Name : Keshav Vithal Sangonda

SRN : PES1UG23CS300

SEC : “E”

| **Issue** | **Type** | **Line(s)** | **Description** | **Fix Approach** |
| --- | --- | --- | --- | --- |
| Mutable default arg | Bug | 12 | logs=[]shared across calls | Change default toNoneand initialize in method |
| Missing module docstring | Style | 1 | No description at the top of the file | Added a module docstring at line 1 |
| Function name not snake\_case | Style | 9, 21, 38, 41, 46, 51, 56, 61, 68 | Used camelCase for functions likeaddItem | Renamed all functions to snake\_case style (e.g.add\_item) |
| Missing function docstring | Style | All | Functions have no explanatory docstrings | Added docstrings to every function |
| Unused import | Code Smell | 3 | import loggingnot used | Removed the unused import |
| Unspecified encoding | Code Smell | 43, 47, 52 | open()calls without encoding | Addedencoding='utf-8'to all file operations |
| Duplicate function (saveData) | Bug | 46, 51 | Function redefined twice | Removed duplicate and kept single correct version |
| Unsafe use of eval | Security | (main) | Usedeval("print('eval used')") | Removedevaland used a direct print statement |
| Final newline missing | Style | 81 | No blank line at file end | Added a final newline at the end of the file |
| String formatting style | Style | 19 | Used%operator for formatting | Changed to f-string formatting |

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

Easiest issues:

Adding docstrings, renaming functions to snake\_case, and removing unused imports were the easiest. These changes were simple and only required changing a few lines without affecting how the code runs.

Hardest issues:

Fixing the mutable default argument (logs=[]) and removing all unsafe uses of eval were harder. They required understanding why these patterns are problems and adjusting both the code and how functions are called. Changing file handling to add encoding also took some time because it meant updating every place files are opened.

## Did the static analysis tools report any false positives? If so, describe one example.

Yes, sometimes static analysis tools show warnings that are not real problems in practice ("false positives"). For example, if a variable is imported but only used when a certain option is given by the user, the tool might say "unused import" even though it is needed in some cases

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

## workflow? Consider continuous integration (CI) or local development practices.

Static analysis tools should be used in two ways:

* **During local development:** Run tools like Pylint, Bandit, and Flake8 regularly while writing code to catch mistakes early.
* **In Continuous Integration (CI):** Set up these tools on every code push in CI services (like GitHub Actions or GitLab CI), so any errors or warnings must be fixed before merging code. This keeps code clean for all team members.

## 4. What tangible improvements did you observe in the code quality, readability, or potential

## robustness after applying the fixes?

* **Code quality:** Fewer bugs and clear logic after removing bad patterns like mutable default args and eval.
* **Readability:** Functions now have docstrings, proper names, and format, which makes code easier to understand for others or yourself later.
* **Robustness:** Error handling is improved and resources (like files) are managed more safely, reducing the chance of bugs when the code is used on different computers.