**DOCUMENT ON PEP 8**

**INTRODUCTION**:

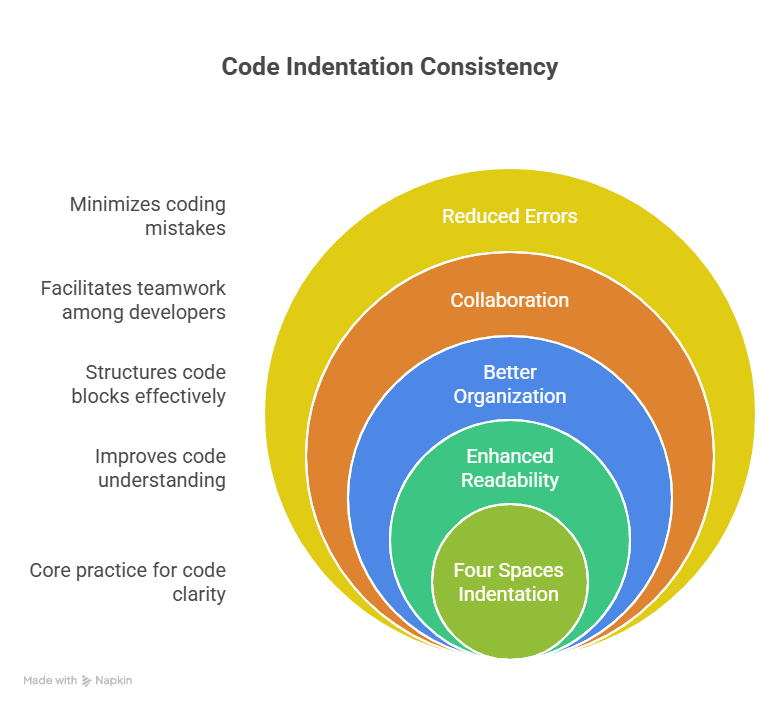
PEP stand for Python Enhancement Proposal. It is a document that standardized new features for Python. PEP8 is the version that sets coding style rule.

It makes python code more readable and standardized. It helps in keeping the project more consistent during the entire project. It helps especially when numerous people or team are working on a single project. It helps in keeping the code uniform during the entire project.

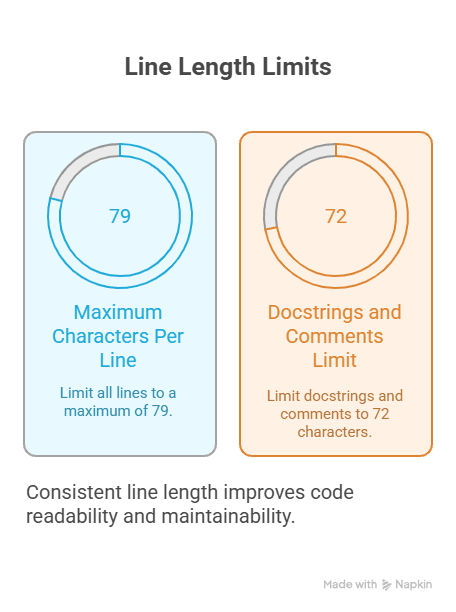
**GUIDELINES OF PEP8: CODE LAYOUT**

There are few guidelines in PEP8 which makes it readable and sets a standardized format.

