

What is “Clean Code”?

It's not about
whether code works

A vast majority of time
is spent reading and
understanding code.

What Is Clean Code?



Should be **readable and meaningful**



Should **reduce cognitive load**



Should be **concise** and “to the point”



Should **avoid unintuitive names, complex nesting and big code blocks**



Should follow **common best practices** and patterns



Should be **fun to write** and **to maintain**

**Clean Code Is Easy
To Understand –
Dirty Code Is Not**

Write A Good Story!



Your code is **like an essay**



You are the **author**



Write it such that it's fun and easy to read
and understand!

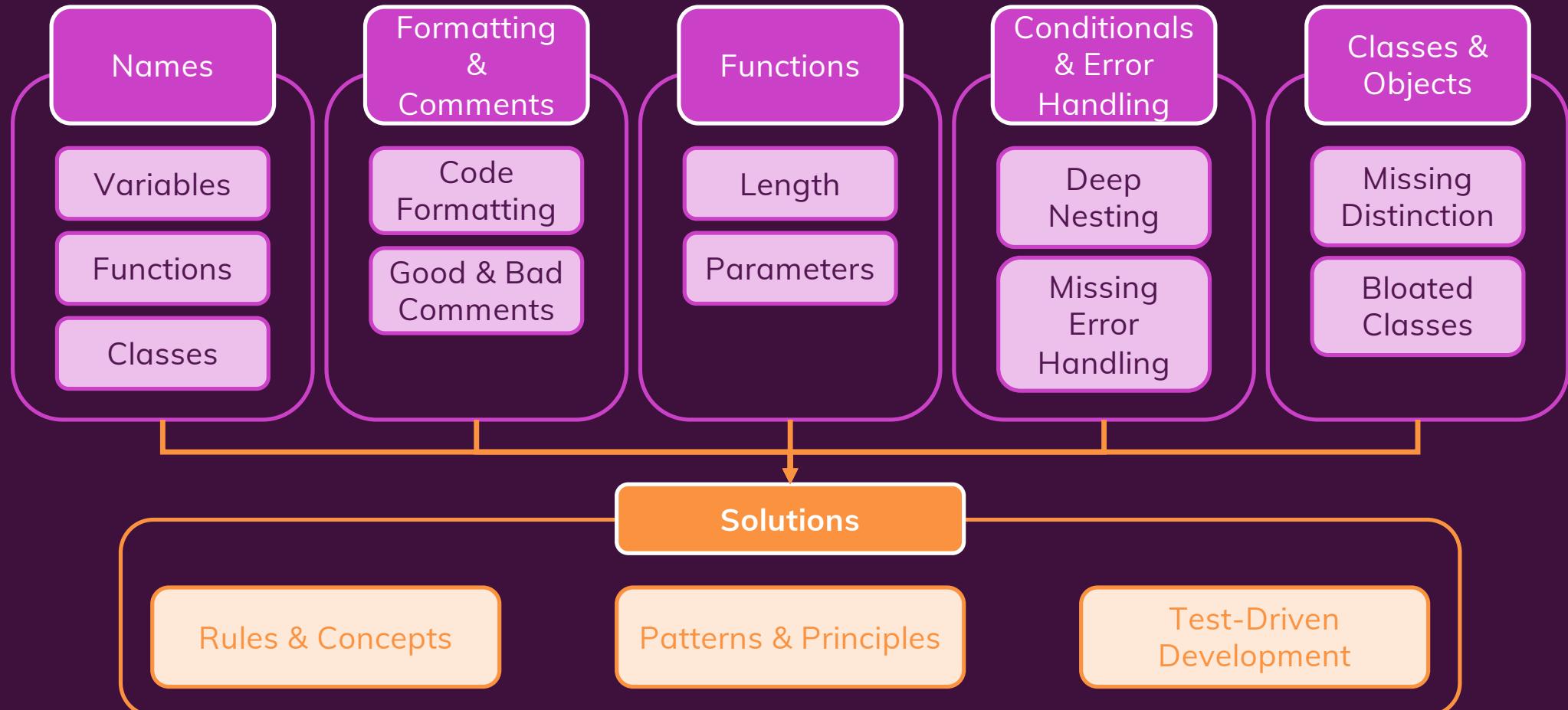
Module Content

Course Content, Structure & Prerequisites

Clean Code vs Patterns & Principles

How To Write Clean Code

Key Pain Points



Course Sections Content



Problems, Rules & Concepts

Learn about bad code and why it's bad

Understand the core rules and concepts you should follow



Demos & Examples

See bad and good code in action

Bad to good code transformations & examples



Challenge Time!

