Accelerated
TDD
@phil_nash

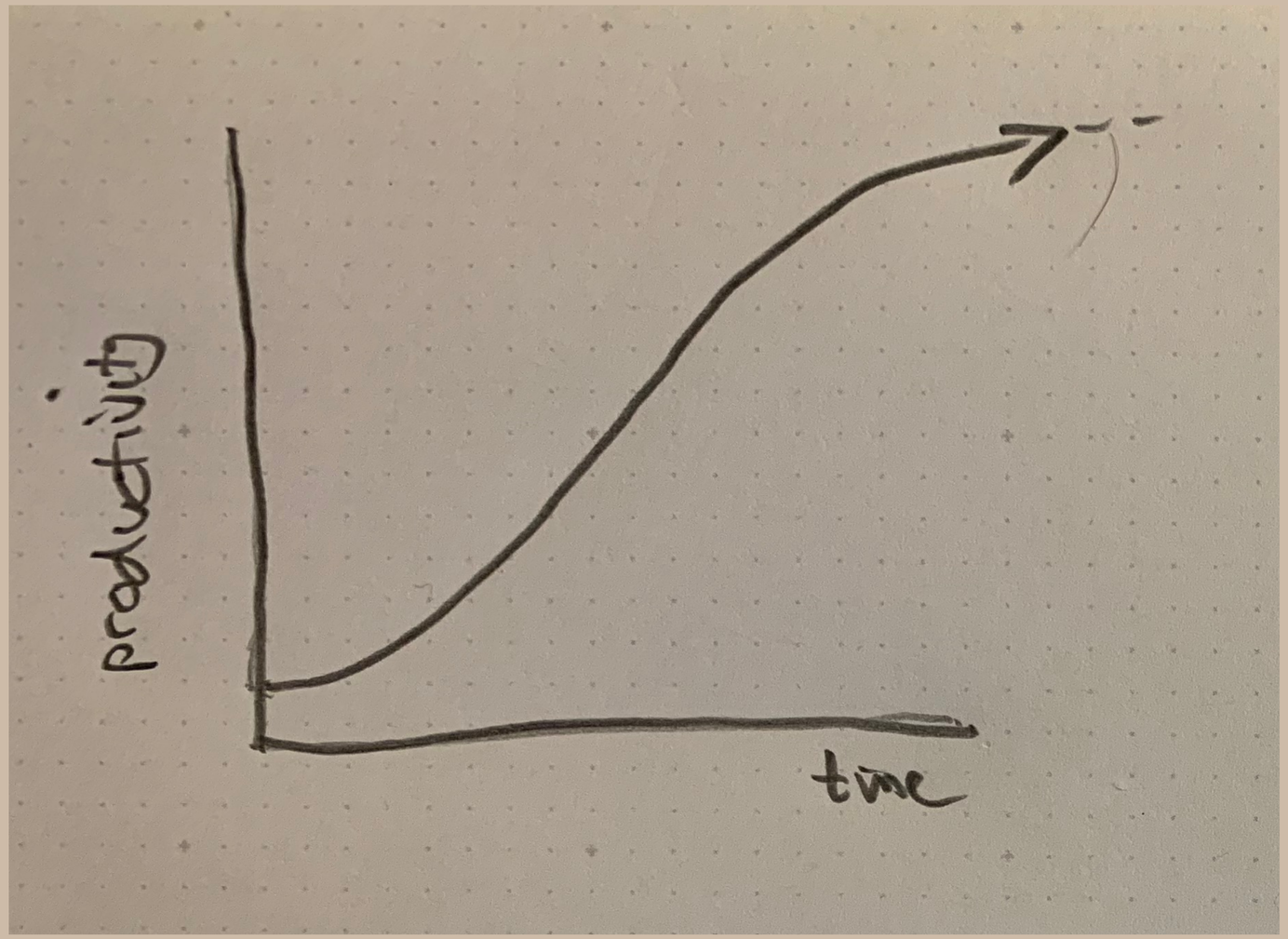
Why Accelerated ?

Why Accelerated ?

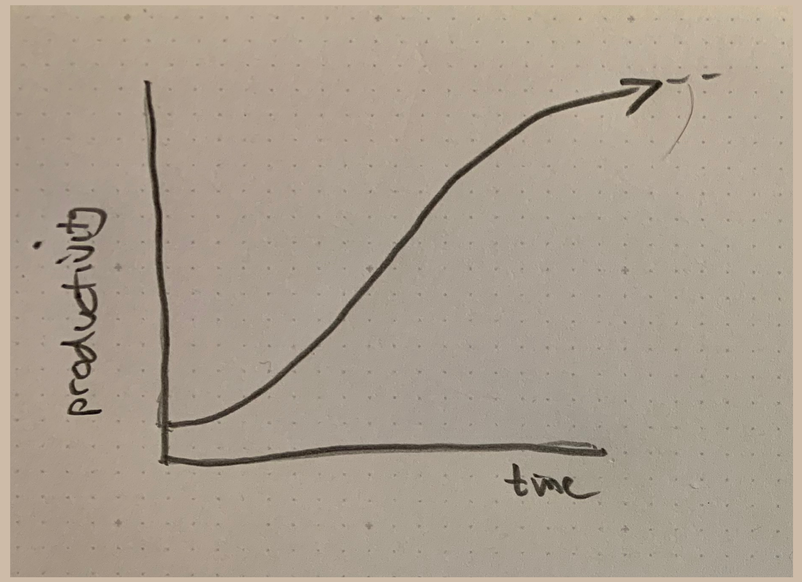
Many dead-ends - learn from hard experience