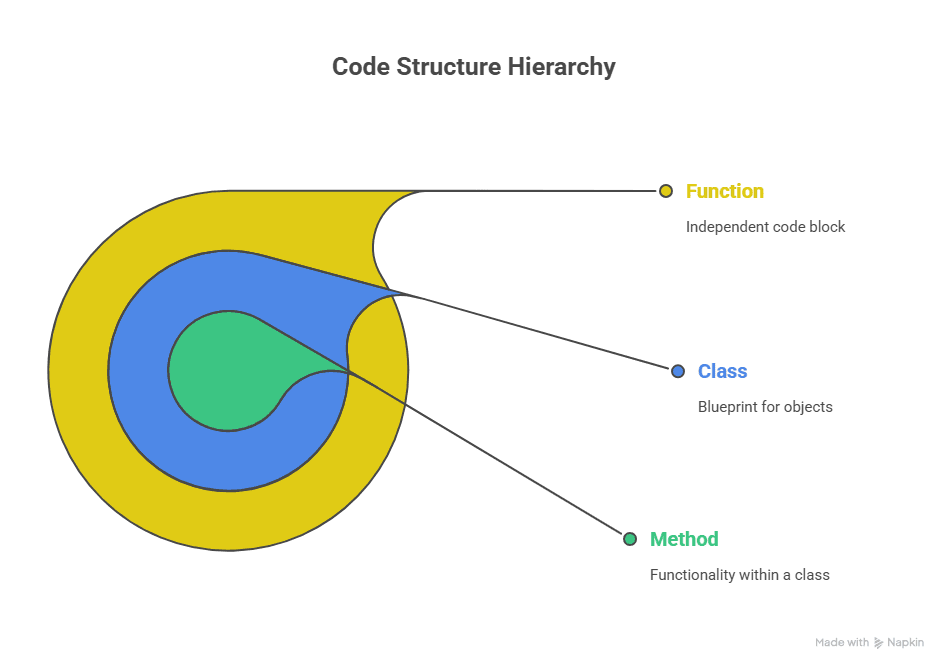
1. INDENTATION : A standard 4 spaces is to be used in each indentation.



1. MAXIMUM LINE LENGTH : Its limit the character of a single line to 79 for clear view.



1. BLANK LINES : Always use blank lines to separate function, class and code. Separate top level function and class with two blank lines and Method definitions inside a class is to be done by one line.

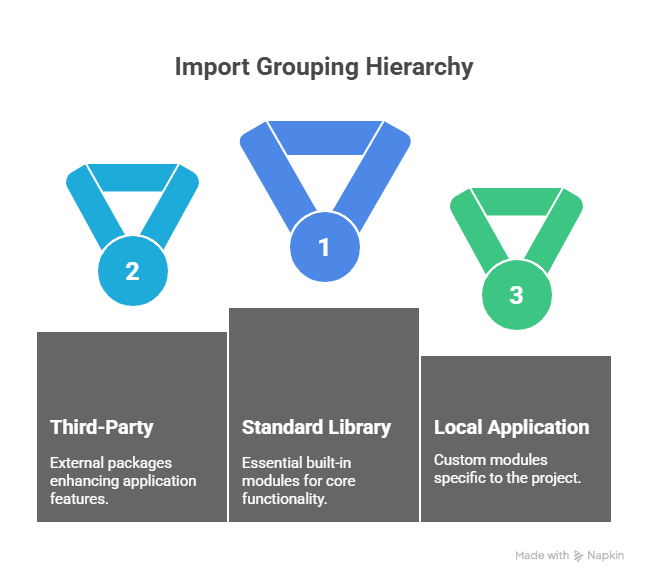


**IMPORTS**

Imports should be grouped in the following order:

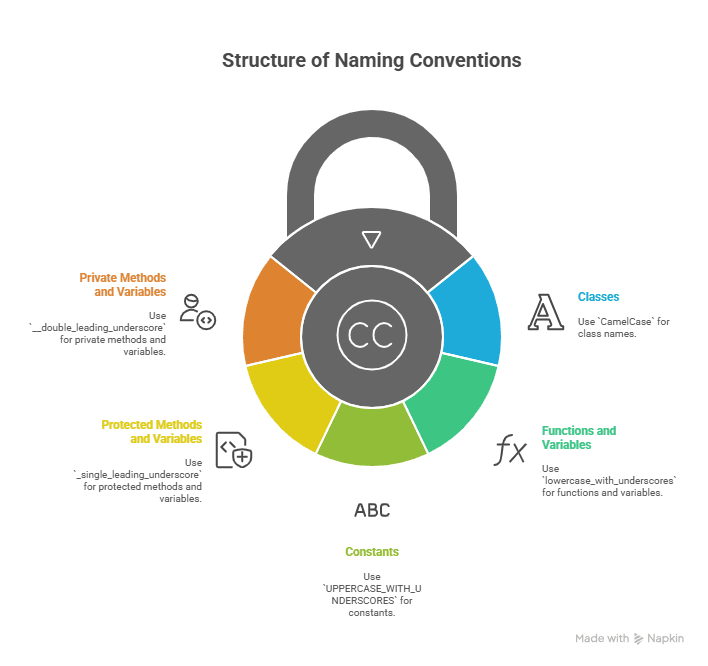
1. Standard library imports
2. Related third-party imports
3. Local application/library specific imports

Separate each group with a blank line. Use absolute imports for clarity.