Analyze and transform code on your own

Apply what you learned

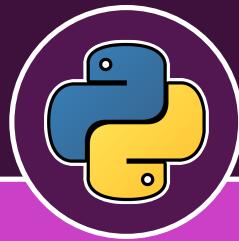
Course Prerequisites

Programming experience is required

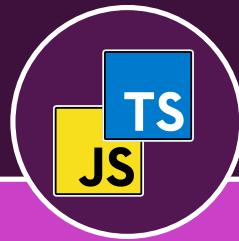


You don't need to be a senior
(not at all)

Programming Languages Used In This Course



Python



JavaScript / TypeScript

You don't need to know or focus on these languages

The concepts taught and examples shown apply to ALL
programming languages

Core Syntax

Variables & Constants

Functions

Classes & Interfaces

Clean Code Doesn't Require Strict Typing

```
function add(num1: number, num2: number) {  
  return num1 + num2;  
}
```

Types can help preventing errors and can improve readability

But code can also be 100% readable and meaningful without types

```
def add(num1, num2):  
  return num1 + num2
```

About The Code Examples Shown In This Course



Short, Focused Examples



Code snippets: Most examples
won't execute



Examples don't use a particular
programming style / paradigm

Functional vs OOP vs Procedural

This course doesn't focus on a specific paradigm



Paradigm-specific rules, conventions, patterns and guides should also be considered



The rules, concepts, ideas and patterns shown in this course apply in general!



The Core Principles & Rules Always Apply!

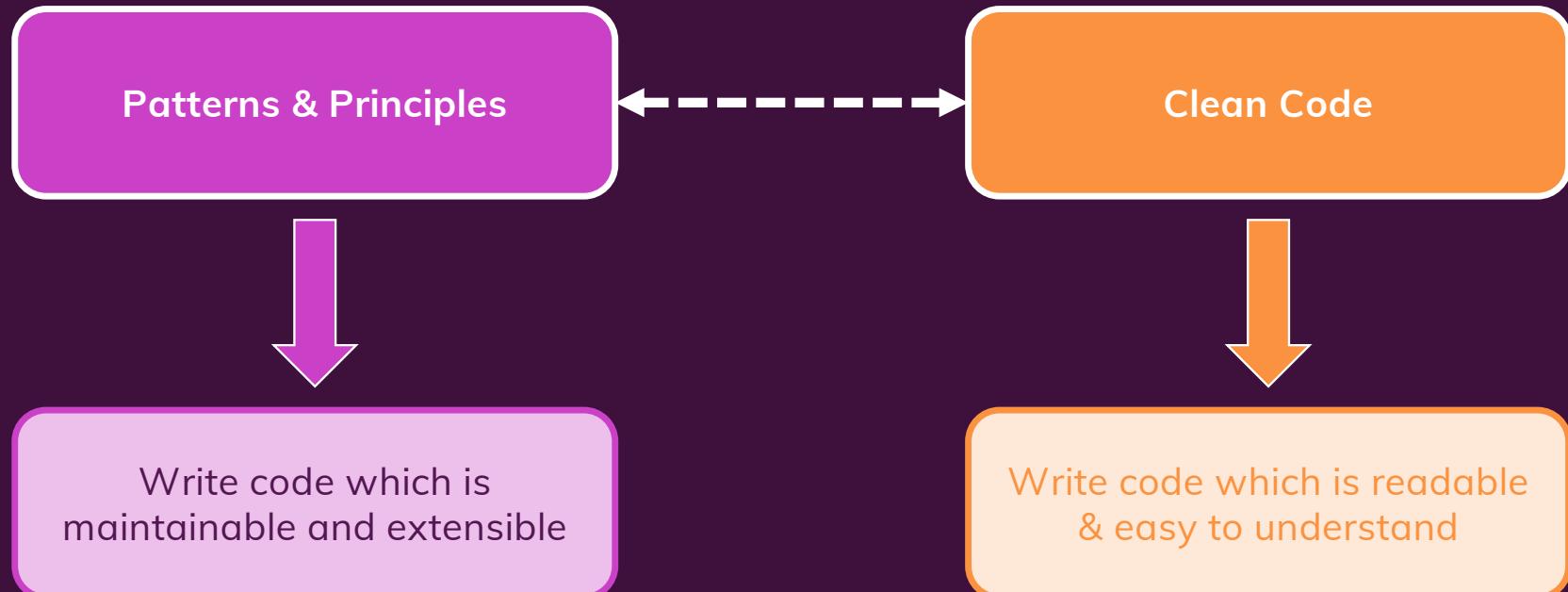
No matter which **programming language or style** you're using...

