NEOBRIM
Elevating Homes

Coding Guidelines

Languages Accepted: Python, R, Julia, Java, C++

1. **Variable and Function Naming:**
    - Use descriptive and meaningful names for variables and functions.
    - Follow the **snake_case** convention for variable and function names.
2. **Comments and Docstrings:**
    - Include comments to explain complex sections of code or provide context.
    - Use docstrings for classes and functions to provide comprehensive documentation.
    - Clearly specify parameters, return values, and functionality in docstrings.
3. **Consistent Indentation and Whitespace:**
    - Maintain consistent indentation (e.g., one tab per level).
    - Ensure proper alignment of code blocks.
    - Use whitespace judiciously for improved readability.
4. **Imports:**
    - Group imports and separate them into standard library imports, third-party library imports, and local imports.
    - Alphabetically order import statements within each group.
5. **Modularity:**
    - Divide the script into logical sections, such as module imports, class definitions, function definitions, and the main execution block.
6. **Consistent Data Types:**
    - Ensure consistent usage of data types across the script.
    - Be mindful of converting data types when necessary.
7. **Exception Handling:**
    - Implement exception handling where appropriate, especially when dealing with external resources like files.
    - Provide clear error messages and logging.
8. **Global Constants:**
    - Define global constants in uppercase with underscores separating words.
9. **Print Statements:**

- Include informative print statements for debugging or demonstration purposes.

10. Unused Imports:

- Remove any unused or redundant import statements to keep the code clean.

11. Error Handling:

- Implement error handling where appropriate, providing clear error messages and logging.

12. Class Initialization:

- Clearly define and document the purpose of class attributes during initialization.

13. Function Length:

- Keep functions concise and focused on a specific task. If a function becomes too long, consider refactoring.

14. File Structure:

- Organize code in a clear and logical file structure, with separate sections for classes, functions, and the main execution block.

NEOBRIM
Elevating Homes

GitHub Collaboration Guidelines for AI Code Development

1. **Repository Forking:**
    - Fork the main repository to your GitHub account.
    - Clone your forked repository to your local machine.
2. **Branching Strategy:**
    - Create a new branch for each significant contribution or experiment.
    - Name branches descriptively and include your GitHub username (e.g., **username/feature/new-feature**, **username/experiment/ai-model**).
    - Each contributor should work in their dedicated branch.
3. **Commit Messages:**
    - Write clear and informative commit messages.
    - Start messages with a concise verb in the present tense (e.g., "Implement neural network," "Optimize hyperparameters").
4. **Code Review:**
    - Request code reviews from team members or supervisors.
    - Use GitHub's pull request review features for collaborative feedback.
5. **Pull Requests:**
    - Open a pull request (PR) for each branch when ready for review.
    - Include a brief description of the AI model or changes in the PR.
    - Reference relevant issues or tasks.
6. **PR Merging:**
    - Do not merge your own PR; wait for at least one team member to review and approve.
    - Resolve conflicts before merging if they arise.
    - Ensure the AI model meets performance expectations and goals.
7. **Code Style and Standards:**
    - Follow established coding standards *(mentioned above)* and adhere to best practices for AI model development.
    - Utilize linters and code formatters to maintain consistent formatting.
8. **Documentation:**

- Document the purpose, architecture, and parameters of your AI model.
- Include comments explaining complex sections of the code.

9. **Issue Tracking:**

- Reference relevant issues in commit messages and pull requests.
- Utilize GitHub issues for discussing AI model improvements, bug reports, or feature requests.

10. **Testing:**

- The code will only be approved to be merged into the main branch after adequate testing by the supervisor.

11. **Communication:**

- Discuss AI model design decisions or challenges through GitHub discussions or Slack.
- Communicate progress updates and potential issues with team members on Email or Slack.

12. **Security Considerations:**

- Be vigilant about data privacy and model security.
- Report any security concerns privately to the project maintainers.

13. **Learning Opportunities:**

- Embrace the collaborative learning environment; share insights and knowledge with team members.
- Seek feedback on AI model strategies and techniques.

Given the collaborative nature of AI code development, contributors are encouraged to have their own branches for independent experimentation and contributions. This branching strategy allows each contributor to work on their AI models without interfering with others' work. If there are dependencies or collaborations, consider merging branches collaboratively or coordinating efforts through discussions.

NEOBRIM
Elevating Homes

Documentation Guidelines

Effective documentation is essential for project understanding and collaboration. Follow these guidelines for clear and concise documentation in Markdown, LaTeX, Word, PDF, Notepad (.txt), or handwritten formats.

1. **File Naming:**
    - Choose a descriptive and consistent filename.
    - Include relevant information like version numbers.
2. **Structure:**
    - Organize documentation with clear sections: Introduction, Installation, Usage, Examples, and Conclusion.
    - Use headings and subheadings for hierarchy.
3. **Formatting:**
    - Follow syntax rules for your chosen format.
    - Maintain consistent styles in Word, PDF, or handwritten documents.
4. **Language:**
    - Write in clear and concise language.
    - Balance technical details with user-friendly explanations.
5. **Code Examples:**
    - Include well-commented code snippets.
    - Use syntax highlighting for clarity.
6. **Visuals:**
    - Integrate diagrams or visuals for better understanding.
    - Clearly label and explain components.
7. **Updates and Versioning:**
    - Include version history and updates if applicable.
    - Clearly state the current version.