Can take while to become productive

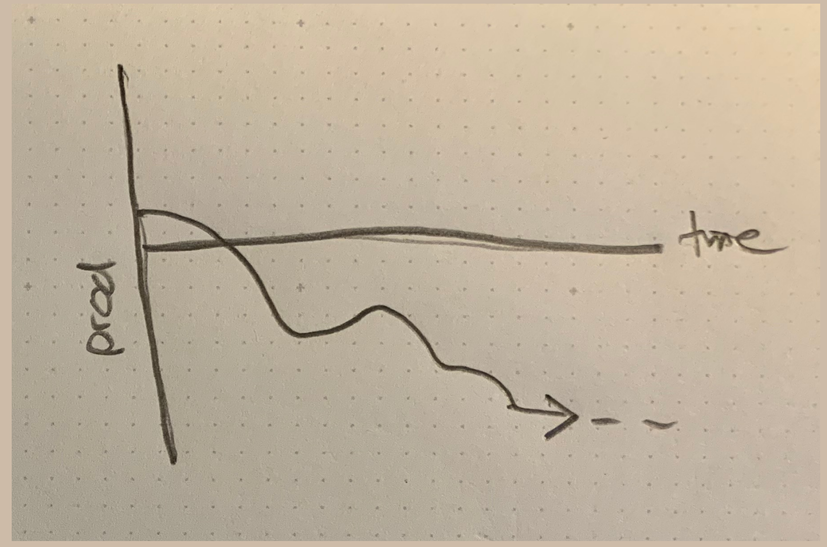
Many give up on TDD too soon



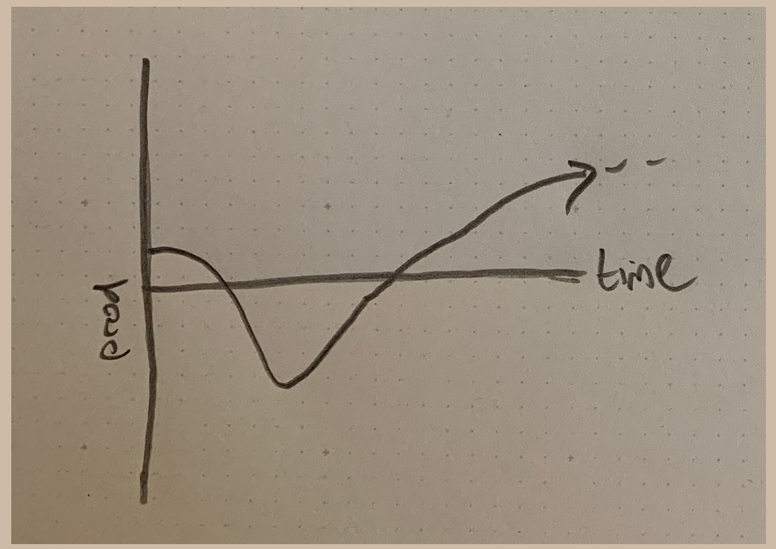
How TDD is sold



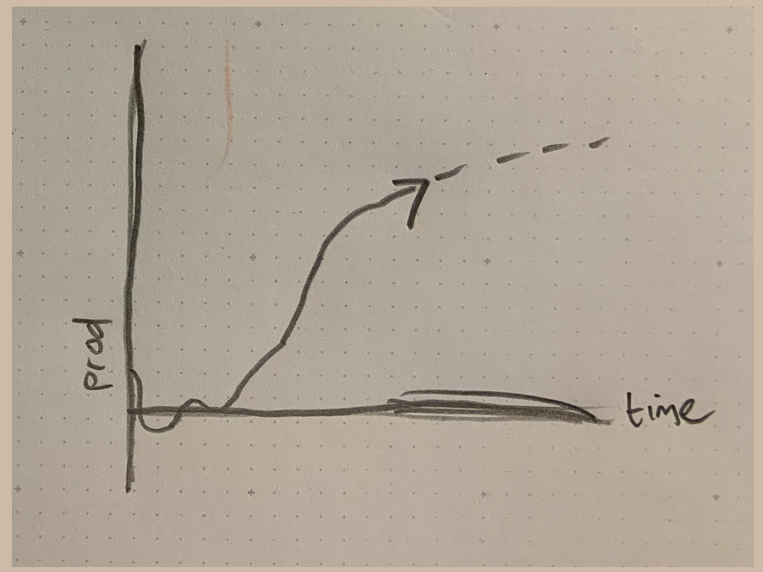
How TDD is sold



What most experience



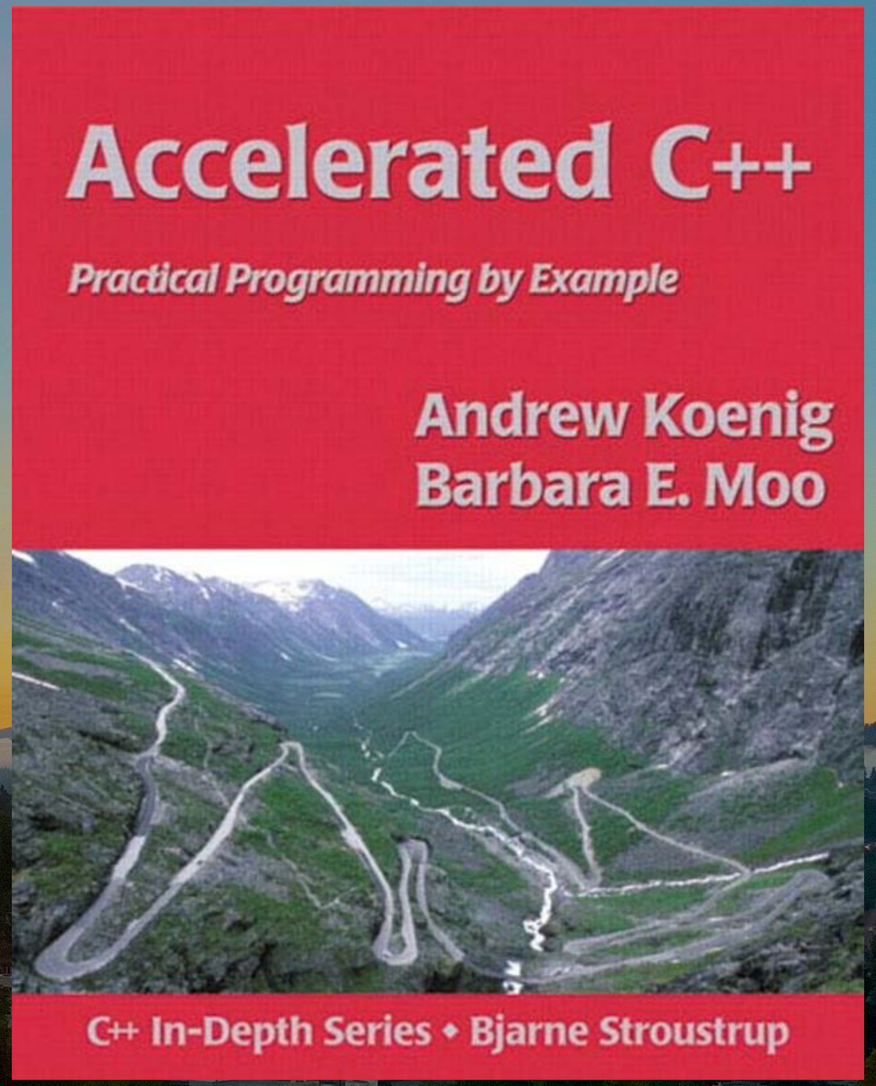
… if you persist



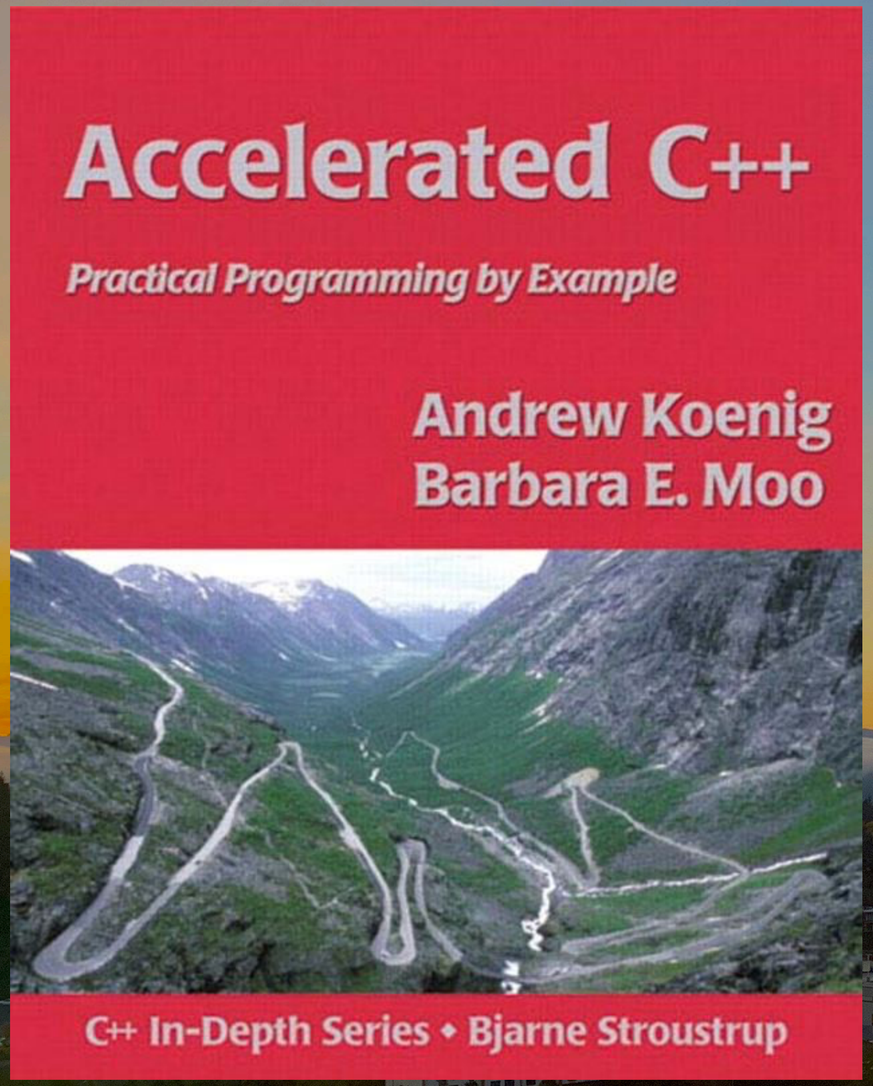
What we hope to do

How Accelerated?

How Accelerated?



How Accelerated?



Stop teaching ~~C++~~ Unit Testing

Practical Examples, early

Start high level

Good information

The CO$T of bugs

The CO$T of *bugs*

The CO$T of *bugs*

2002 NIST study

The CO$T of *bugs*

2002 NIST study

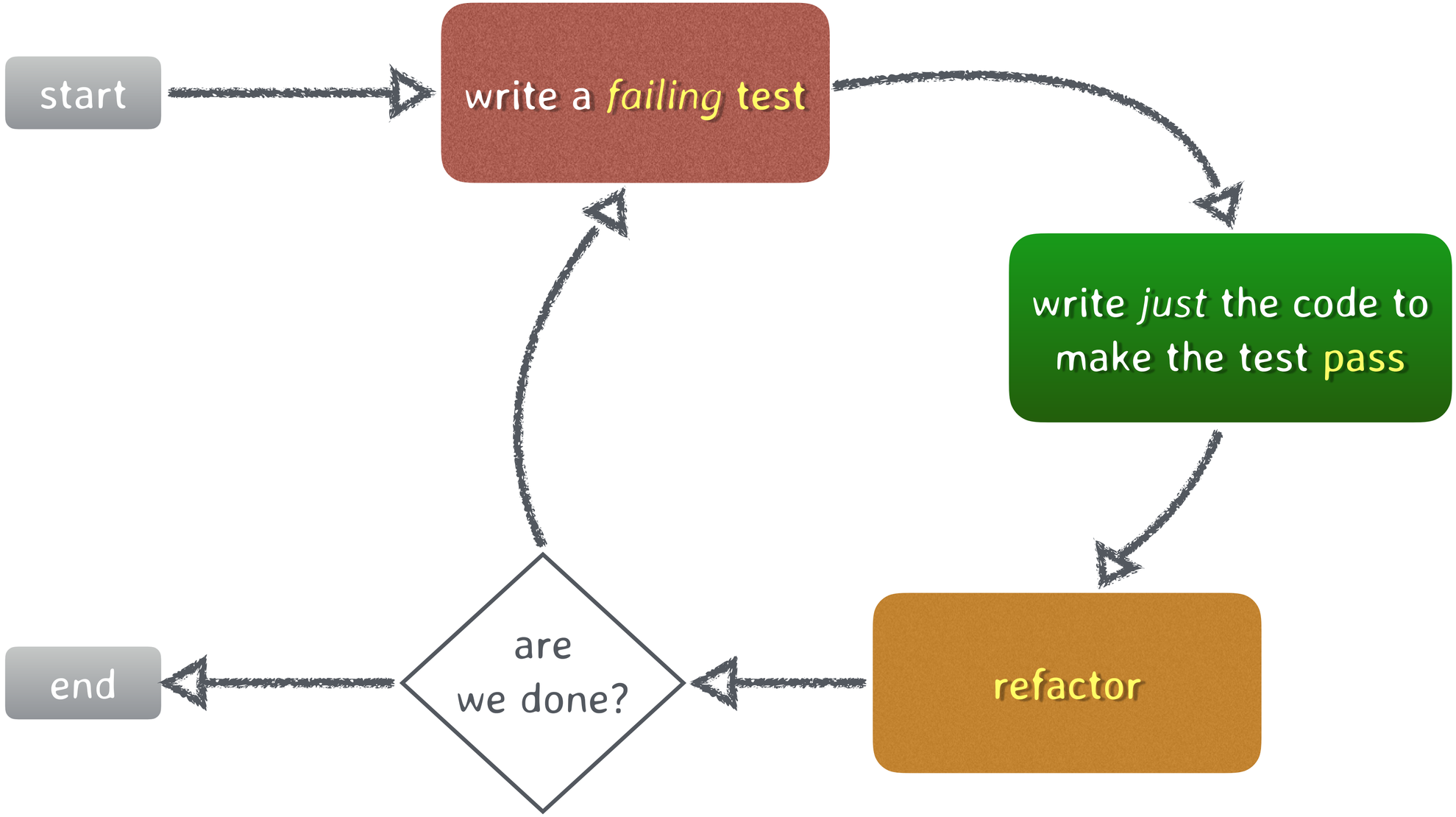
"Software bugs are costing the U.S. economy an estimated $59.5 billion each year"

