A Short Guide to Naming

Understand how and why to better name modules, classes, functions, and variables.

Tim Ottinger

A Short Guide to Naming

Understand how and why to better name modules, classes, functions, and variables.

Tim Ottinger

This book is available at https://leanpub.com/naming_shortguide

This version was published on 2026-03-03

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Also By Tim Ottinger

Use Vim Like A Pro

Contents

Foreword

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Introduction: What This Book Assumes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

The Foundation We Build Upon

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

So why should we consider naming?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

What is the Point?

The point of good naming is not “to have good naming.”

Nor is it “to have followed the rules.”

Nor is it “standardisation.”

When people reduce the art of naming to rule-following and source-quoting, they have lost the point of this whole exercise.

Arguing Against Naming Skill

You and I can express the same programming concept in dozens of different ways, even inside a single programming language. Some ways are more efficient, more obvious, more elegant, more robust, or more pleasing than others.

All those ways will work.

Occasionally, specific names may be required by a framework that works via reflection to find specially named members or functions. Then we have to comply in order for our software to work.

In other cases, names don’t matter to the underlying technology.

```
def a8f4e2c1(b7d9c3a5: int) -> List[str]:
    e2b6f9d4: List[str] = []
    for c1e4a8b7 in range(1, b7d9c3a5 + 1):
        if c1e4a8b7 % 15 == 0:
            e2b6f9d4.append("FizzBuzz")
        elif c1e4a8b7 % 3 == 0:
            e2b6f9d4.append("Fizz")
        elif c1e4a8b7 % 5 == 0:
            e2b6f9d4.append("Buzz")
        else:
            e2b6f9d4.append(str(c1e4a8b7))
    return e2b6f9d4
```

The compiler doesn’t care.

So why bother choosing “good names?”

LLMs don't care

Perhaps you are generating more of your code with LLMs or other coding assistants, and they also don't care about the names.

When I have code generated for me, the variable names are not what we would consider exemplary. Instead, we get very long and complicated functions with short and cryptic names and lots of comments between steps of an algorithm.

Often these are largely untestable as given.

They also tend to make poor use of existing functions from the standard library and even our own utility functions are ignored (not part of the context).

But the code works. If that's all that matters, then why bother having good names?

It certainly can seem like a self-satisfying, pointless ritual and a needless concession from a certain point of view.

Does it Actually Matter?

If all practices are equally effective, the only reasonable standard is to use personal preference. If all code will only be read, debugged, tested, and used by compilers then names don't really matter.

But that's not how it plays out.

We can't trust our code generators. We certainly should use them, and use them to our advantage, but we need to examine their output to ensure that they are doing the right things, and only the right things.

For that matter, the same is true of many of our developers. Face it, people do things in the worst ways when their entire focus is just getting it off their plates.

But what about when something unexpectedly goes wrong in the middle of the night? Can you find and fix the problem?

Who takes responsiblity for the quality of the program and for its continuing operability? Does the LLM come with guarantees? Does the programmer who likes to delete vowels and add digits to the ends of filenames want to come and puzzle the meaning of the code for you?

You need source code that works for people who live in the codebase.

There is an engineering concept:

> How we do our work makes a difference.

We think that the way to get a better result is to do the work differently.

Good naming in source code is essential for rapid comprehension, maintainability, and software quality.

1. Well-chosen word identifiers improve speed to find defects (by nearly 20%)
2. Descriptive identifiers speed accurate source code comprehension.
3. Where good naming is used, less documentation is necessary (including comments)
4. Poor identifiers are associated with low-quality code (although this may be a tautology, since poor naming is flagged by lint tools as poor code).

Since it does make a difference, we need to understand why and how.

Features of Good Names

Humans read, write, debug, and review code. Even when some AI is used to help generate the code, it's humans who are held accountable. The AI can't take responsibility for programs.

Benner's Principles

Tom Benner wrote a pragmatic and helpful book called "*Naming Things: The Hardest problem in software engineering.*"[^naming]

I recommend this book for further reading, and I will present some nuance here that isn't present in that volume.

[^naming]: add isbn and such here

Benner gives us four criteria that are crucial, which I will paraphrase slightly for our purposes:

1. **Understandability**: The name is known, or needs to be known, by the reader to describe the concept it represents.
2. **Conciseness**: The name has as few words as possible without sacrificing understandability.
3. **Consistency**: Names are used and formatted consistently.
4. **Distinguishability**: Each name stands out as different from the names around it.

