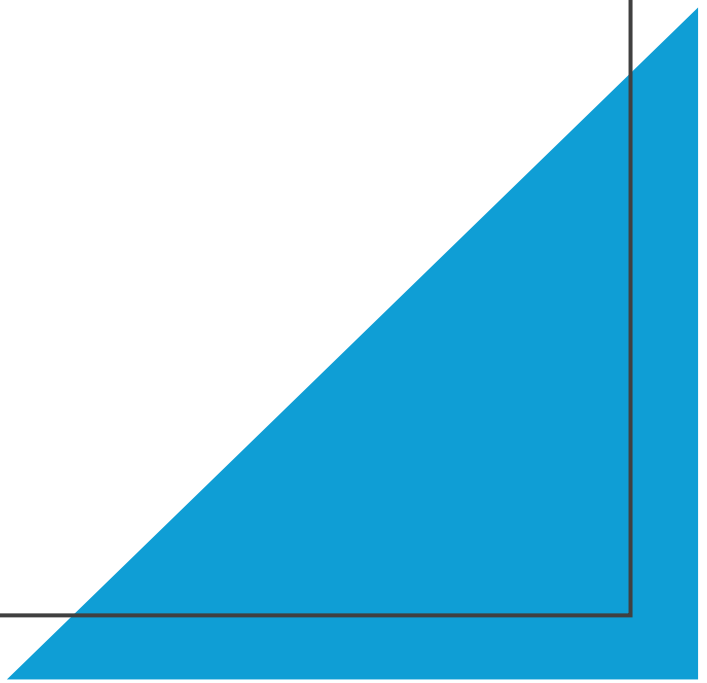
FP with Java

An Introduction...



Content

1

Part I

- FP is unfamiliar
- 3 Core Principles of FP design
- Pattern 1: Functions as parameters

2

Part II

- Map – Filter – Collect

3

Part III

- FP vs. OOP

Sources / Credits

[1] Functional Design Patterns / Scott Wlaschin

[2] The Functional Toolkit / Scott Wlaschin

[3] OOP vs. FP with C# and F# / Urs Enzler

Part I

FP in General

Functional programming is ~~scary~~ unfamiliar

Functor

Currying

Catamorphism

Applicative

Monad

Monoid

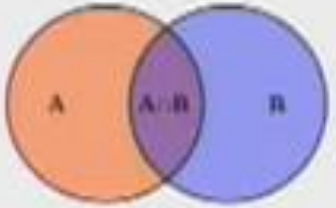
- 3 Core Principles of FP design
- Pattern 1: Functions as parameters
- Pattern 2: Composing multi-parameter functions
- Pattern 3: "bind"
- Pattern 4: "map"
- Pattern 5: Monoids