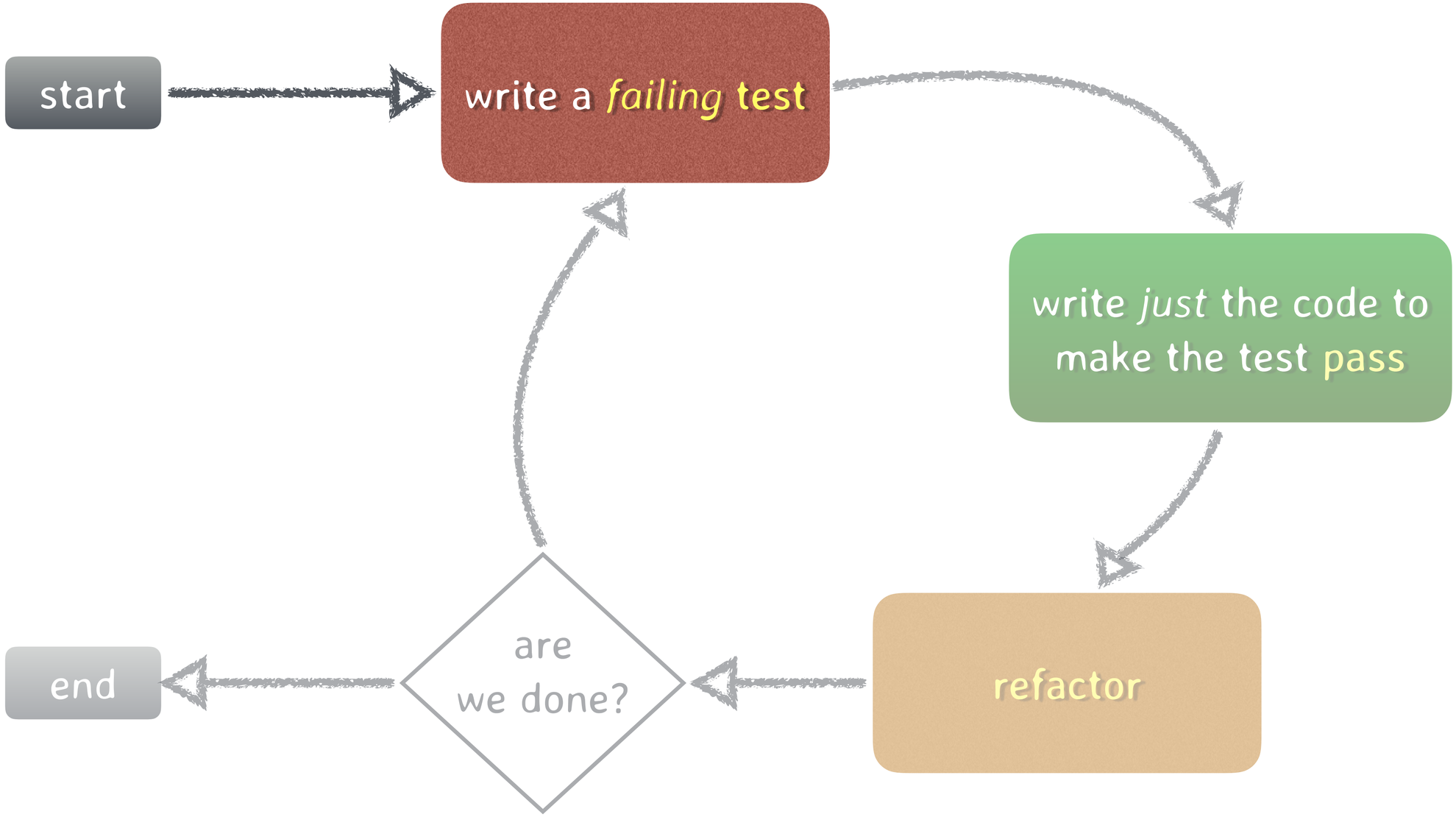
"Improvements in testing could reduce this cost by about a third, or $22.5 billion

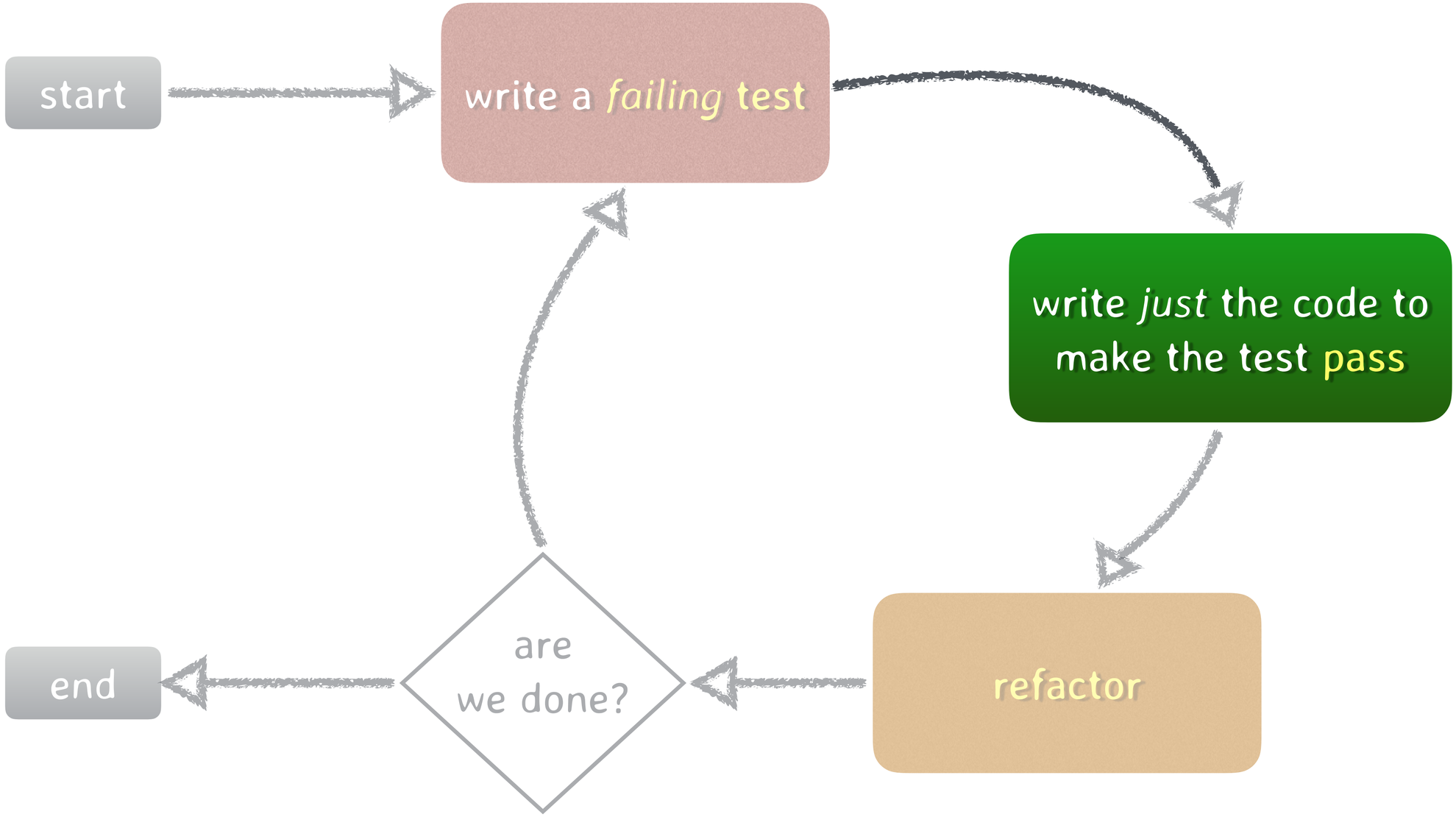
"The general consensus seems to be [that] the *current state of the art with respect to testing* is poor and can be sufficiently improved"

