|  |  |
| --- | --- |
| Date |  |
| Developer (Person whose code is being reviewed) |  |
| Developer’s lab version (A, B, or C) and lab partner group name |  |
| Reviewer |  |

Instructions

The reviewer will complete this form for the beta version of a lab assignment done by one of their lab partners. After filling out the “Beta” column and adding comments, the reviewer will upload this document to the Lab Beta + Review Forum.

The developer will revise the beta version of their lab work and fill out the “Production” column to reflect any changes they have made. The developer will submit this completed form along with the production version of their lab assignment.

Note: Links to **Code Review FAQs**, **Coding Style** and **Best Practices** are at the end of this document.

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Beta** | **Release** |
| Is the developer’s name and the date in comments in each file? |  |  |
| Is the code free from errors as shown in the IDE? (If no, list errors) |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Is i/o from inference code separated (in separate functions or modules)? |  |  |
| Does it run without errors? (if no, list them) |  |  |
|  | | |
| Does it have all required functionality? (List anything that’s missing) |  |  |
|  | | |
|  | | |
| Are there tests to check all input combinations? (list anything that’s missing) |  |  |
|  | | |
| Does the style conform to Python coding conventions? (list what doesn’t) |  |  |
|  | | |
|  | | |
| Do design and implementation conform to best practices? (list what doesn’t) |  |  |
|  | | |
|  | | |
| Comments | | |
|  | | |

**Appendix**

[Code Review Procedure and FAQs](file:///Volumes/DataCard/Repos/CS210-Repos/CS210-CourseMaterials/Labs/Lab02-RuleBasedSystems/CodeReviewProcedure.md%20CodeReviewProcedure.html)

Aspects of coding style to check

* Is proper indentation used?
* Are variables and function names descriptive and meaningful?
* Have unnecessary lines of code, commented-out code, and unused files been removed?
* Are there clear and concise comments or docstrings explaining complex code or functions?
* Do variable, function, and method names use snake\_case?
* Are class names written using PascalCase (aka TitleCase)?
* Are constant names written using ALL\_CAPS (typically defined at the module level)?
* Are import statements organized (standard library first, then third-party, then local imports)?

Best practices to check

* Is the code DRY (Don’t Repeat Yourself) — no duplicated logic or copy-pasted code?
* Are named constants or configuration variables used instead of hard-coded literal values?
* s business logic separated from input/output code (e.g., computation in one module, CLI handling in another)?
* Are instance variables intended for internal use prefixed with an underscore (e.g., \_value)?
* Are local variables used inside methods whenever possible, instead of storing data in instance attributes unnecessarily?
* Does each function or method do one clear task and have a single responsibility (no “Swiss Armey” methods)?
* Are classes cohesive (each has a clear, well-defined purpose) and loosely coupled (minimal dependencies on other classes)
* Is inheritance used appropriately, or replaced with composition where simpler?
* Are data classes (@dataclass) used where appropriate for simple data containers?