These "patterns" can be built-in or not, depending on your programming language

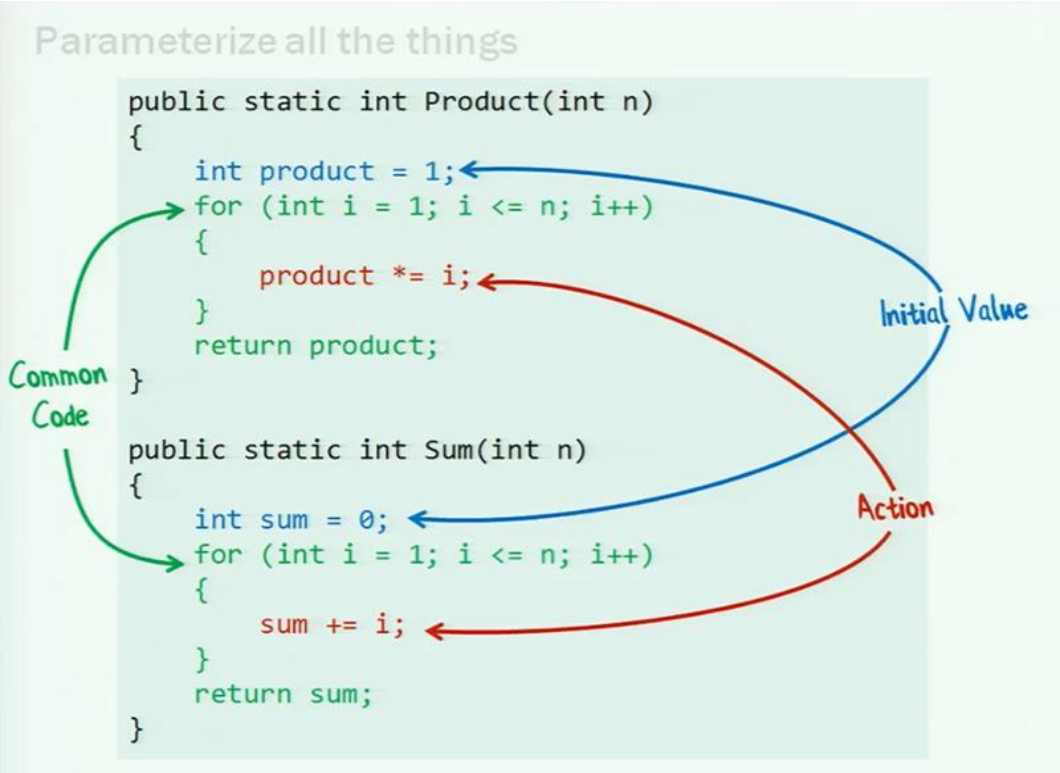
Functions are things

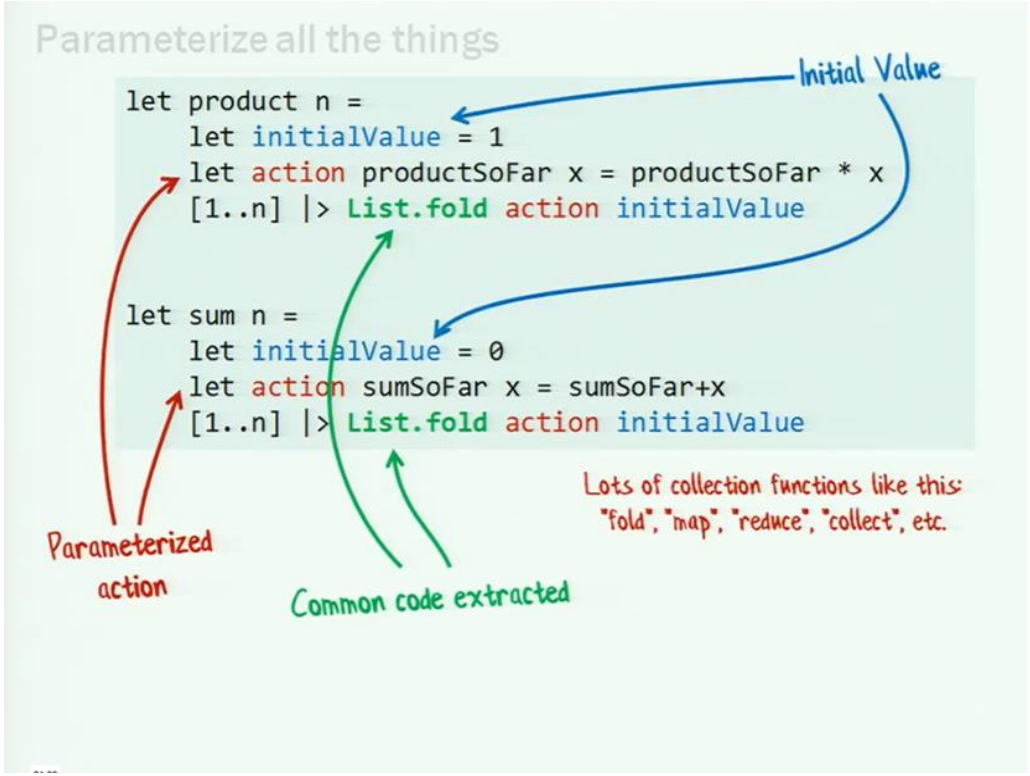
Composition everywhere

Types are not classes



