Software Engineering

### Code quality

* **CLEAN:** readable, simple, and concise. A characteristic of production quality code that is crucial for collaboration and maintainability in software development.
* **MODULAR:** logically broken up into functions and modules. Also an important characteristic of production quality code that makes your code more organized, efficient, and reusable.
* **MODULE:** a file. Modules allow code to be reused by encapsulating them into files that can be imported into other files.
* **REFACTORING:** restructuring your code to improve its internal structure, without changing its external functionality. This gives you a chance to clean and modularize your program after you've got it working.

### Code formatting

# **Writing Clean Code: Meaningful Names**

#### ***Tip: Use meaningful names***

* ***Be descriptive and imply type*** *- E.g. for booleans, you can prefix with is\_ or has\_ to make it clear it is a condition. You can also use part of speech to imply types, like verbs for functions and nouns for variables.*
* ***Be consistent but clearly differentiate*** *- E.g. age\_list and age is easier to differentiate than ages and age.*
* ***Avoid abbreviations and especially single letters*** *- (Exception: counters and common math variables) Choosing when these exceptions can be made can be determined based on the audience for your code. If you work with other data scientists, certain variables may be common knowledge. While if you work with full stack engineers, it might be necessary to provide more descriptive names in these cases as well.*
* ***Long names != descriptive names*** *- You should be descriptive, but only with relevant information. E.g. good functions names describe what they do well without including details about implementation or highly specific uses.*

Try testing how effective your names are by asking a fellow programmer to guess the purpose of a function or variable based on its name, without looking at your code. Coming up with meaningful names often requires effort to get right.

# **Writing Clean Code: Nice Whitespace**

#### ***Tip: Use whitespace properly***

* *Organize your code with consistent indentation - the standard is to use 4 spaces for each indent. You can make this a default in your text editor.*
* *Separate sections with blank lines to keep your code well organized and readable.*
* *Try to limit your lines to around 79 characters, which is the guideline given in the PEP 8 style guide. In many good text editors, there is a setting to display a subtle line that indicates where the 79 character limit is.*

Read PEP 8: https://www.python.org/dev/peps/pep-0008/?#code-lay-out

# **Writing Modular Code**

#### ***Tip: DRY (Don't Repeat Yourself)***

*Don't repeat yourself! Modularization allows you to reuse parts of your code. Generalize and consolidate repeated code in functions or loops.*

#### ***Tip: Abstract out logic to improve readability***

*Abstracting out code into a function not only makes it less repetitive, but also improves readability with descriptive function names. Although your code can become more readable when you abstract out logic into functions, it is possible to over-engineer this and have way too many modules, so use your judgement.*

#### ***Tip: Minimize the number of entities (functions, classes, modules, etc.)***

*There are tradeoffs to having function calls instead of inline logic. If you have broken up your code into an unnecessary amount of functions and modules, you'll have to jump around everywhere if you want to view the implementation details for something that may be too small to be worth it. Creating more modules doesn't necessarily result in effective modularization.*

#### ***Tip: Functions should do one thing***

*Each function you write should be focused on doing one thing. If a function is doing multiple things, it becomes more difficult to generalize and reuse. Generally, if there's an "and" in your function name, consider refactoring.*

#### ***Tip: Arbitrary variable names can be more effective in certain functions***

*Arbitrary variable names in general functions can actually make the code more readable.*

#### ***Tip: Try to use fewer than three arguments per function***

*Try to use no more than three arguments when possible. This is not a hard rule and there are times it is more appropriate to use many parameters. But in many cases, it's more effective to use fewer arguments. Remember we are modularizing to simplify our code and make it more efficient to work with. If your function has a lot of parameters, you may want to rethink how you are splitting this up.*