Bug Detection & Fixing AI (AST-Based) - Functional Report

Objective:

To build a basic AI system that detects predefined bug patterns in Python code using AST (Abstract Syntax Tree), suggests fixes, and outputs a report file describing each issue.

1. What It Does:

- Reads Python code
- Parses the code into its structural tree form using AST
- Matches known buggy patterns using predefined logic
- Suggests corrections
- Generates a report with line numbers, issue descriptions, and suggested fixes

2. How It Works:

a. Parsing with AST:

- Python's ast module is used to convert raw code into a tree structure.
- This tree helps analyze code at a structural level-like checking if a condition has an invalid assignment or if a function call is malformed.

b. Bug Pattern Matching:

Predefined buggy patterns are stored (as logic or data). Examples:

- Assignment in condition (if x = 5: instead of ==)
- Print without parentheses (print "Hello" instead of print("Hello"))
- Missing colon in for, if, while statements

The model checks each AST node to match against these patterns.

c. Bug Fixing Logic:

When a match is found, the model suggests the most likely fix:

- Replace = with ==
- Add parentheses to print
- Add missing colon

d. Report Generation:

For every bug found, the system logs:

Line X: Detected issue - [description]

Suggested Fix: [fix]

All entries are compiled into a file called report.txt.

3. Why This Approach is Used:

- AST ensures structural understanding - more accurate than simple string matching.
- Safe and Pythonic - uses native modules (ast, astor).
- Scalable - new bug patterns can be added easily.
- Foundation for ML Integration - this logic can later evolve into a learning-based model.

Example Entry in report.txt:

```
Line 4: Assignment used in conditional expression

Suggested Fix: Use '==' instead of '=' in if condition


Line 7: Function call to 'print' missing parentheses

Suggested Fix: Change to 'print(...)'
```