Lab 5: Static Code Analysis

NAME : Unnathi Baskar
SRN : PES2UG23CS663
SECTION : K

Known Issue Table:

Issue	Type	Line(s)	Description	Fix Approach
Mutable default arg	Bug	12	logs=[] shared across calls	Change default to None and initialize in method
Bare except	Bug	19	The except: block catches all errors but does not handle or logging them.	fixed it using specific exceptions: except KeyError, except (TypeError, ValueError) and on exception print error messages.
Use of eval()	Security	59	The eval() function can execute arbitrary code, making it unsafe.	Replace with json.loads()
Unused import	Style	2	The logging module is imported but never used in the code.	Remove the unused import statement.
Naming style issues	Style	8, 14, 22, 25, 31, 36, 41	Function names like addItem and removeItem don't follow Python's snake_case naming rule.	Rename functions using lowercase with underscores (e.g., add_item).
Missing documentation	Maintainability	1, 8, 14, 22, 25, 31, 36, 41, 48	Several functions and the module itself are missing docstrings, reducing readability.	Add descriptive docstrings
Improper file handling	Code smell	26, 32	Files are opened with open() and not closed.	Use with open(...) as f: statement to handle files automatically.

REFLECTION ANSWERS :

1. Which issues were the easiest to fix, and which were the hardest? Why?

- Easiest fixes: Formatting and naming convention issues (like missing docstrings and non-snake_case function names) were simple to correct since they mostly required style changes.
- Hardest fixes: Handling exceptions properly and removing the use of eval() were the hardest. These required understanding how the code worked, adding error-specific *except* blocks, and replacing unsafe code with safer alternatives like *json.loads()*.

2. Did the static analysis tools report any false positives?

- Yes, there was a minor false positive where Bandit flagged a low-severity issue for a simple try-except block that intentionally handled missing items in the inventory. Even though the tool warned against a bare except, it was already controlled and safe error-handling case.

3. How would you integrate static analysis tools into your actual software development workflow?

- Integrate pylint, flake8, and bandit into the CI/CD pipeline so every commit and pull request is automatically scanned.
- Use pre-commit hooks locally to catch style and security issues before pushing code.
- Generate reports automatically and review them as part of the development process to maintain consistent code quality.

4. What tangible improvements did you observe after applying the fixes?

- The code became more readable and maintainable, with clear function documentation and consistent formatting.
- Security improved by removing unsafe eval() and proper file handling.
- Error handling became more robust, reducing the chance of runtime crashes.