ACADEMIND

What is “Clean Code”?

ACADEMIND

It’s not about whether code works

ACADEMIND

Wasting time reading messy code

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

ACADEMIND

What Is Clean Code?

Should be **readable and meaningful**

Should **reduce cognitive load**

easy to understand

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**

ACADEMIND

Clean Code Is Easy To Understand – Dirty Code Is Not

ACADEMIND

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!

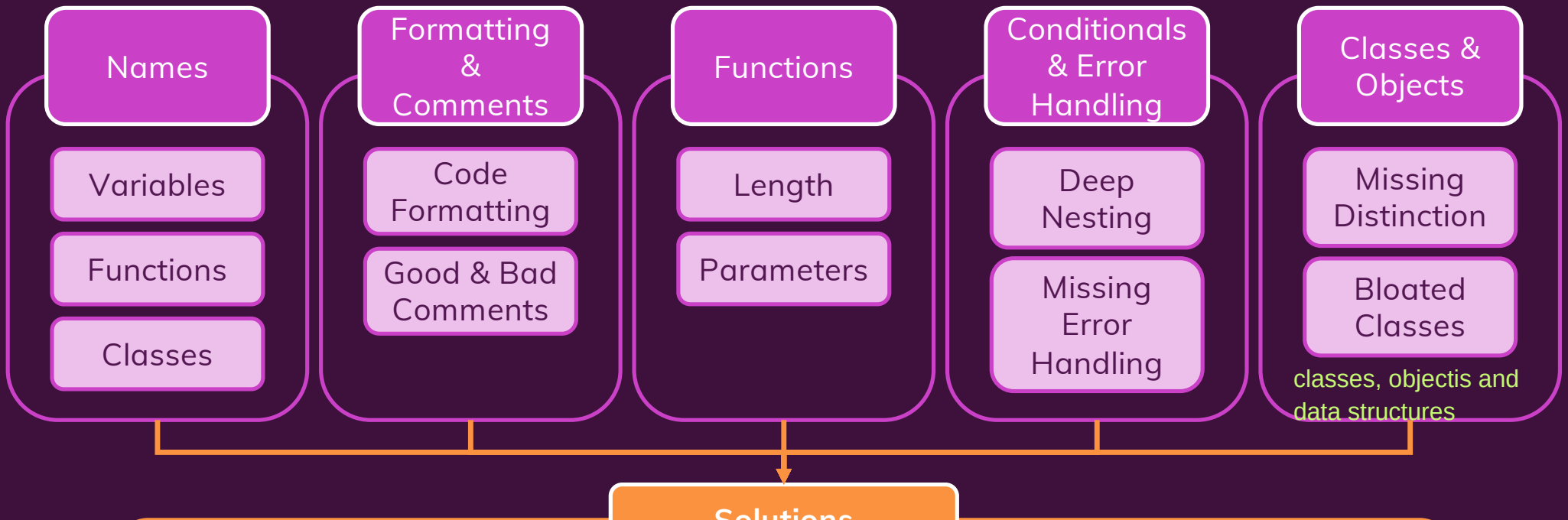
ACADEMIND

Module Content

- Course Content, Structure & Prerequisites
- Clean Code vs Patterns & Principles
- How To Write Clean Code

ACADEMIND

Key Pain Points



ACADEMIND

Course Sections Content

Problems, Rules & Concepts

Demos & Examples

Challenge Time!

Learn about bad code and why it's bad

Understand the core rules and concepts you should follow

good
See bad and ~~code~~ code in action

Bad to good code transformations & examples

Analyze and transform code on your own

Apply what you learned

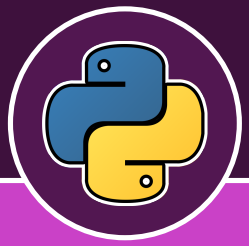
Course Prerequisites

Programming experience is required

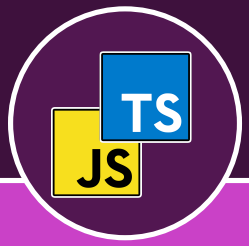
You don't need to be a senior
(not at all)