I'm skipping referential transparency, purity, etc

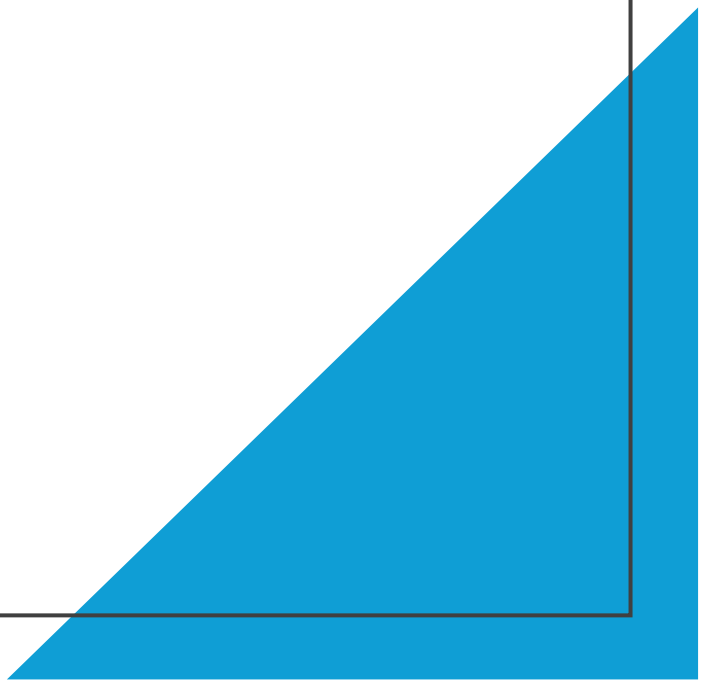


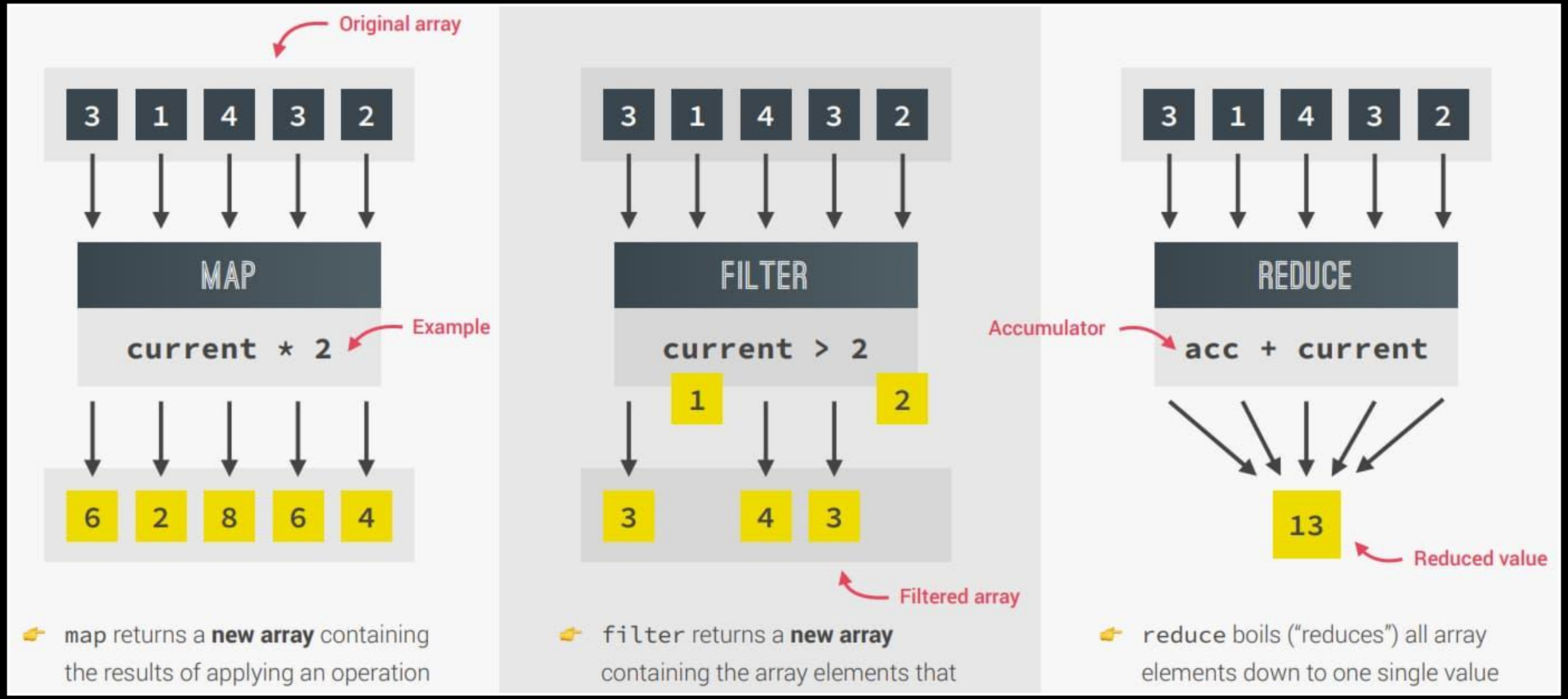


Pattern 1 - Functions as Parameters

Part II

FP with Java



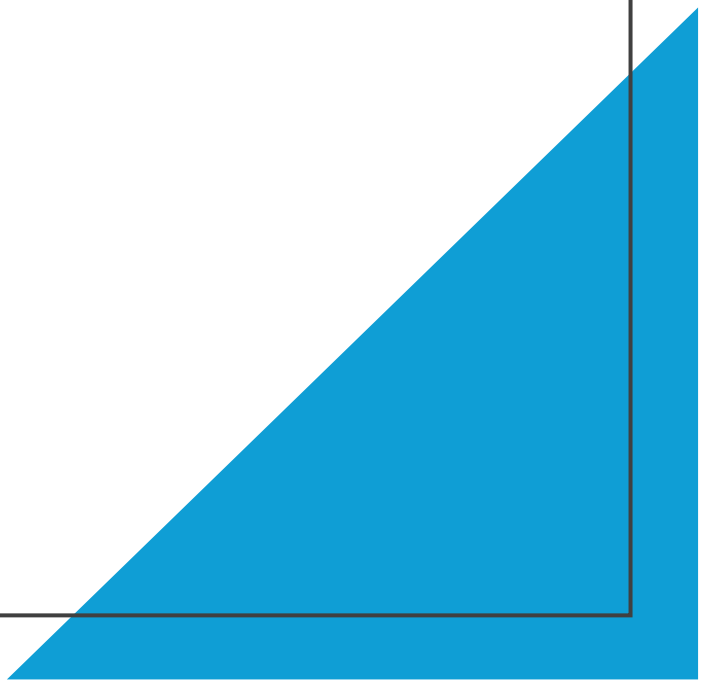


