# **README:**

# **Script Analyzer**

# ## Overview

The Script Analyzer is a Python script designed to analyze C/C++ scripts for various coding standards and practices. It performs checks related to indentation, naming conventions, modularization, file encoding, consistency, code reuse, and memory leaks. Additionally, it can email a log file with analysis results to a specified recipient.

# ## How to Use

1. \*\*Environment Setup:\*\* Make sure you have Python installed on your system.

2. \*\*Clone the Repository:\*\* Clone the repository containing the Script Analyzer code.

3. \*\*Install Dependencies:\*\* Install the required Python dependencies using `pip install -r requirements.txt`.

4. \*\*Run the Script:\*\* Execute the script by providing the full path to the C/C++ script you want to analyze when prompted.

# ## Functions

1. \*\*`ScriptAnalyzer` Class:\*\*

- `\_\_init\_\_(self, script\_path)`: Initializes the Script Analyzer with the path to the script.

- `run\_analysis(self)`: Runs the analysis on the script, performing various checks and emailing the analysis log.

2. \*\*Analysis Checks:\*\*

- `check\_total\_lines(self)`: Checks if the total number of lines in the script exceeds a recommended maximum.

- `check\_indentation(self)`: Checks for indentation issues in the script.

- `check\_naming\_conventions(self)`: Checks for naming convention violations in variables, functions, and types.

- `check\_modularization(self)`: Checks for repeated sequences of code that could be refactored into functions.

- `check\_file\_encoding(self)`: Checks the file encoding of the script.

- `check\_consistency(self)`: Checks for consistency issues in line endings and file syntax.

- `check\_code\_reuse(self)`: Checks for possible code duplication through import or require statements.

- `check\_whitespace(self)`: Checks for excessive whitespace or trailing spaces in the script.

- `check\_memory\_leaks(self)`: Checks for memory leaks in C/C++ scripts using Dr.Memory or Valgrind.

3. \*\*Utility Functions:\*\*

- `get\_log\_file\_name(self)`: Generates a unique log file name based on the script's name and current date/time.

- `preprocess\_cpp\_file(self)`: Preprocesses a C/C++ script using the C preprocessor (`cpp`).

4. \*\*Email Functionality:\*\*

- `send\_email(sender\_email, sender\_password, recipient\_email, attachment\_path)`: Sends an email with an attached log file.

# ## Logic and Practicality

- The Script Analyzer employs a combination of regular expressions, AST parsing, and file I/O to perform various checks on C/C++ scripts.

- It utilizes logging to capture warnings and errors during analysis, providing a detailed log file for review.

- The modular design allows for easy extension with additional checks or functionality.

- The script's email functionality enables automatic sharing of analysis results with stakeholders.

# ## Additional Notes

- Ensure that the SMTP server configuration (SMTP\_SERVER, SMTP\_PORT, SENDER\_EMAIL, SENDER\_PASSWORD, RECIPIENT\_EMAIL) is correctly set up for emailing log files.

- Some checks, such as memory leak detection, require external tools (Dr.Memory, Valgrind) to be installed and available in the system PATH.