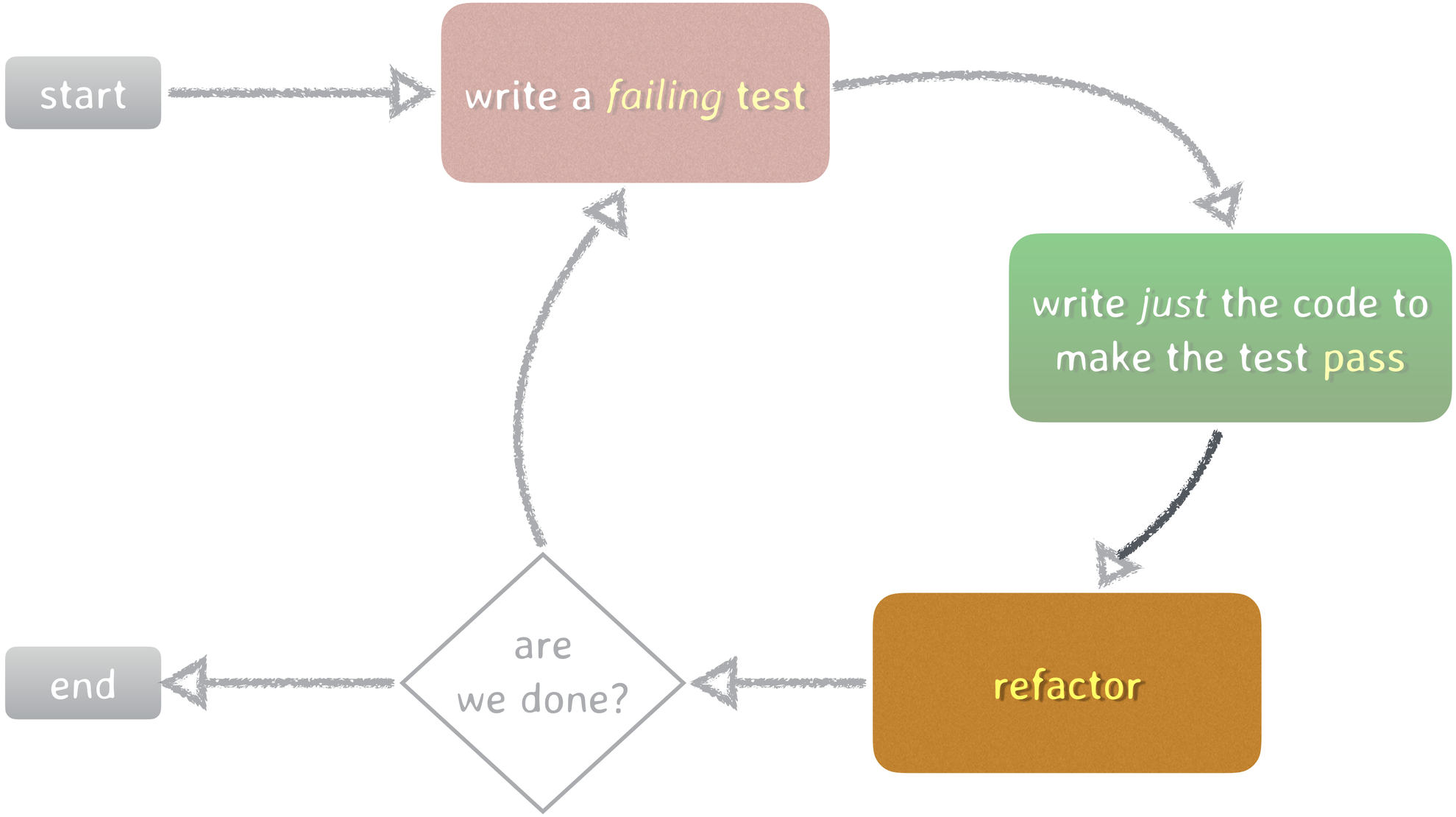
TDD IS HARD
LET'S GO SHOPPING (AND WRACK UP TECHNICAL DEBT)
imgflip.com

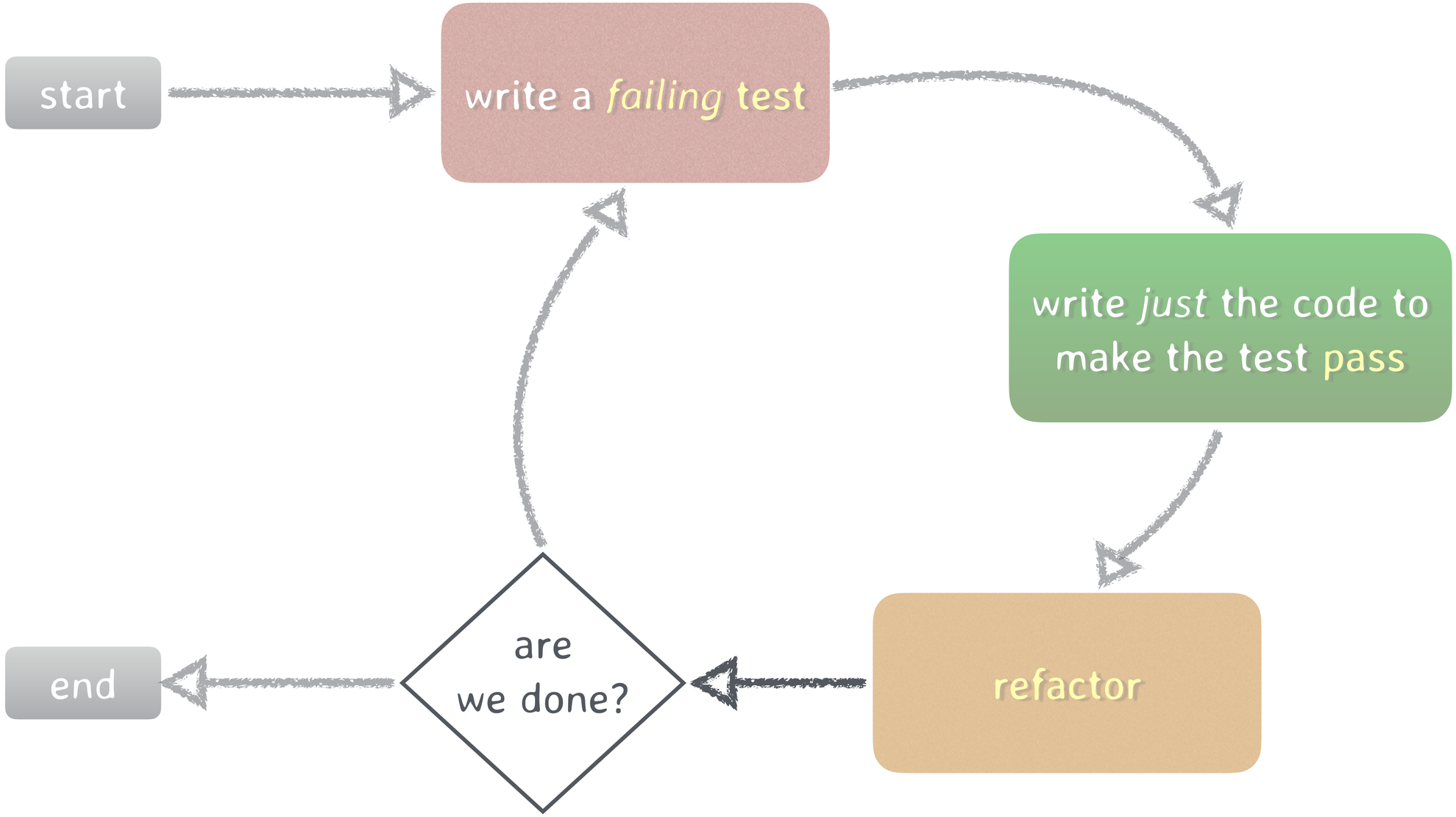
The TDD Cycle

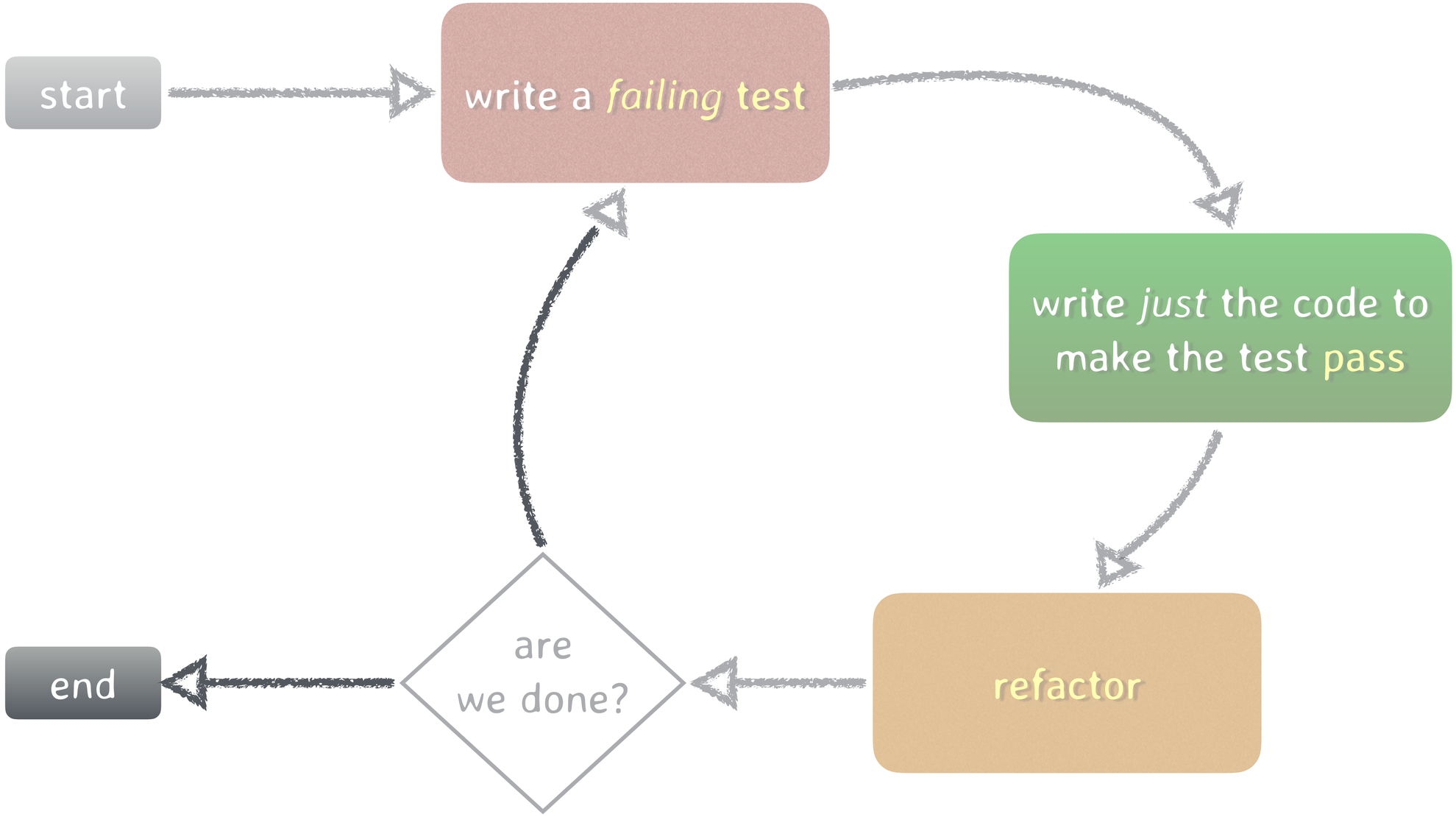

start
write a failing test
write just the code to make the test pass
refactor
are we done?
end