...you still want **readable** and meaningful names

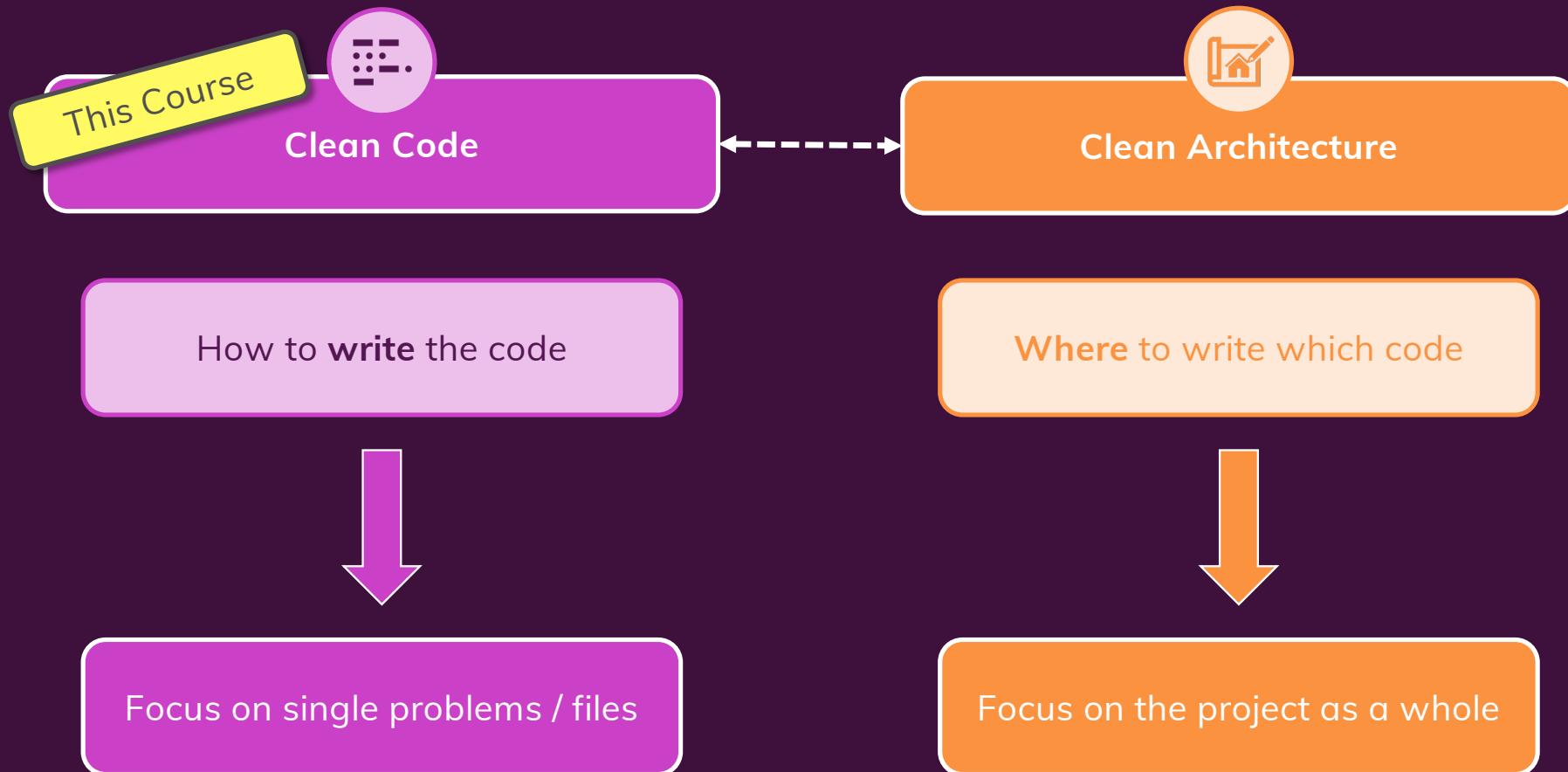
...you still want **slim, concise** functions or methods

