1. Use Descriptive Names: Choose variable and method names that clearly describe their purpose or what they represent, making the code self-documenting.

2. Keep Functions Small and Focused: Each function should do one thing only, improving readability, reusability, and ease of debugging.

3. Write Self-Documenting Code: Write your code in such a way that its purpose and operation are clear to another developer without needing extensive comments.

4. Implement Error Handling Gracefully: Use try-except blocks to manage errors smoothly, ensuring the program’s stability and providing meaningful error messages.

5. Adhere to the DRY Principle: "Don’t Repeat Yourself" - avoid code duplication by using functions, loops, or other constructs to consolidate repeated logic.

6. Format Code Consistently: Use consistent indentation, spacing, and other formatting standards to make the code easier to read and maintain.

7. Use Meaningful Comments Wisely: While striving for self-documenting code, use comments to explain "why" something is done, not "what" is done.

8. Optimize Data Structures and Algorithms: Choose the most appropriate data structures and algorithms for your use case to enhance performance.

9. Refactor Regularly: Continuously review and improve the code to simplify complex sections, remove redundancy, and apply best practices.

10. Write Tests: Implement unit tests and other testing methodologies to verify your code works as expected and to facilitate safe refactoring.