Map – Filter – ~~Reduce~~ Collect

Map – Filter – Collect

Part III

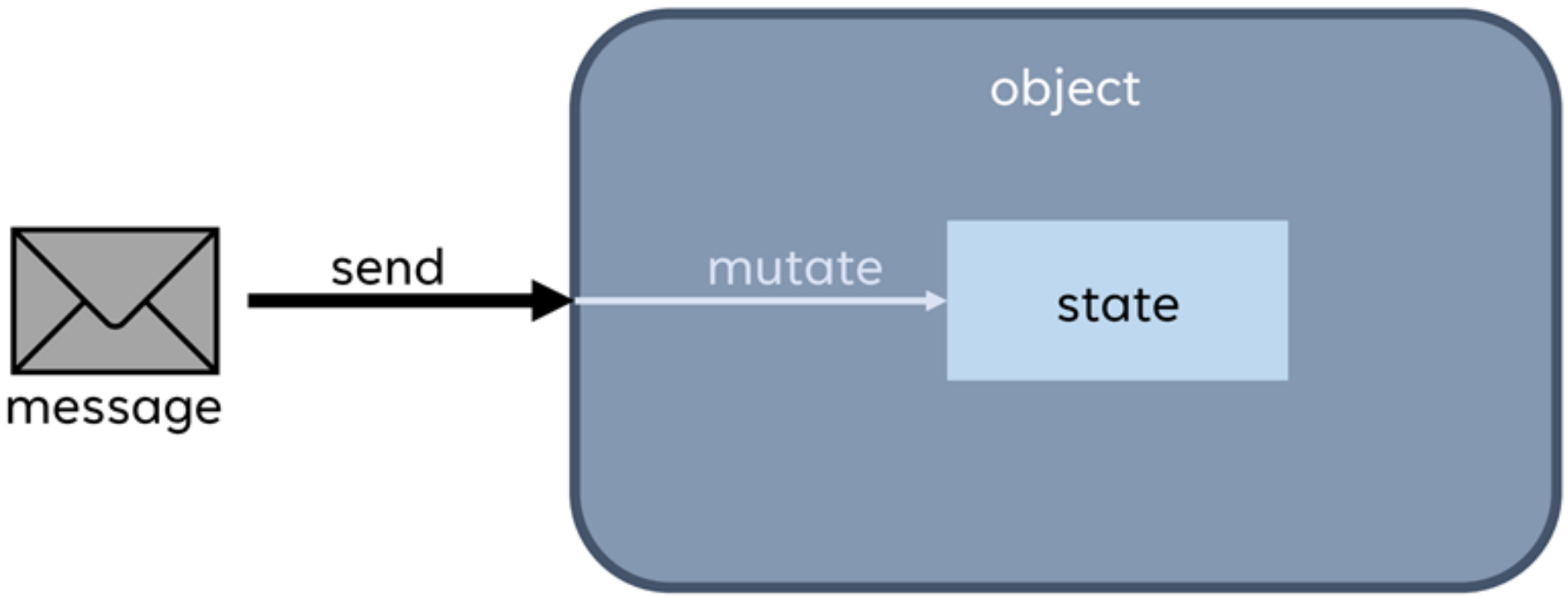
FP vs. OOP



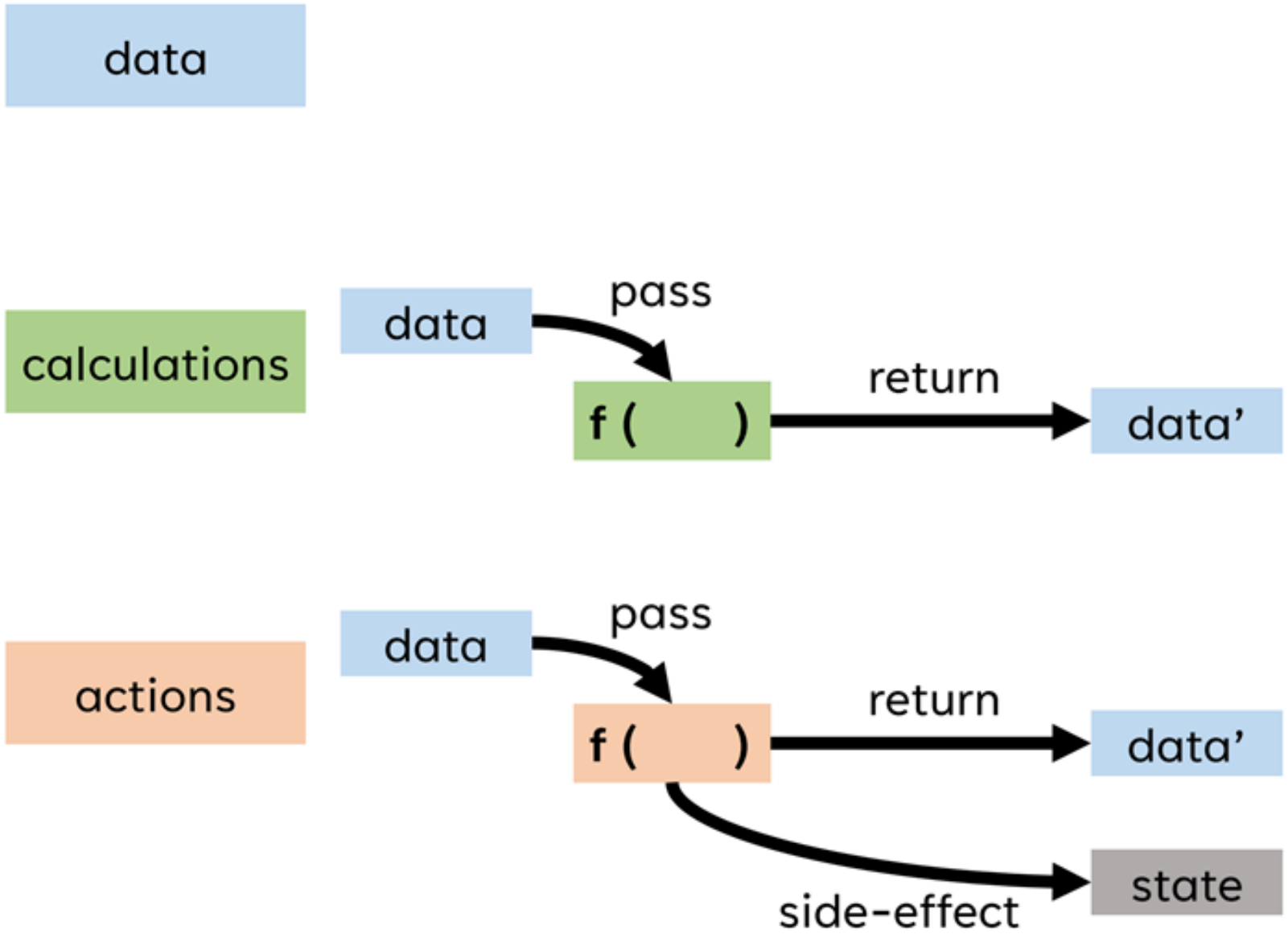
OO pattern/principle	FP equivalent
Single Responsibility Principle	Functions
Open/Closed principle	Functions
Dependency Inversion Principle	Functions, also
Interface Segregation Principle	Functions
Factory pattern	You will be assimilated!
Strategy pattern	Functions again
Decorator pattern	Functions
Visitor pattern	Resistance is futile!