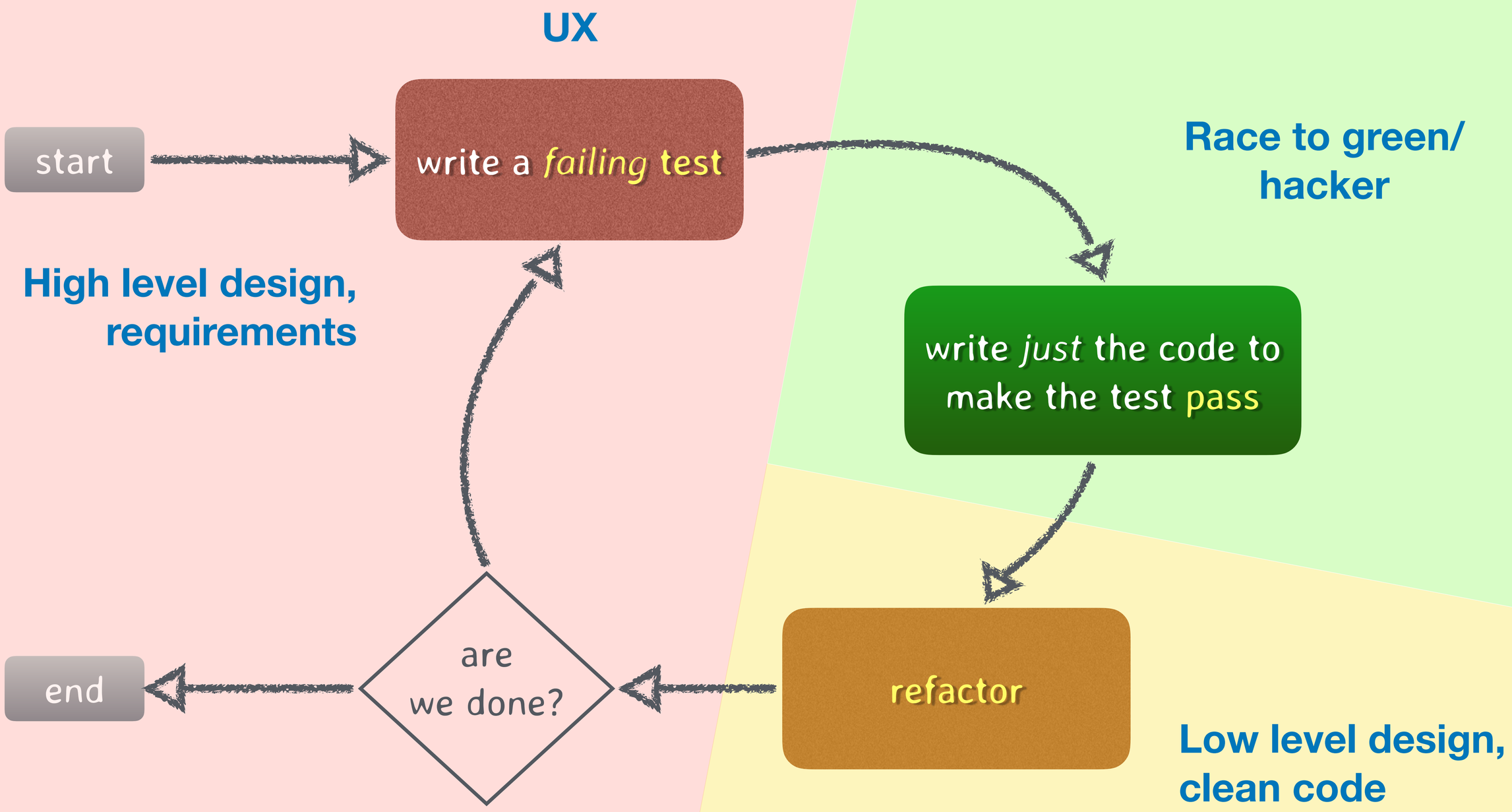

start
write a failing test
write just the code to make the test pass
refactor
are we done?
end


start
write a failing test
write just the code to make the test pass
refactor
are we done?
end


start
write a failing test
write just the code to make the test pass
refactor
are we done?
end


start
write a failing test
write just the code to make the test pass
refactor
are we done?
end


start
write a failing test
write just the code to make the test pass
refactor
are we done?
end


UX
start
write a failing test
Race to green/
hacker
High level design,
requirements
write just the code to
make the test pass
are
we done?
end
refactor
Low level design,
clean code

Why is
TDD hard?