ACADEMIND

Programming Languages Used In This Course



Python



JavaScript / TypeScript

Typescript adds strict typing

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

The concepts taught and examples shown apply to ALL programming languages

ACADEMIND

Core Syntax

Variables & Constants

Functions

Classes & Interfaces

ACADEMIND

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

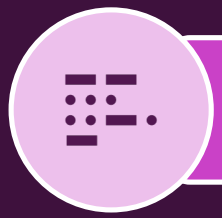
But code can also be 100% readable and meaningful without types

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

ACADEMIND

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

ACADEMIND

Functional vs OOP vs Procedural

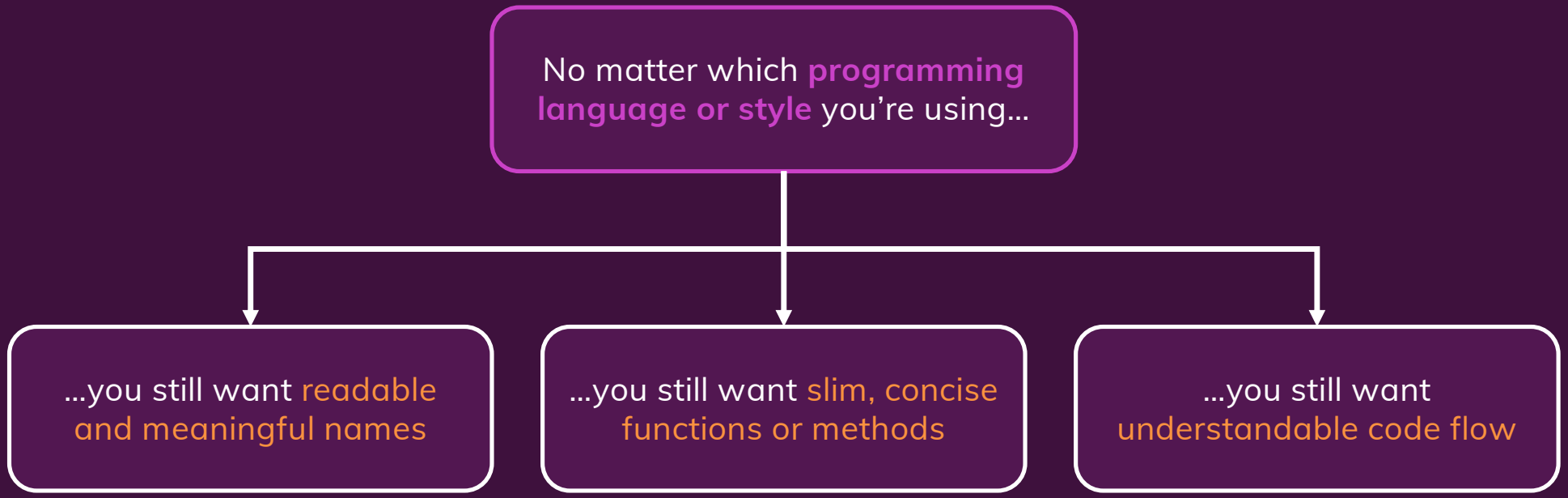
This course doesn't focus on a specific paradigm