...you still want **understandable** code flow

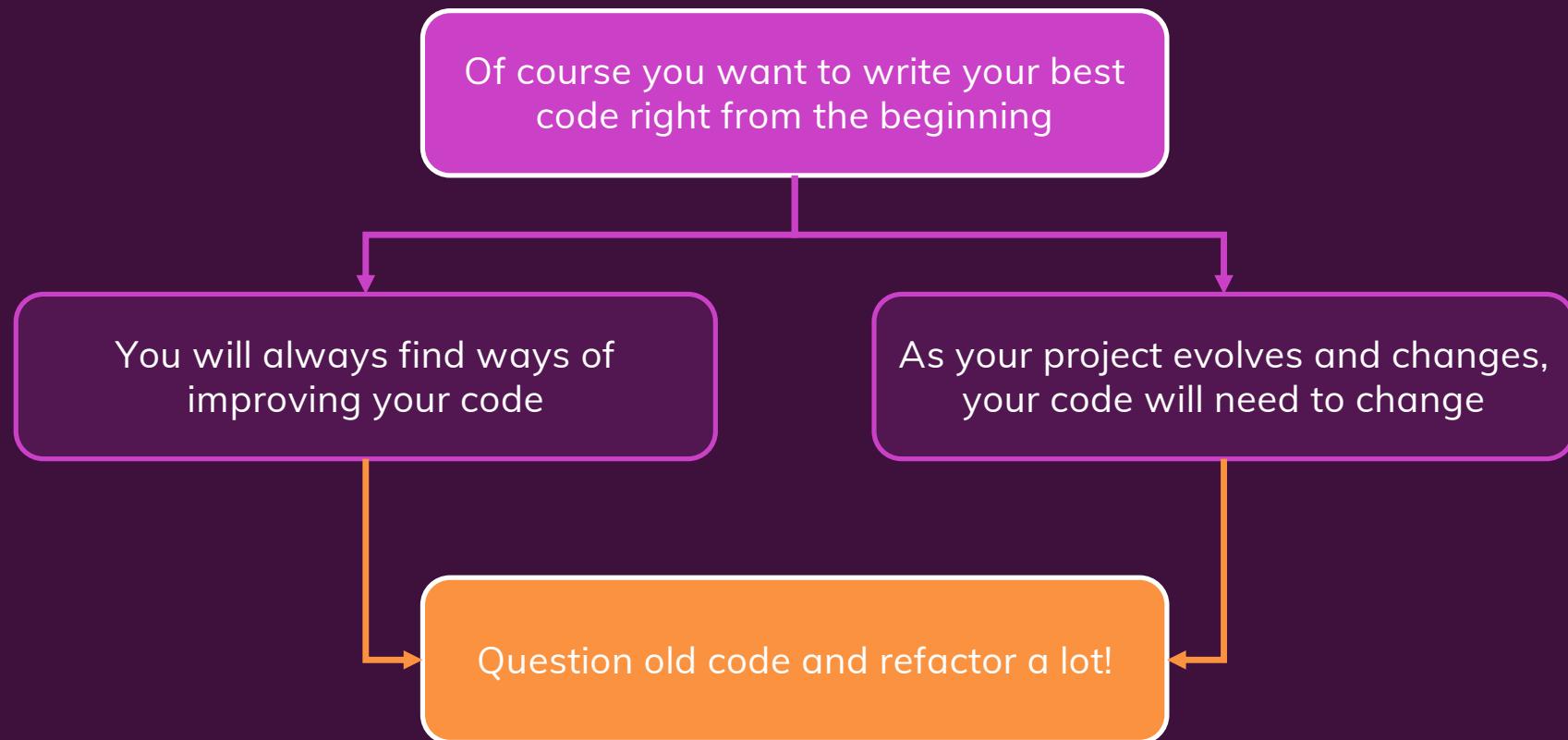
Clean Code and Patterns & Principles



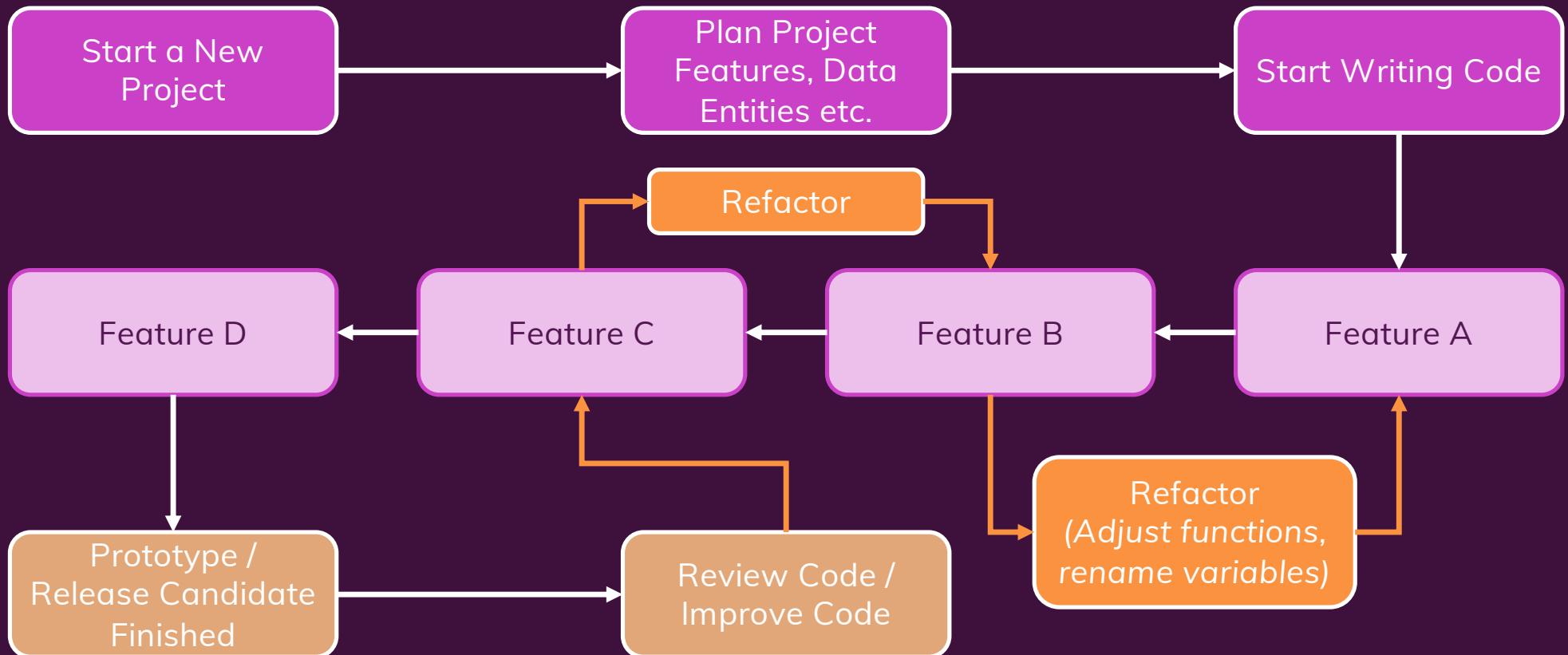
Clean Code vs Clean Architecture



Clean Code Is Written Over Time!



How To Write Clean Code



Embrace Refactoring!



Refactoring today is work you save tomorrow



A codebase can only survive and stay maintainable if it's continuously improved and refactored



Pro tip: Whenever you add something new, try to improve existing code along the way

Clean Code vs Quick Code