Seriously, FP patterns are different

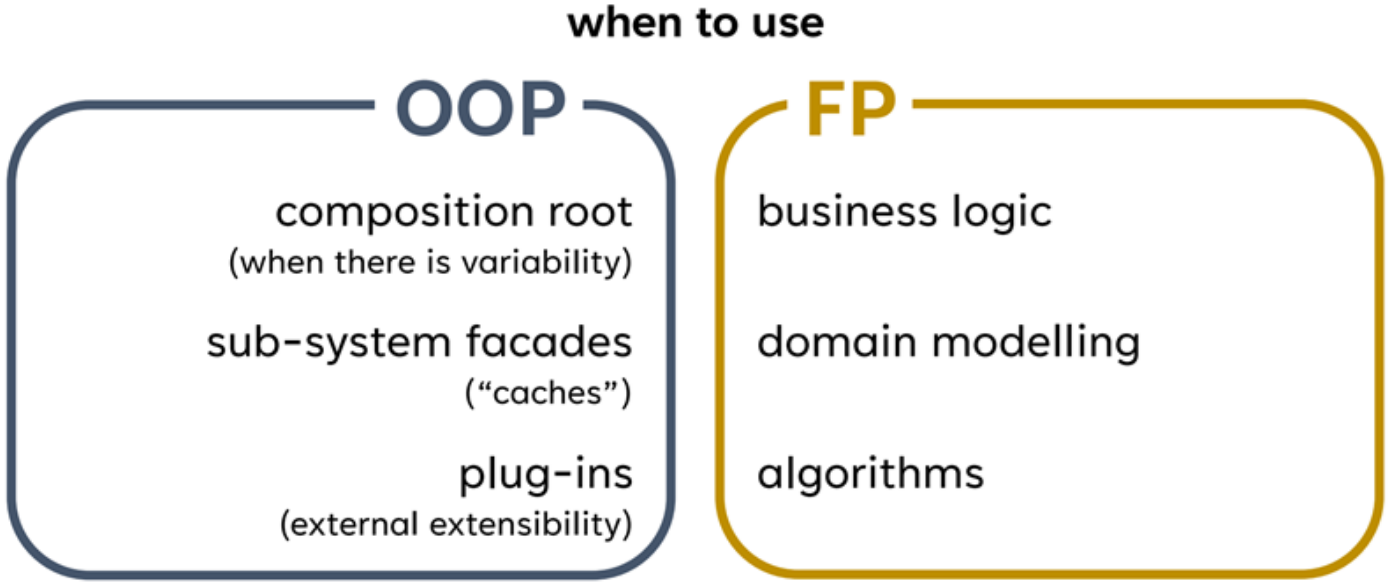
OOP (simplified)



FP (simplified)



FP vs. OOP



Mix concepts from both paradigms!