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

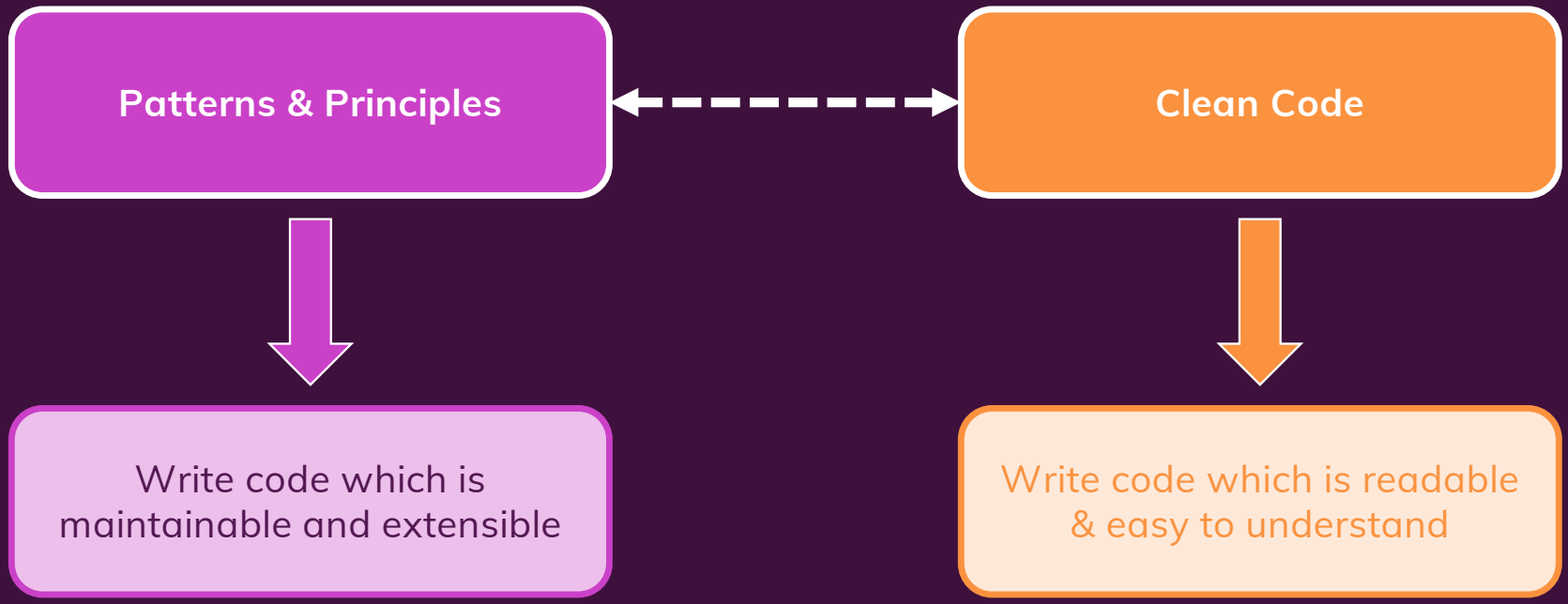
The rules, concepts, ideas and patterns shown in this course apply in general!

ACADEMIND

The Core Principles & Rules Always Apply!

