Converting Scraped Code into Static Code (Python & C/C++)

1. Methods Used in Static Code Analysis

• **Lexical Analysis**: Breaks down the source code into tokens such as keywords, identifiers, and symbols.

• **Syntax Analysis (AST):** Builds an Abstract Syntax Tree (AST) to represent the structural organization of the code.

• **Data Flow Analysis:** Tracks how data moves through the program to detect uninitialized or unused variables.

• **Control Flow Analysis:** Analyzes possible execution paths and logic flows to identify unreachable code or infinite loops.

• **Pattern Matching:** Finds predefined patterns or bad coding practices, often used for detecting code smells.

• **Complexity Analysis:** Measures how complex a function or module is using metrics like cyclomatic complexity.

• **Taint Analysis (Security):** Traces untrusted data flow (e.g., user input) to detect potential vulnerabilities like injections.

2. Tools for Python Static Code Conversion

• AST (Abstract Syntax Tree): A built-in Python module that parses code text into a syntax tree structure. It helps extract functions, classes, and control flow from code.

• Pylint: A static code analyzer that detects code smells, errors, and naming violations. It provides detailed quality scores for Python files.

• Flake8: Combines PyFlakes, pycodestyle, and McCabe complexity checker. It helps detect syntax errors, unused variables, and poor formatting.

• Radon: Analyzes code complexity and maintainability index. It calculates cyclomatic complexity and other static metrics.

• Bandit: Performs static security analysis to find potential vulnerabilities in Python code.

3. Tools for C/C++ Static Code Conversion

• Clang-Tidy: A powerful C/C++ static analysis and linting tool built into the LLVM project. It checks for coding style violations, performance issues, and best practices.

• Cppcheck: Performs static analysis on C/C++ code to detect undefined behavior, memory leaks, and possible logic errors.
• Flawfinder: Scans C/C++ source code for potential security weaknesses, especially buffer overflows.

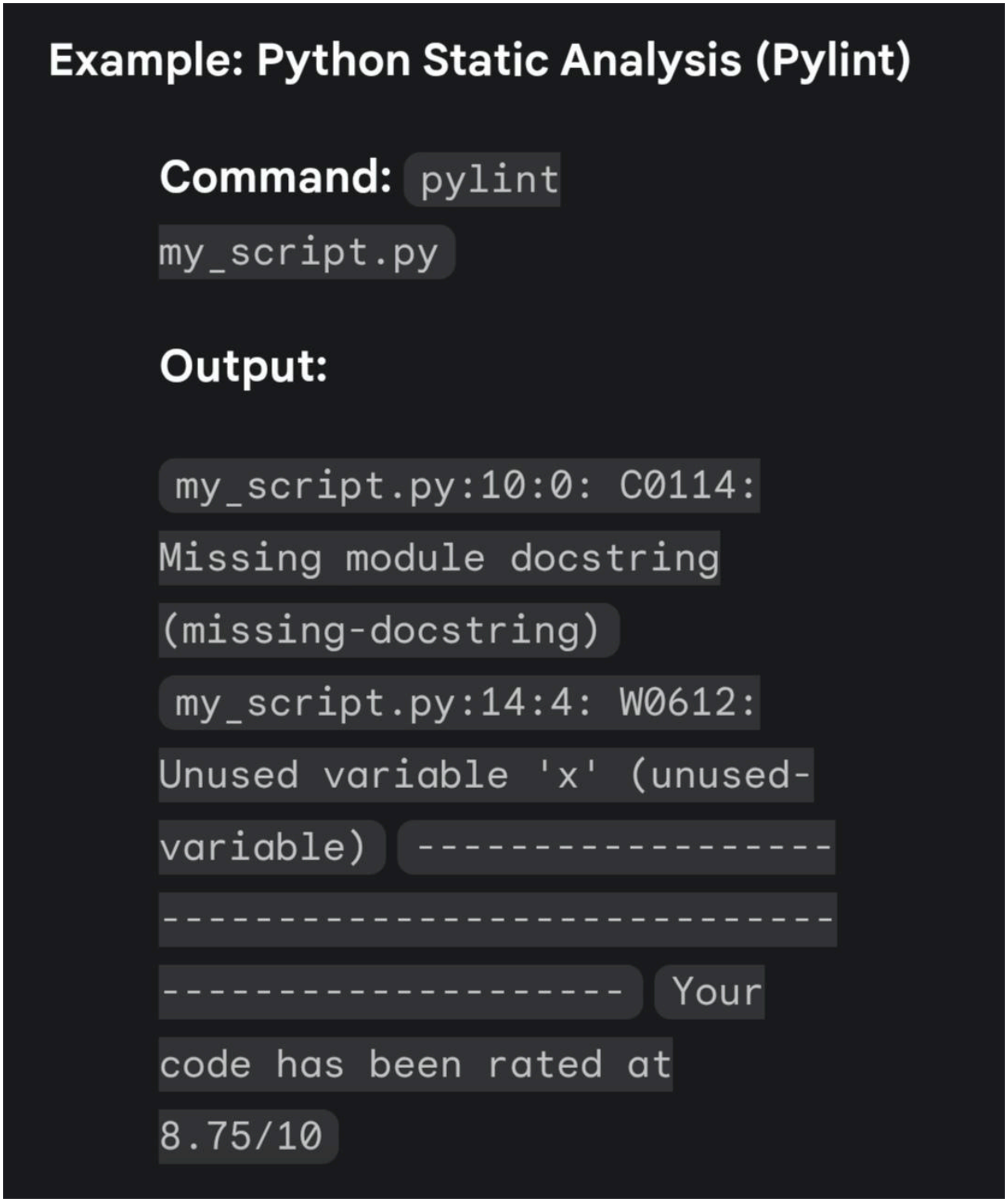
• Infer: A static analysis tool by Meta (Facebook) that detects null pointer exceptions, resource leaks, and concurrency issues in C/C++ programs.

• SonarQube: A multi-language static analysis platform that supports C/C++ as well as Python and others. It provides comprehensive dashboards with metrics on code quality and maintainability.

4.Outputs of Static Code Analysis

Output Type	Description
Error Reports	**Lists of syntax or logical errors found in the source code.**
Warnings	**Non-critical issues or style inconsistencies that may reduce code readability.**
Code Metrics	**Quantitative measures of code quality, such as complexity and maintainability.**
Security Vulnerabilities	**Highlights unsafe functions, injection risks, or buffer overflows.**

Code Smells	**Detects poor coding practices or design issues.**
Quality Scores	**Aggregated ratings based on tool analysis results (e.g., Pylint score: 8.7/10).**



5.Static Analysis Outputs Used as Training Data

Common Features Used as Training Data

Feature Category	Example Features	Source Tools
Complexity Metrics	Cyclomatic complexity, function length, nesting depth	Radon, Cppcheck
Style Metrics	Number of linting errors, code formatting issues, naming violations	Flake8, Pylint
Maintainability Metrics	Maintainability index, duplication %, readability	SonarQube, Radon
Security Metrics	Number of detected vulnerabilities, unsafe functions	Bandit, Flawfinder
Documentation Metrics	Presence of comments and docstrings	Pylint, SonarQube

Example: Training Data Table

File	Cyclomatic Complexity	Maintainability Index	Bug Count	Security Issues	Quality Score	Label (Target)

`file1.py`	3	85	0	0	9.5	**Good**
`file2.py`	14	40	3	1	6.8	**Poor**
`file3.cpp`	8	65	1	0	7.9	**Average**