# 4. Imperative programming

Imperative program consists of commands for the computer to perform.

In much the same way that the imperative mood in natural languages expresses commands, an imperative program consists of commands for the computer to perform.

Imperative programming focuses on:

* describing how a program operates
* how to execute
* defines control flow as statements that change a program state.

Number object set to 3 -> now be 5

Customer object -> change name to “John Smith”

State and mutation are the essence of imperative programming!

# 5. Declarative programming

Programming is done with expressions or declarations instead of statements.

In functional programming we are defining terms, not giving orders.

Declarative programming:

* focuses on what to execute
* defines program logic but not detailed control flow

If we bound number 3 to identifier id – that is we gave another name, an alias to number 3, we cannot go and say: now be 5! The number doesn’t change!

We define a function – a mapping – of how to go from number 3 to number 5

There is not state, no mutation!

# 8. Immutability

Threading state can be simplified by using STATE MONAD!

Pros of immutability:

* Programs are less complicated and easier to reason about
* No need for defensive copies of objects
* Easy to test

Good for sharing information between threads in a multi-threaded environment since they don't need to be synchronized

# 9. Pure Functions

Benefits of pure functions:

* They’re easier to reason about
* They’re easier to combine
* They’re easier to test
* They’re easier to debug
* They’re easier to parallelize

# 10. Higher order functions

Type inference

# 15. Pattern matching

Match expression

Triangle of base \* height

A = h \* b/2

More on pattern matching:

* OR / AND pattern
* CONS pattern
* List / Array pattern
* Type test pattern
* Active pattern