Programming language Concepts

	Rust ✓	Ruby	OCaML	
①	Haskell	Julia ✓	Go ✓	
②	Python	Swift ✓	BASIC	
	R	Fortran	Pascal	Niklaus
	Java	Algol	LISP	Wirth
	Cobol	APL	Perl	
	C	C++		

Functional

Haskell

No "stored" data

Prolog - Logic Programming

Relational Programming

Imperative

Python

Goal → Instructions

Justify why the steps work

```
sum []     = 0
sum (x:xs) = x + (sum xs)
```

"Declarative"

Declarative

What we want

↓

Computation as rewriting

λ-calculus (Alonzo Church)

Type inference

Imperative

How to do it

↓

Types - abstract datatypes

- object-oriented programming

Error handling

Concurrent programming

Event driven programing

Niklaus Wirth

Algorithms + Data Structures = Programs

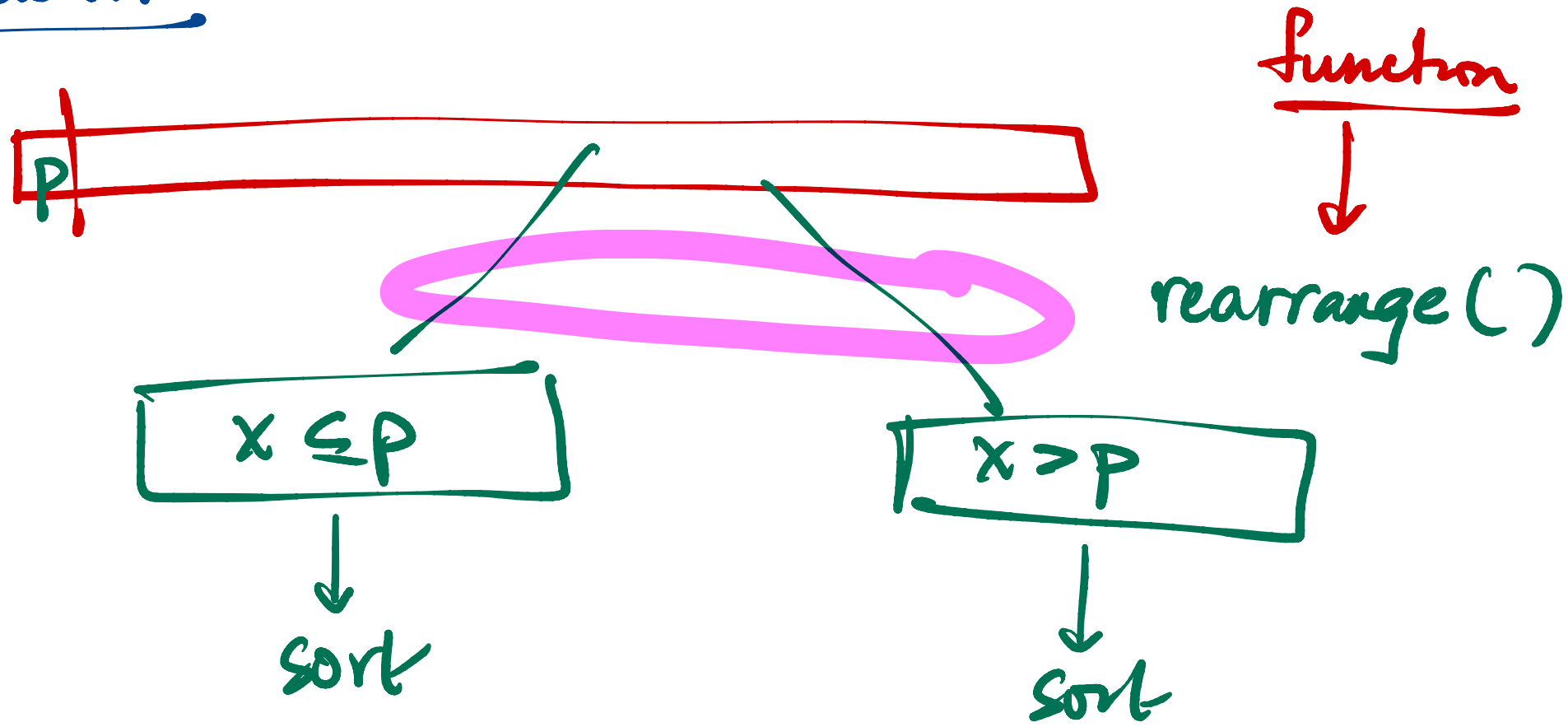
Control

Data

Fundamental concept underlying programming is

Abstraction

Quicksort



$$l_1 \mathbin{+\!\!+} [p] \mathbin{+\!\!+} l_2$$

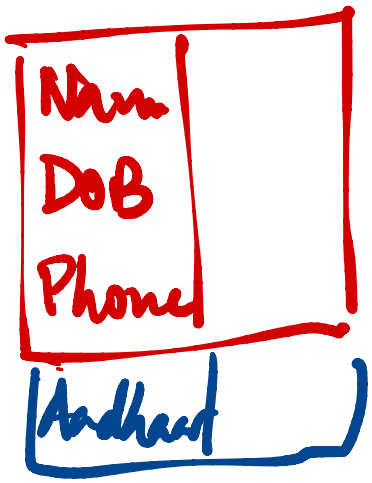
Control flow

Functions are used for abstraction

Abstract → "Concrete" Refinement

Data refinement

