

Clean Code – Overview

“Clean Code” is code which is easy to read and understand

There are certain **key areas, rules and concepts** that help with writing clean code



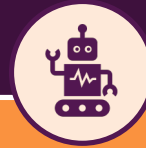
Naming



Comments &
Formatting



Functions



Control
Structures



Classes &
Objects

Naming



Choose **descriptive** names



Nouns for **variables** and **properties** (or **short phrases with adjectives**)
Nouns for **classes**
Verbs for **methods** (or **short phrases with adjectives**)



Be **specific** if possible but **don't be redundant**



Avoid **slang**, **unknown abbreviations** and be **consistent** with your names

Comments & Formatting



Most comments are bad – avoid them!
Feel free to add “good” comments (legal information, warnings, required explanations, todos)



Use vertical formatting (blank lines, line breaks) to keep related concepts together (vertical density) and separate concepts which are not closely related (vertical distance)



Keep lines short (horizontal formatting), add line breaks to improve readability and use indentation



Follow language-specific style guides (e.g. PEP8 for Python) and use IDE auto-formatting to “generate” clean code

Functions



Limit the number of parameters your functions use – look for ways of shrinking the number (e.g. use dictionaries or objects as “value containers”)



Clean functions should be small and “do only one thing”.



Explore the levels of abstraction of your function code to close big gaps between function name and actual code as well as to avoid mixed levels of abstraction in one function.



Write DRY code and avoid unexpected side effects.

Control Structures



Prefer **positive wording**



Avoid deep nesting – for example by using “**Guards**” or by **extracting** control structures into separate functions



Consider using **polymorphism** and **factory functions** to avoid code duplication



Use “**real errors**” instead of “synthetic errors” replicated with if statements

Objects



Use “**real objects**” or **data structures / containers** - depending on what you need



Clean classes should be **small**: Focused on **one responsibility** (which is NOT “one method”!)



Follow the “**Law of Demeter**” when working with **real objects**



Consider following the **SOLID principles**, especially the **SRP** and **OCP**