**1. Define the Purpose of Comments**

Explain the why, not just the what: Comments should clarify why certain decisions were made, not just what the code is doing.

Avoid restating the obvious: Don't write comments that merely restate the code's functionality; instead, explain complex logic or unusual approaches.

**2. Choose a Comment Style**

Decide the type of comments to use based on your programming language:

Single-line comments: Use for brief, inline explanations.

In C, C++, Java, etc.: // Comment

In Python: # Comment

Block comments: Use for more detailed explanations.

In C, C++, Java: /\* Comment \*/

In Python: """ Comment """

**3. Guideline for Types of Comments**

File Header Comments: Summarize the file’s purpose, the author, date, and version information.

Function/Method Comments: Include a brief description of the function, its inputs, outputs, and any side effects.

Inline Comments: Briefly explain tricky or non-obvious sections of code.

Section Comments: For large blocks or logical sections of code, use section comments to make navigation easier.

**4. Formatting and Style Consistency**

Language: Use clear, concise, and correct grammar.

Punctuation: Decide if comments should be complete sentences (recommended for clarity).

Capitalization: Consistently use capitalization at the beginning of comments.

Alignment: Maintain consistent indentation and alignment for readability.

**5. Avoid Over-Commenting**

Comments should add value. Avoid over-commenting or explaining trivial code like basic loops or variable assignments unless there is a need for clarification.

**6. Update Comments**

Ensure comments are updated when the code changes. Stale or incorrect comments can be misleading.

**7. Special Comment Notations**

TODO: Use to mark code that needs to be completed or revisited later.

*Example:*

# TODO: Handle edge case where window\_size > len(data)

FIXME: Highlight problematic or buggy areas of code.

*Example:*

# FIXME: This doesn't handle negative numbers correctly