# Naming Conventions

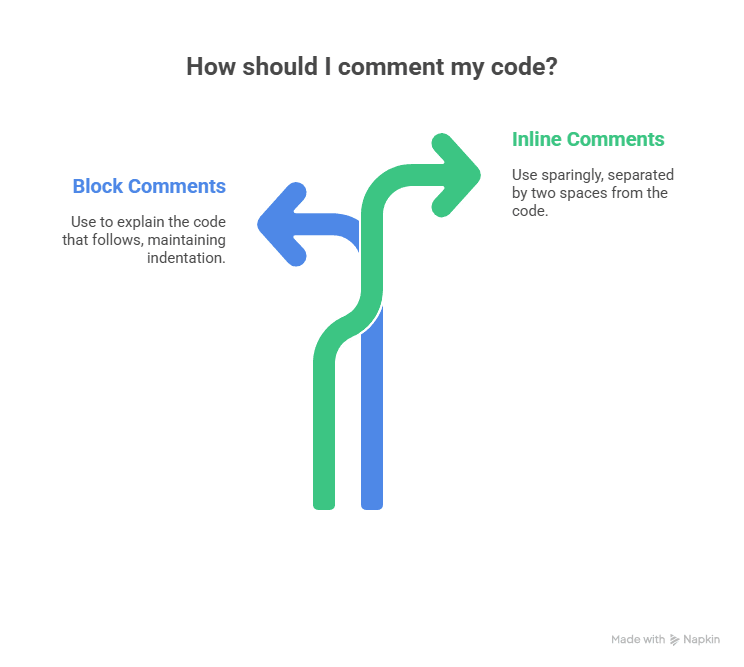
* **Classes:** Use CamelCase.
* **Functions and variables:** Use lowercase\_with\_underscores.
* **Constants:** Use UPPERCASE\_WITH\_UNDERSCORES.
* **Methods and instance variables (protected):** Use \_single\_leading\_underscore.
* **Methods and instance variables (private):** Use \_\_double\_leading\_underscore.



# Comments

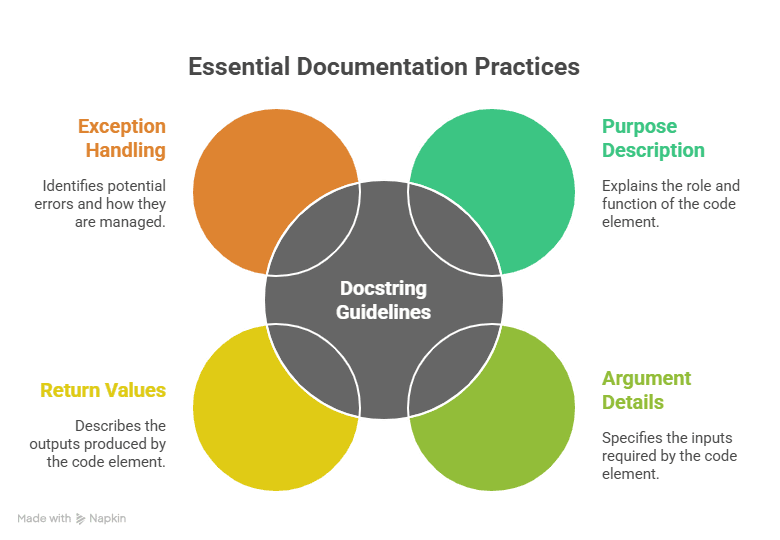
Write clear and concise comments to explain non-obvious code. Comments should be complete sentences and updated when the code changes.

* **Block comments:** Explain the code that follows. Indent to the same level as the code.
* **Inline comments:** Use sparingly and separate from the code by at least two spaces.