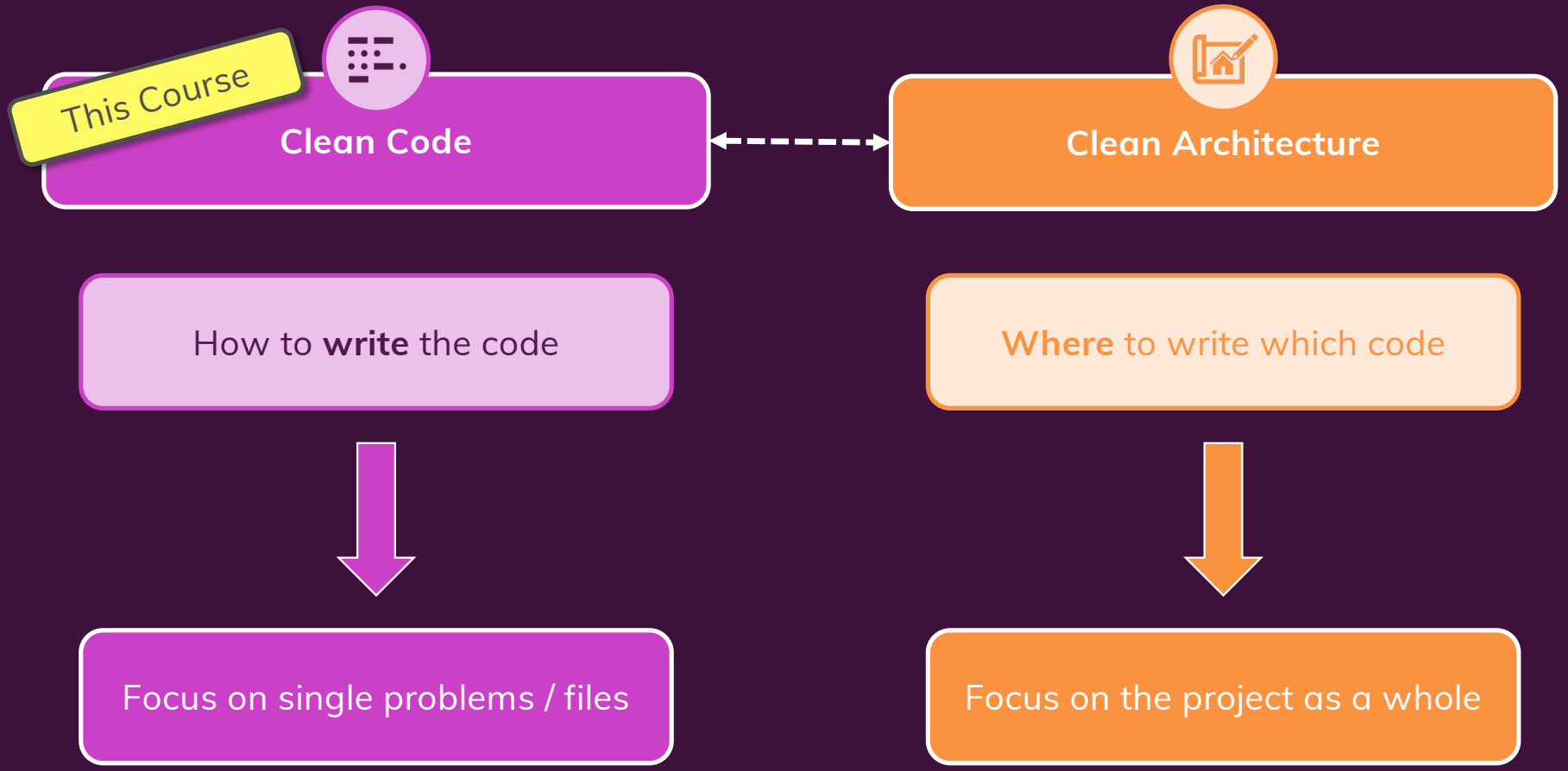


Clean Code and Patterns & Principles



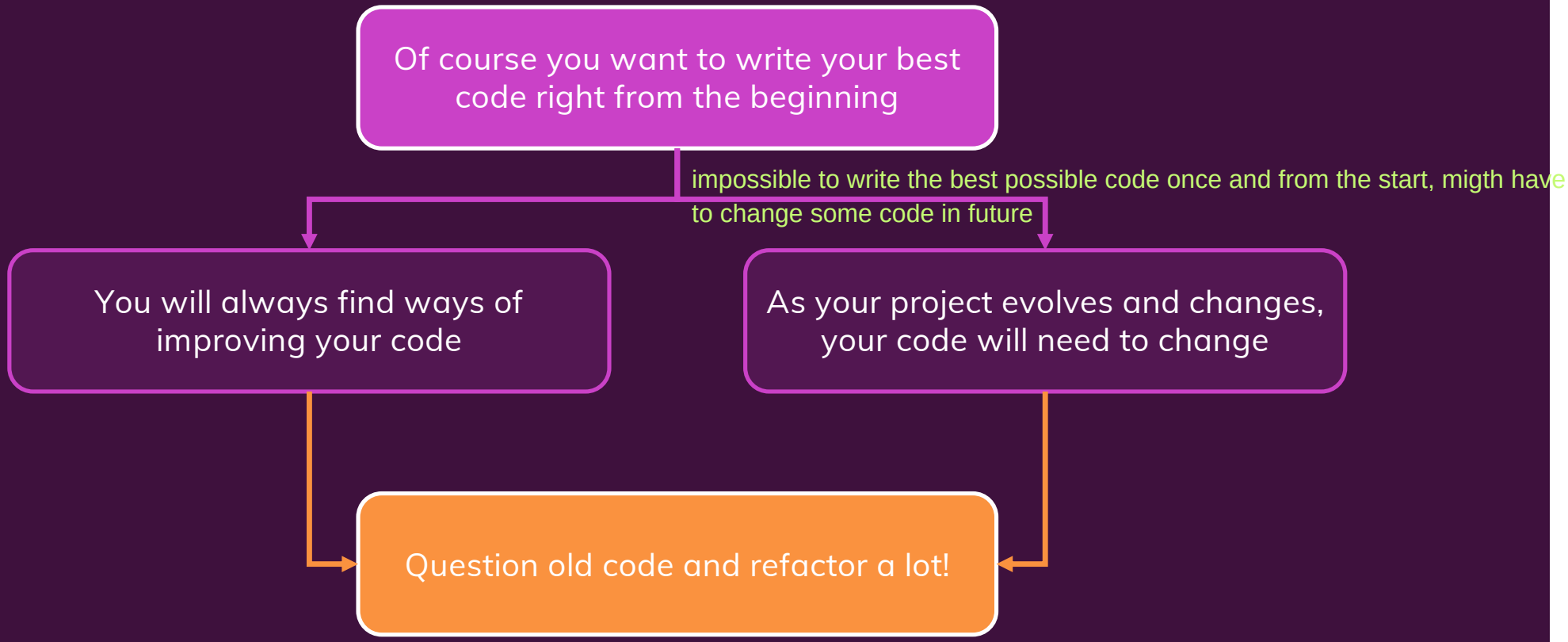
ACADEMIND

Clean Code vs Clean Architecture

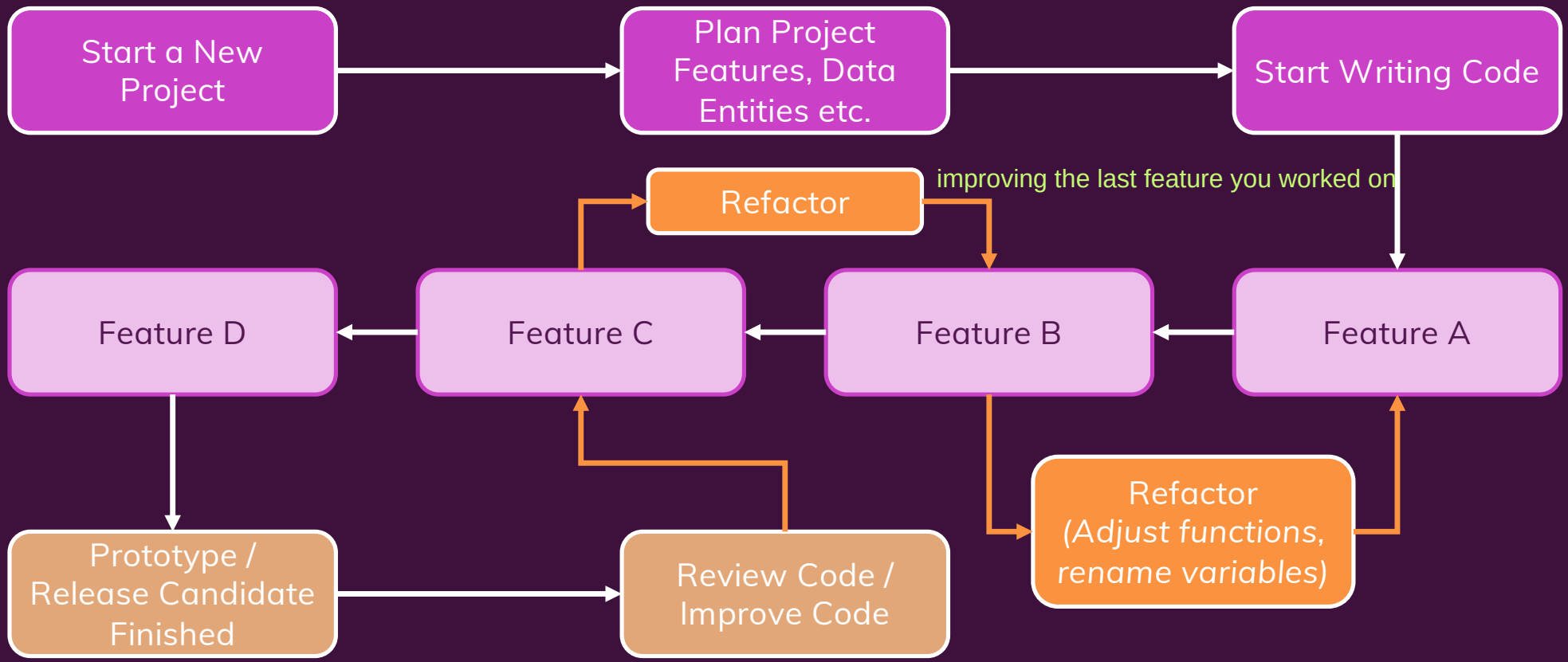


ACADEMIND

Clean Code Is Written Over Time!



How To Write Clean Code



ACADEMIND

Embrace Refactoring!

Refactoring today is work you save tomorrow

better code!

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

ACADEMIND

Clean Code vs Quick Code