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.*

For more guidelines, check out the code layout section of PEP 8 in the notes below.

Writing Clean Code: Nice Whitespace

PEP 8 guidelines for code layout

NEXT