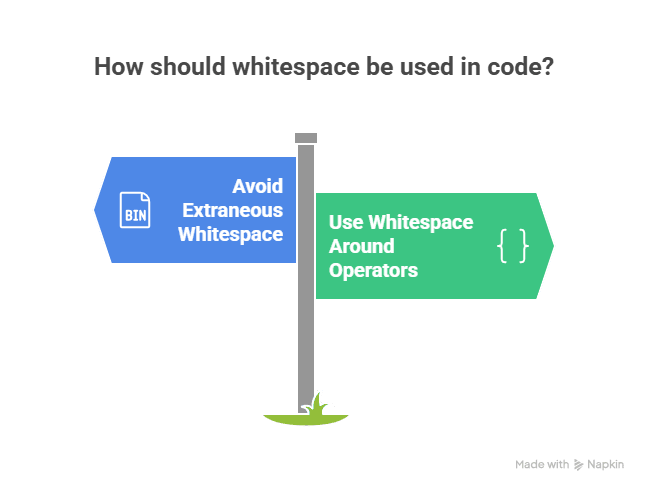
# Docstrings

Write docstrings for all public modules, classes, functions, and methods. Docstrings should describe the purpose, arguments, return values, and any exceptions raised. Use triple double quotes ("""Docstring""").



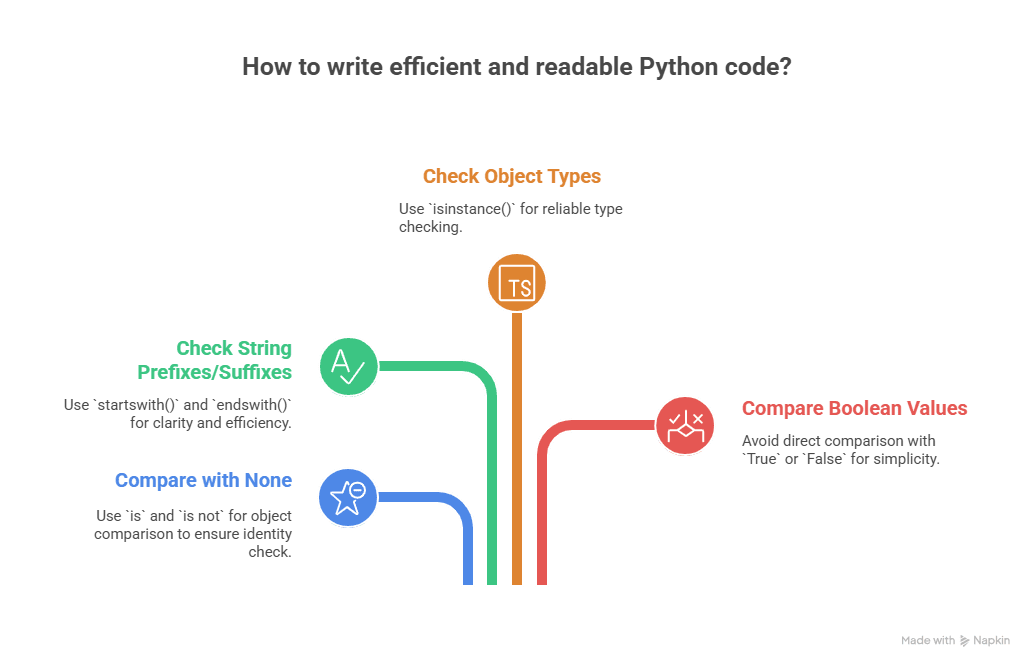
# Whitespace

* Avoid extraneous whitespace:
  + Immediately inside parentheses, brackets, or braces: spam(ham[1], {eggs: 2}) (not spam( ham[ 1 ], { eggs: 2 } ))
  + Before a comma, semicolon, or colon: if x == 4: print x, y; x, y = y, x (not if x == 4 : print x , y ; x , y = y , x)
  + However, a colon *is* like a binary operator, and should have equal amounts on either side: ham[1:9], ham[1:9:3], ham[:9:3], ham[1::3], ham[1:9:]ham[lower:upper] ham[lower:upper:] ham[lower::step]ham[::step] ham[:upper:step]
* Use whitespace around operators: x = 1 + 2 (not x=1+2)



# Programming Recommendations

* Use is and is not to compare objects to None.
* Use startswith() and endswith() instead of string slicing to check for prefixes or suffixes.
* Use isinstance() instead of comparing types directly.
* Don't compare boolean values to True or False using ==.



# Conclusion

Adhering to PEP 8 guidelines significantly improves the quality and maintainability of Python code. By following these recommendations, developers can write code that is easier to read, understand, and collaborate on, ultimately leading to more successful projects. While tools like flake8 and pylint can help enforce PEP 8 compliance, understanding the underlying principles is crucial for writing truly Pythonic code.

# 