Student personal data

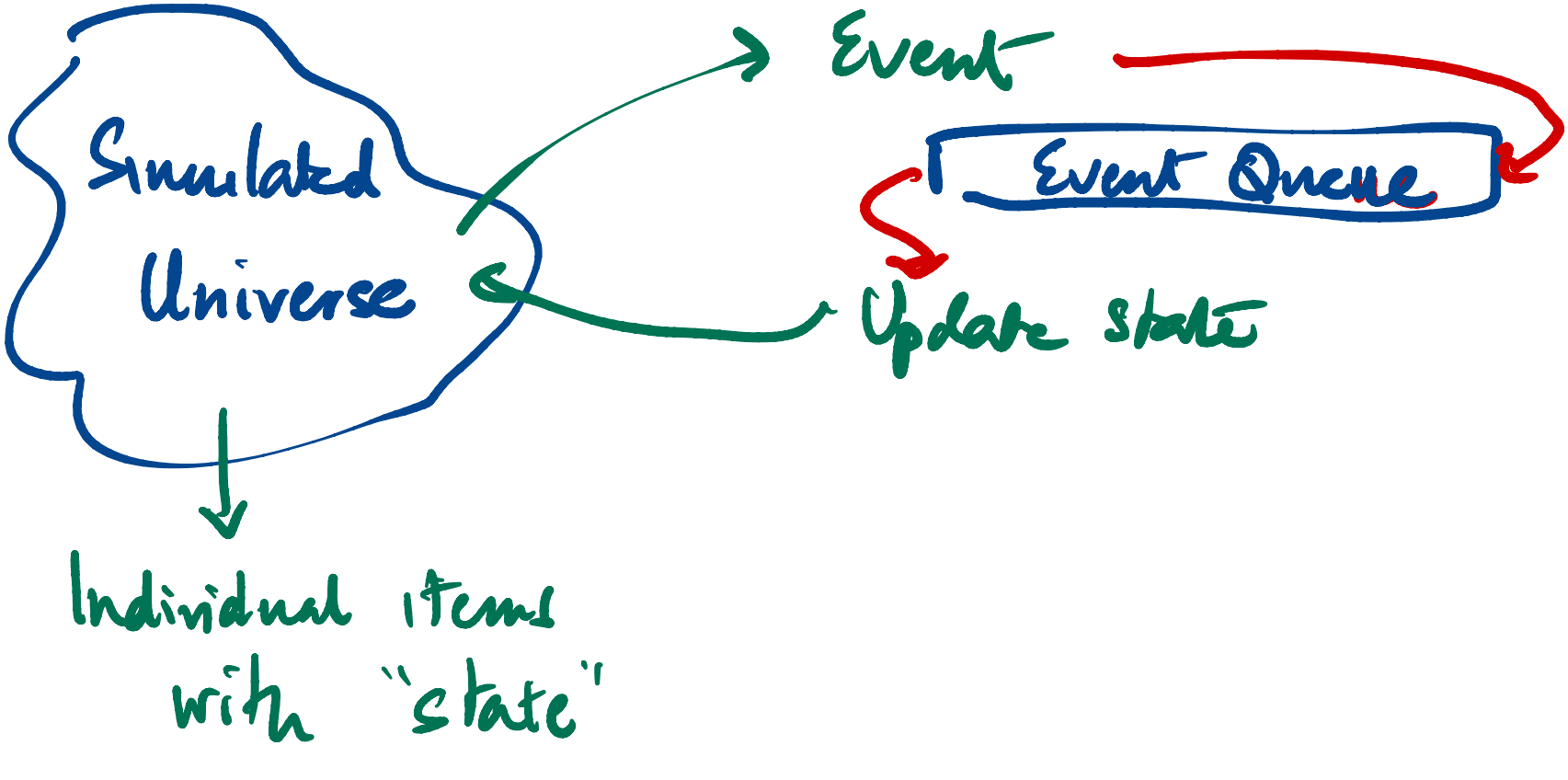


History of "objects"

Simula

Simulation language

```
loop
    e ← head(queue)
    simulate(e)
forever
```



Abstract Data Type

Internal state (private)

Available functions (public interface) — behavioral specification

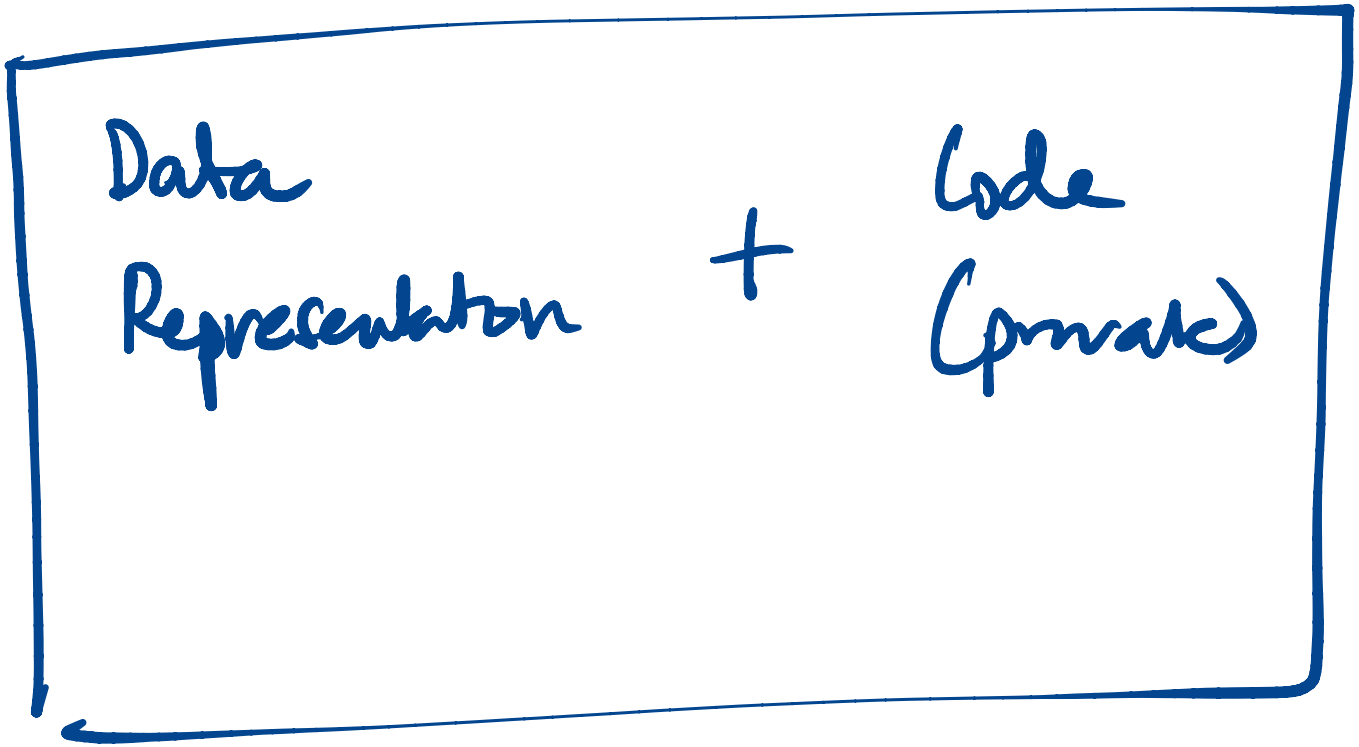
Priority Queue ⟶ Heap

add(p)

remove_max()

"Private" binary tree with suitable structure

Representation of information change → functions have to be updated



Java
Python
Rust