email addresses found in the file.

Step-by-Step Guide:

1. Open the Text File:

- Use Python's built-in file handling to open and read the contents of **lab1example.txt**.

Example:

```
# Open and read the content of lab1example.txt
with open('lab1example.txt', 'r', encoding='utf-8') as file:
    text = file.read()
```

2. **Create a Regex Pattern for Email Addresses**:

Use the following regex pattern to match valid email addresses:

```
r'[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}'
```

- This pattern matches common email address formats like john.doe@example.com.

Explanation of the Pattern:

- **[a-zA-Z0-9._%+-]+**: Matches the username part of the email (letters, numbers, dots, underscores, and some special characters).
- **@**: Matches the literal "@" symbol.
- **[a-zA-Z0-9.-]+**: Matches the domain name part (letters, numbers, dots, hyphens).
- **\.[a-zA-Z]{2,}**: Matches the dot and the top-level domain (e.g., .com, .org).

3. Search for Email Addresses Using re.findall():

- Use Python's re.findall() method to find all occurrences of the email pattern in the text.

4. Save the Results:

- Optionally, modify your script to save the extracted email addresses to a new file (e.g., extracted_emails.txt).

Expected Output:

When your script is run, it should extract the following email addresses from the **lab1example.txt** file:

1. john.doe@example.com
2. jane.doe@example.net
3. sarah.connor@fake.org
4. emily.blunt@random.com

Extended Task (Optional):

- Modify your regex pattern to ignore any malformed email addresses (e.g., those without a valid top-level domain).