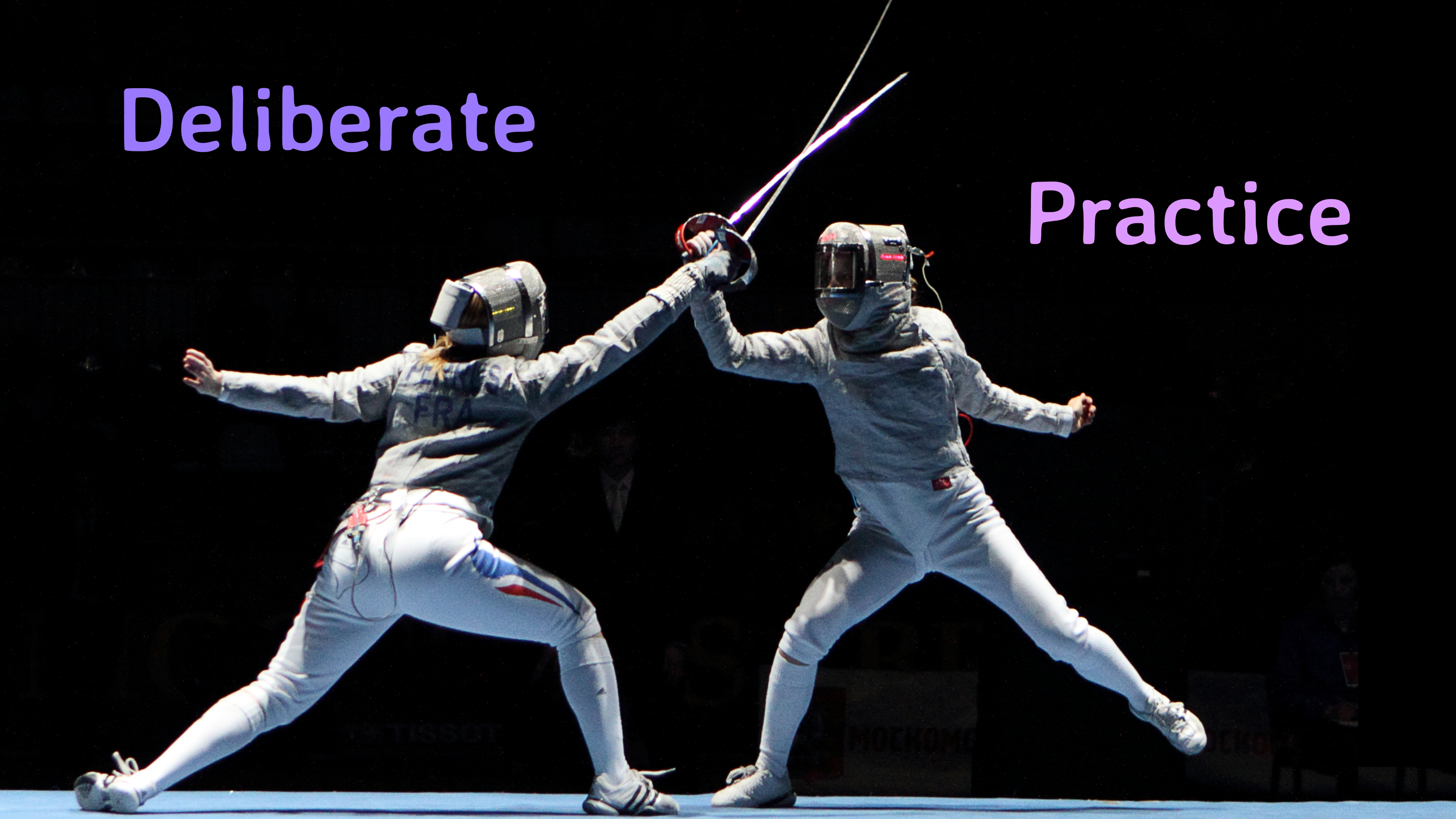
The trouble with TDD

The trouble with TDD

Hard to learn the *art*
- requires experience

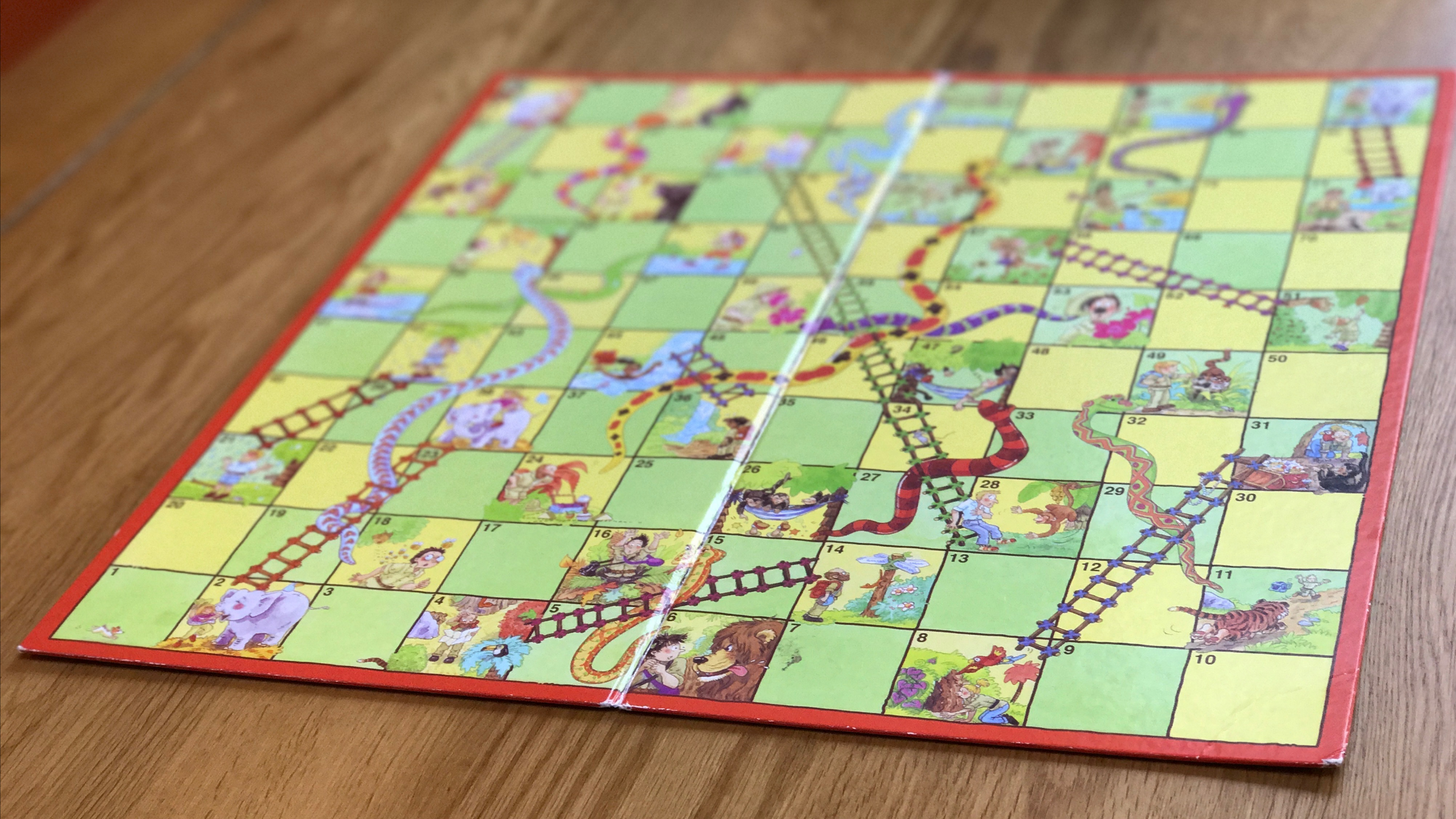
Productivity goes down before it goes up

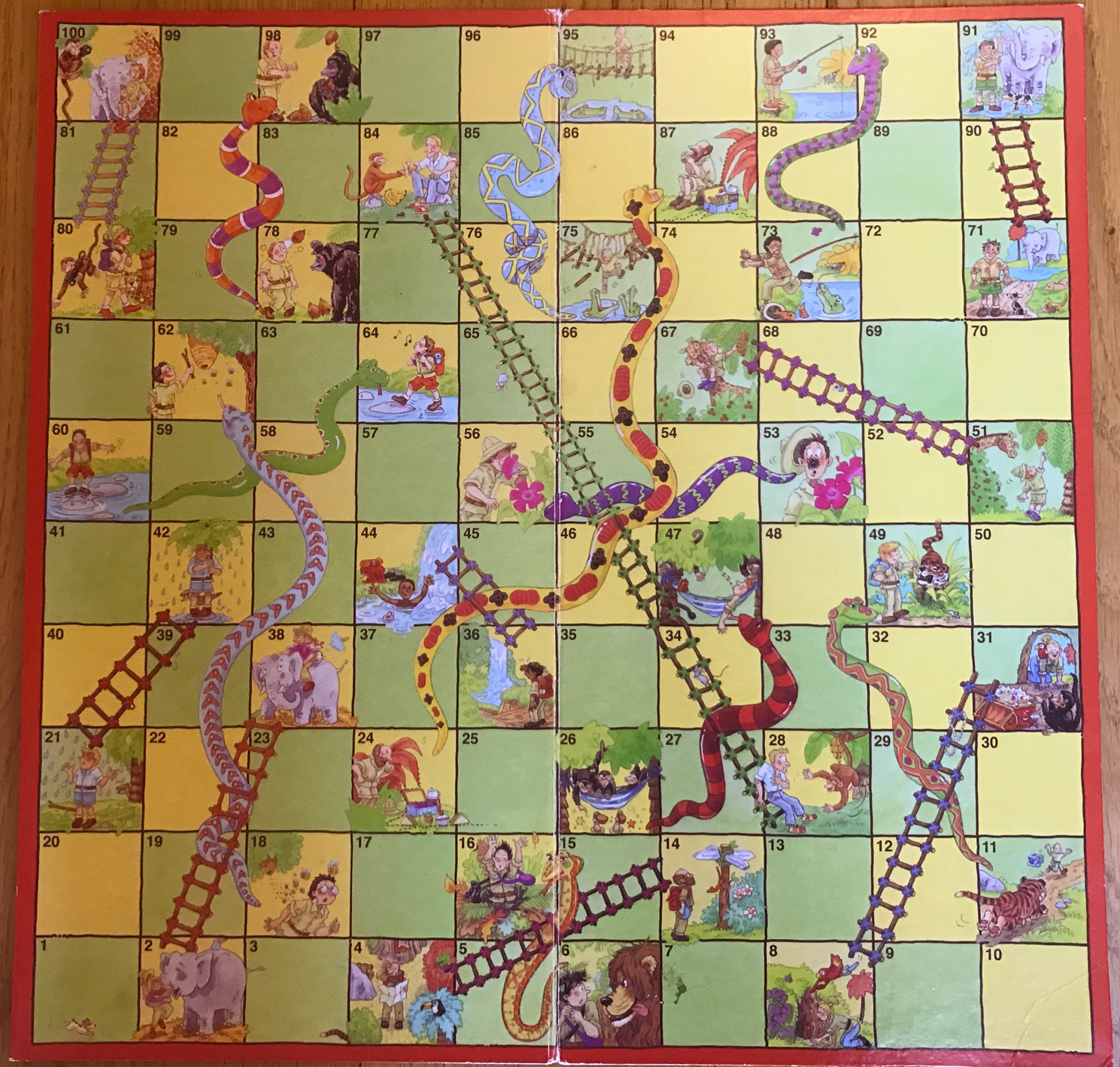
Practicing on the job

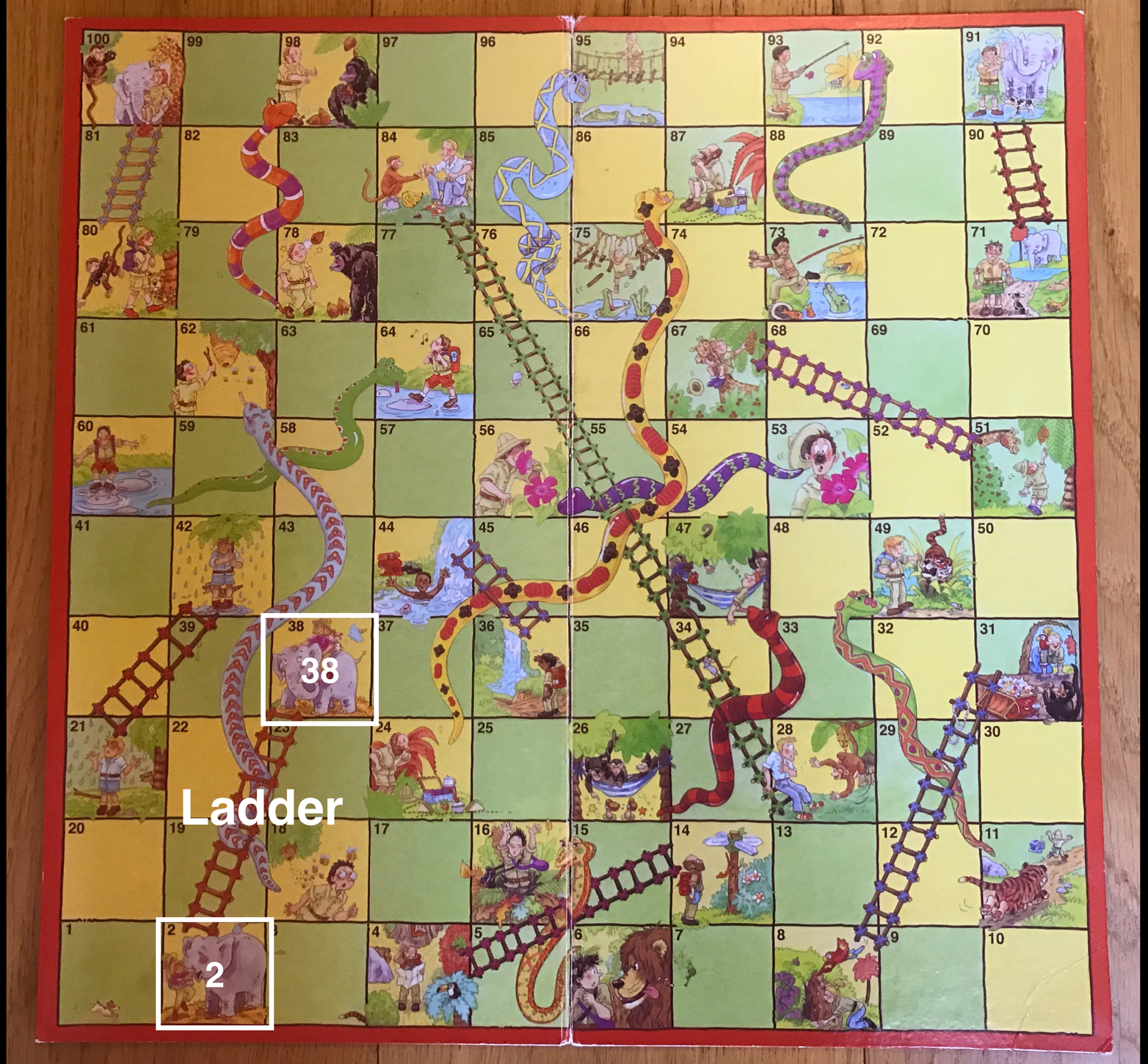

Deliberate
Practice

atdd.dev

snakes & ladders




100 99 98 97 96 95 94 93 92 91
81 82 83 84 85 86 87 88 89 90
80 79 78 77 76 75 74 73 72 71
61 62 63 64 65 66 67 68 69 70
60 59 58 57 56 55 54 53 52 51
41 42 43 44 45 46 47 48 49 50
40 39 38 37 36 35 34 33 32 31
21 22 23 24 25 26 27 28 29 30
20 19 18 17 16 15 14 13 12 11
1 2 3 4 5 6 7 8 9 10


