LAB 05

Modules, and File Handling

Task 01

1. **Package and Module Structure:**
    - Create a package named **calculator**.
    - Inside the **calculator** package, create multiple modules (Python files) for different types of calculations, such as **basic.py**, **scientific.py**, and **converter.py**.
2. **Basic Calculator (basic.py):**
    - Inside the **basic.py** module, create functions for basic arithmetic operations (addition, subtraction, multiplication, division).
    - Each function should take two numbers as input and return the result.
3. **Scientific Calculator (scientific.py):**
    - Inside the **scientific.py** module, create functions for scientific calculations, such as square root, exponentiation, trigonometric functions, etc.
    - Implement at least three different scientific functions. Choose any three.
4. **Converter (converter.py):**
    - Inside the **converter.py** module, create functions to convert between different units or measurement systems. For example, you can create functions for length conversion, temperature conversion, currency conversion, etc. Choose any three.
5. **Main Calculator (main.py):**
    - Create a separate Python script named **main.py** outside the package.
    - In **main.py**, import the functions from the **calculator** package modules.
    - Implement a menu-driven interface where the user can select the type of calculation (basic, scientific, converter).
    - Depending on the user's choice, call the respective function(s) from the package modules and display the result.

Task 02

1. **File Input:**
    - Ask the user to enter the path to a text file they want to parse. Ensure the file exists and handle any potential exceptions.
2. **File Reading:**
    - Read the content of the text file.
3. **Parsing Options:**
    - Provide the user with several parsing options, such as:

- Count the number of lines in the file.
- Count the number of words in the file.
- Extract and display all email addresses from the file.
- Extract and display all URLs from the file.
- Search for specific keywords or phrases in the file and display their occurrences.

4. **Parsing Functions:**
    - Implement separate functions for each parsing option, such as **count_lines**, **count_words**, **extract_emails**, **extract_urls**, and **search_keywords**.
    - These functions should take the file's content as input and return the respective results.
5. **User Interaction:**
    - Display a menu for the user to select which parsing option they want to perform.
    - Depending on the user's choice, call the corresponding parsing function and display the results.
6. **Output:**
    - For each parsing option, display the results to the user.

Sample.txt:

Sample Text File for Parsing

This is a sample text file that you can use for testing the file parsing task in Python.

Contact Information:

- Email: user@example.com

- Phone: +1 (123) 456-7890

- Website: https://www.example.com

Keywords: Python, parsing, file handling, text, sample

Here's another email: support@example.com.

URLs:

- https://www.python.org

- https://www.github.com

- https://www.stackoverflow.com