There is nothing wrong with any of these suggestions. They are well-explained and the examples are great.

I will lean more on relational, hierarchical, and incremental aspects, hopefully without conflicting with either my work in *Clean Code*, or Tom Benner's work in *Naming*.

Who Cares?

Note that in the earlier example with random names, the compiler had no problem with these words.

Each term is equally distinguishable to the machine, even if the name is 1000 characters long and only has one bit of difference from another name. Understandability is beyond the purview of compilers. Consistency and conciseness are not involved in the generation of machine code.

All of these qualities affect only humans.

The reason for having a "good" name is that it speeds humans in their work.

1. We want people to find the code they need to change, and find it quickly.
2. We don't want them to make errors due to misunderstanding the code.
3. We don't want to spend a lot of time explaining the code.

Who cares? Everyone who will be tasked with understanding this code, whether to extend its functionality, correct a flaw, or improve its performance, cares.

You care.

And it's up to you to care for others.

And now, the question is. "which others am I to consider, and how?"

Readability as a Matter of Familiarity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

The Relationship Nature of Readability

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

The Team as the Audience

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Profitable vs. Unprofitable Struggle

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Building Shared Familiarity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Practical Guidelines

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Context Creates Clarity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

The Hierarchical Nature of Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

What is the domain name associated with this concept?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Windshield Naming

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Unnecessary Context

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

The Context Hierarchy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

System Context

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Module/Package Context

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

File Context

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Class Context

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Method Context

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

The Variable Name Sweet Spot

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Context-Aware Naming Examples

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Over-Contextualization Anti-Pattern

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

A quick sidenote

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Domain Context: Speaking the Language

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Process Context: Understanding the Flow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Mathy or Prosey?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Mathematical Operations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Index Variables in Clear Contexts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

When to Use Minimal Names

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

When Context Is Mixed

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Practical Guidelines for Context-Aware Naming

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Length of Names

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

The Factors To Consider

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Typing Them

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Searching for them

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

So, No Single-Letter Identifiers?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

@ A name carries meaning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

@@ Redundant Context Carries No Meaning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Primacy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Programming Nicknames

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Rules of Thumb

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Extraction and Naming - Two Sides of the Same Coin

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

The Dual Nature of Code Organization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Paragraph markers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Naming Complex Expressions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Option 1: Extract the Expression

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Option 2: Use an explanatory variable to name the Result

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

When Functions Aren't Enough: Extracting Classes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Cohesion: The Principle Behind Good Extraction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

The Naming-Extraction Feedback Loop

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Practical Guidelines for Extraction and Naming

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

When to Extract Functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

When to Extract Classes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Naming Extracted Elements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Nouns And Verbs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

The Nouns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

The Verbs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Other Parts Of Speech

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Frameworks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Rules of Thumb

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Incrementalism

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Naming is a Process

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Intentionally Bad Names

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Obviously Temporary Names

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Renaming Safely

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Caveats

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

The Art of Incremental Improvement

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/naming_shortguide.

Conclusion

Key Takeaways

- Remember that this is about enabling yourself and your teams to work more efficiently and effectively, not about following other people's rules.
- The "best" names are the ones that either are most familiar to teammates, or which are worth learning.
- The name of a variable, function, or class isn't just the name of that item, but the chain of names extending to the class, module, or project.
- You don't have to have a perfect name to begin with; aim to have better names as they occur.

The Naming Checklist

- If you can't think of a good name, use a very bad one.
- Safely rename things when better names appear, though this may require changing the design to make a better name obvious.

Moving Forward

Improving your naming skills is an ongoing journey. Start small:

1. **Be conscious** of the names you choose
2. **Refactor ruthlessly** when you find better names
3. **Get feedback** from your teammates
4. **Practice** these principles daily

Remember: code is read far more often than it's written, especially in the age of agentic code generation. Invest the time to make the names work for your team.

Further Reading

- “Naming” by Tom Benner
- “Clean Code” by Robert C. Martin et al (I wrote the naming chapter).
- “Code Complete” by Steve McConnell
- Your team’s style guide and conventions

* * *

Thank you for reading the Naming Short Guide. Better naming leads to better code, and better code leads to better software.