38
Ladder
2

2 Ladder 38

2 Ladder 38

2	Ladder	38
4	Ladder	14

2	Ladder	38
4	Ladder	14
8	Ladder	31
21	Ladder	42
28	Ladder	84
36	Ladder	44
47	Snake	26
49	Snake	11
51	Ladder	67
56	Snake	53
62	Snake	18
64	Snake	60
71	Ladder	91
80	Ladder	100
87	Snake	24
93	Snake	73
95	Snake	75
98	Snake	78

```
{ 2, { Portal::Type::Ladder, 38 } },
{ 4, { Portal::Type::Ladder, 14 } },
{ 8, { Portal::Type::Ladder, 31 } },
{ 21, { Portal::Type::Ladder, 42 } },
{ 28, { Portal::Type::Ladder, 84 } },
{ 36, { Portal::Type::Ladder, 44 } },
{ 47, { Portal::Type::Snake, 26 } },
{ 49, { Portal::Type::Snake, 11 } },
{ 51, { Portal::Type::Ladder, 67 } },
{ 56, { Portal::Type::Snake, 53 } },
{ 62, { Portal::Type::Snake, 18 } },
{ 64, { Portal::Type::Snake, 60 } },
{ 71, { Portal::Type::Ladder, 91 } },
{ 80, { Portal::Type::Ladder, 100 } },
{ 87, { Portal::Type::Snake, 24 } },
{ 93, { Portal::Type::Snake, 73 } },
{ 95, { Portal::Type::Snake, 75 } },
{ 98, { Portal::Type::Snake, 78 } }
```

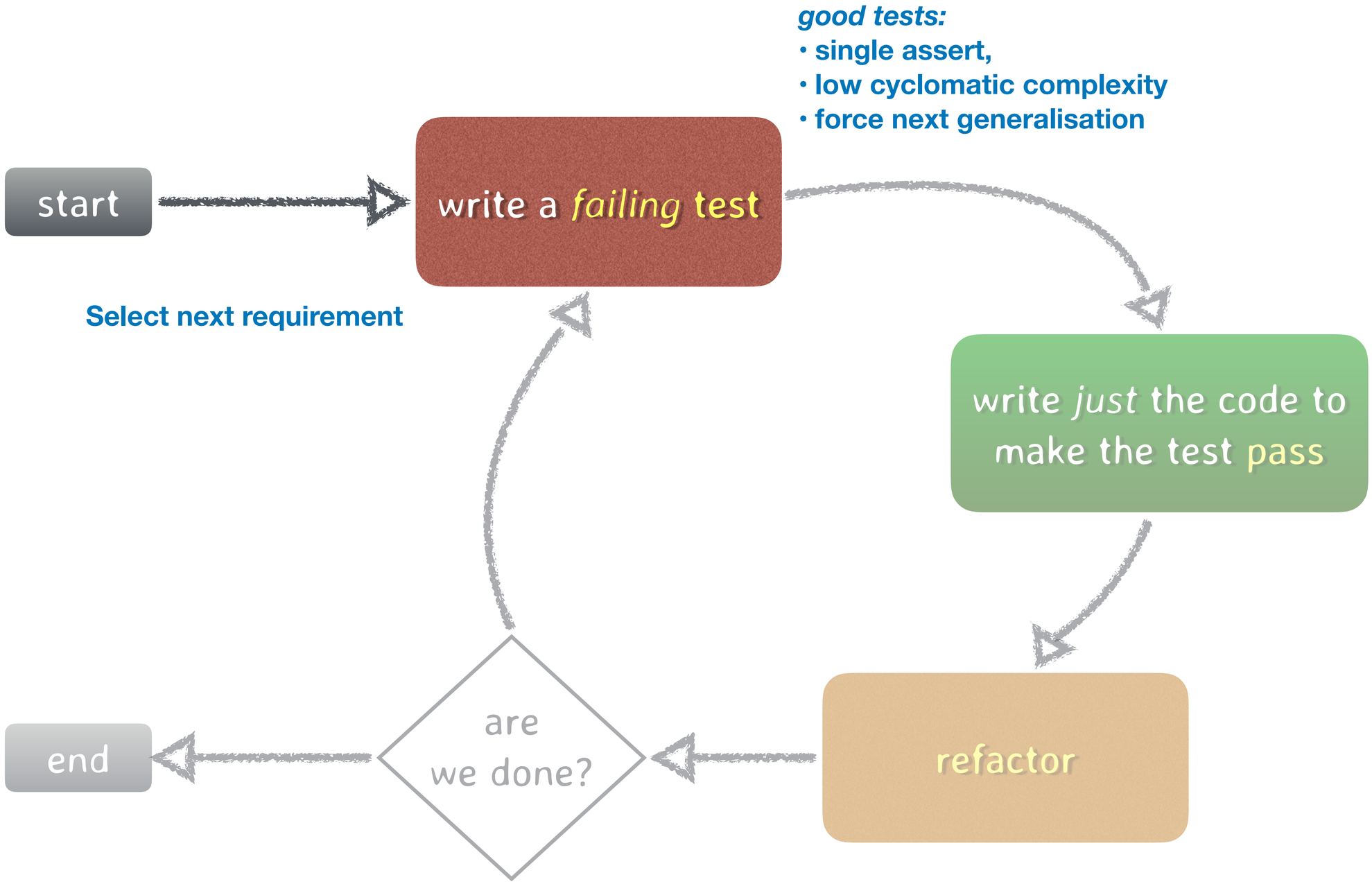
```
std::map<uint32_t, Portal> portals =
    {
        { 2, { Portal::Type::Ladder, 38 } },
        { 4, { Portal::Type::Ladder, 14 } },
        { 8, { Portal::Type::Ladder, 31 } },
        { 21, { Portal::Type::Ladder, 42 } },
        { 28, { Portal::Type::Ladder, 84 } },
        { 36, { Portal::Type::Ladder, 44 } },
        { 47, { Portal::Type::Snake, 26 } },
        { 49, { Portal::Type::Snake, 11 } },
        { 51, { Portal::Type::Ladder, 67 } },
        { 56, { Portal::Type::Snake, 53 } },
        { 62, { Portal::Type::Snake, 18 } },
        { 64, { Portal::Type::Snake, 60 } },
        { 71, { Portal::Type::Ladder, 91 } },
        { 80, { Portal::Type::Ladder, 100 } },
        { 87, { Portal::Type::Snake, 24 } },
        { 93, { Portal::Type::Snake, 73 } },
        { 95, { Portal::Type::Snake, 75 } },
        { 98, { Portal::Type::Snake, 78 } }
    };
```

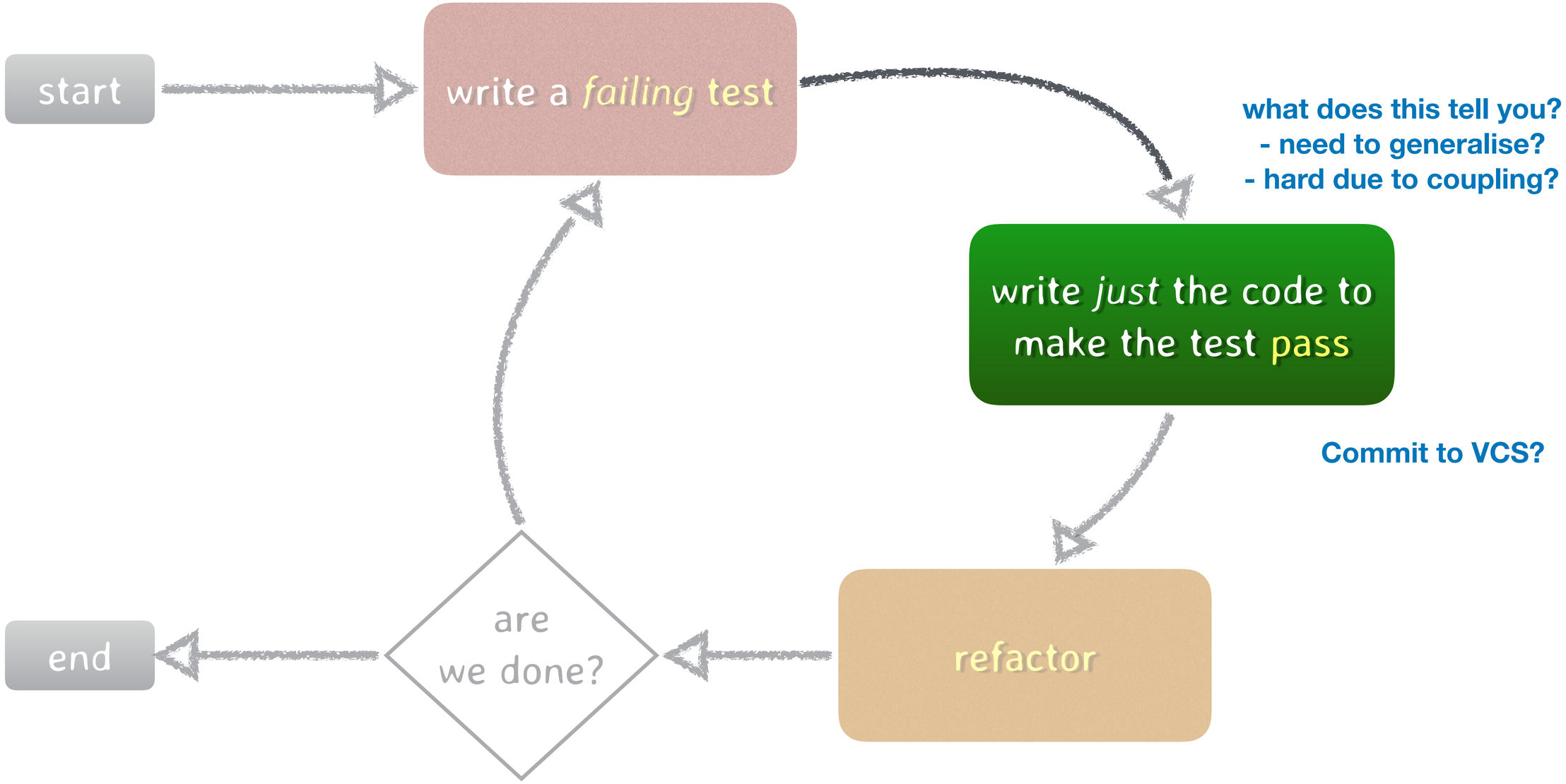
```
struct Portal {
    enum class Type { Snake, Ladder };
    Type type;
    uint32_t target;
};

std::map<uint32_t, Portal> portals =
    {
        { 2, { Portal::Type::Ladder, 38 } },
        { 4, { Portal::Type::Ladder, 14 } },
        { 8, { Portal::Type::Ladder, 31 } },
        { 21, { Portal::Type::Ladder, 42 } },
        { 28, { Portal::Type::Ladder, 84 } },
        { 36, { Portal::Type::Ladder, 44 } },
        { 47, { Portal::Type::Snake, 26 } },
        { 49, { Portal::Type::Snake, 11 } },
        { 51, { Portal::Type::Ladder, 67 } },
        { 56, { Portal::Type::Snake, 53 } },
        { 62, { Portal::Type::Snake, 18 } },
        { 64, { Portal::Type::Snake, 60 } },
        { 71, { Portal::Type::Ladder, 91 } },
        { 80, { Portal::Type::Ladder, 100 } },
```

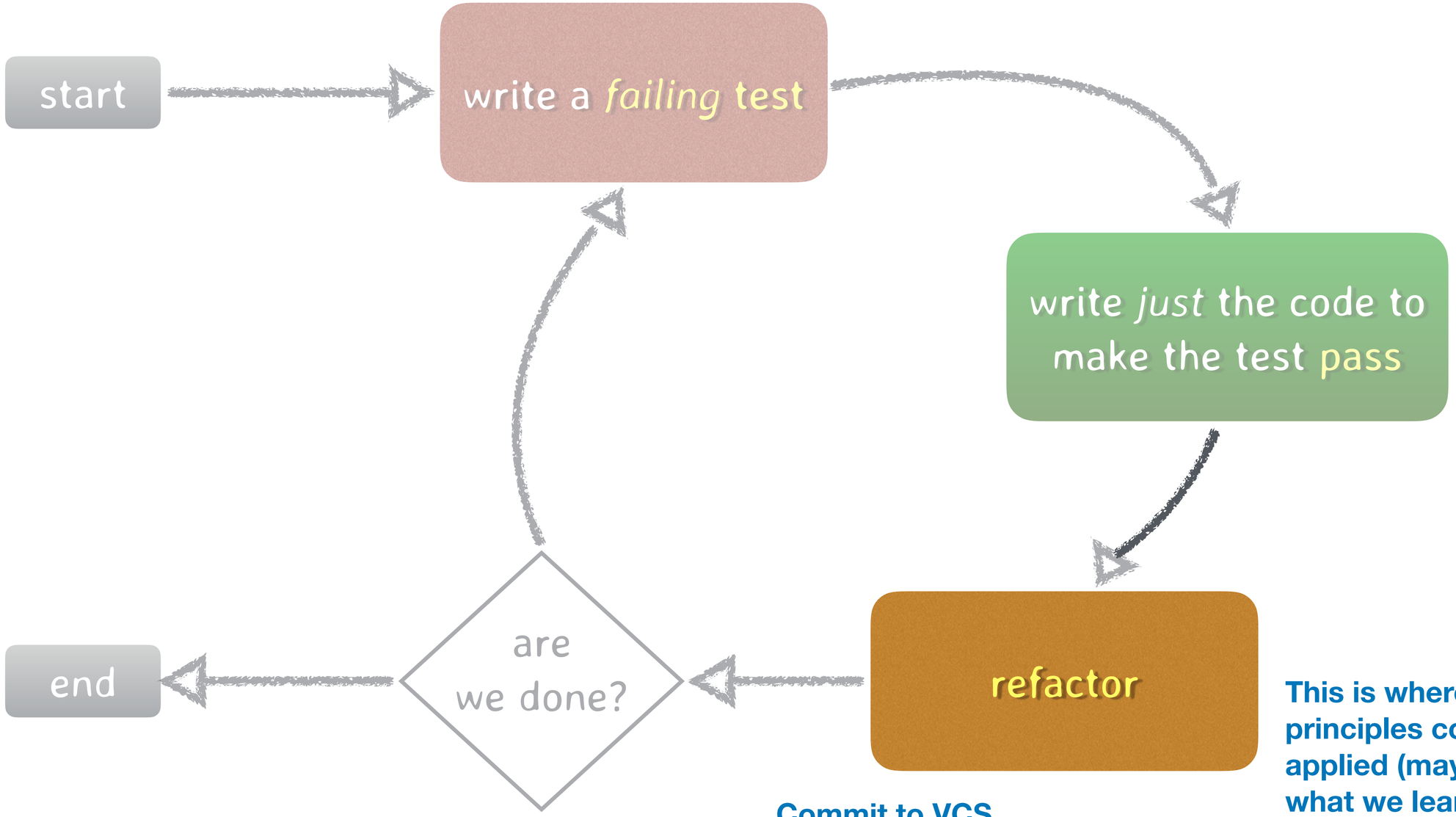
Walk through

The TDD Cycle

pt II

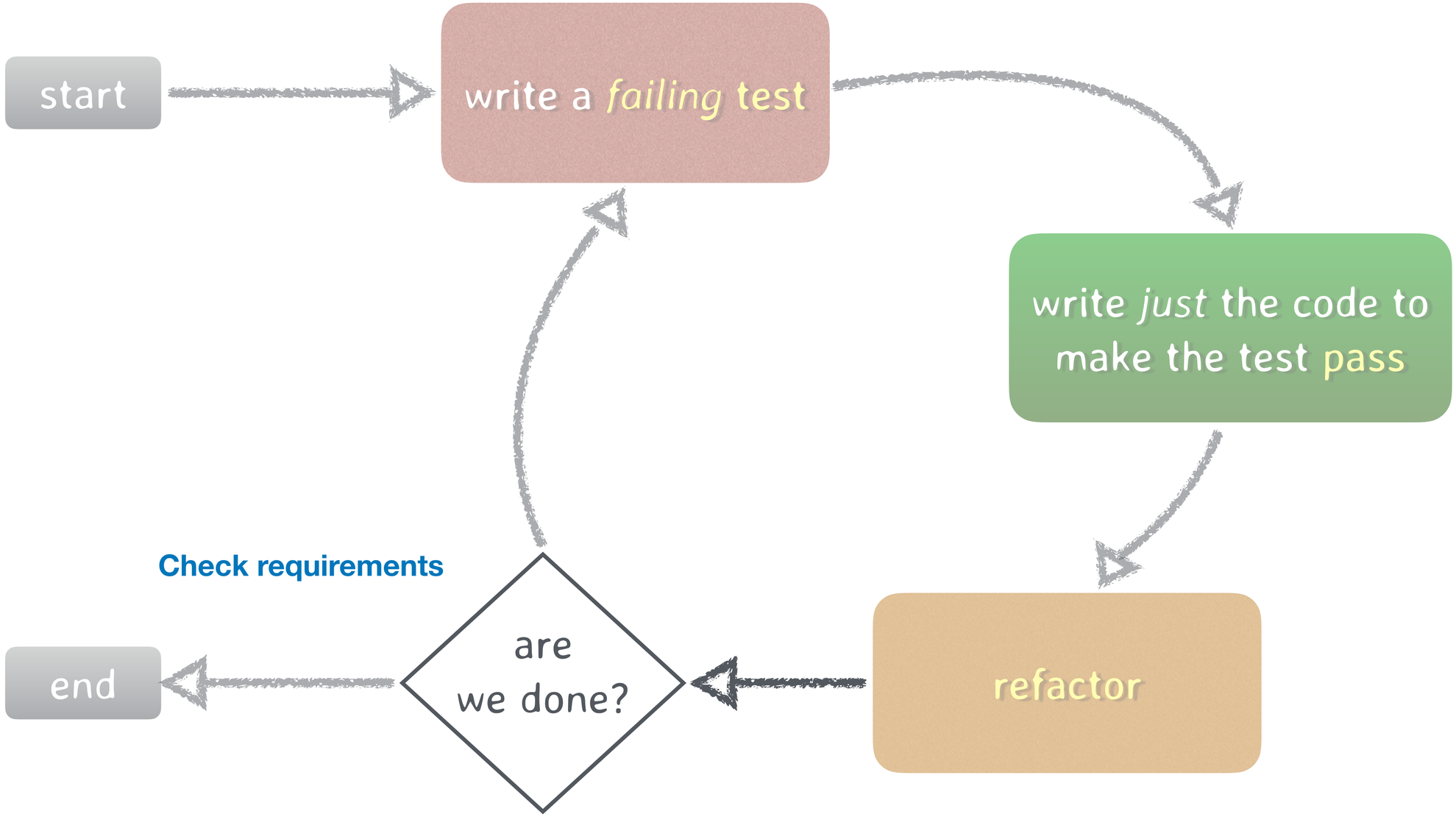

good tests:
• single assert,
• low cyclomatic complexity
• force next generalisation
start
write a failing test
Select next requirement
write just the code to make the test pass
refactor
are we done?
end

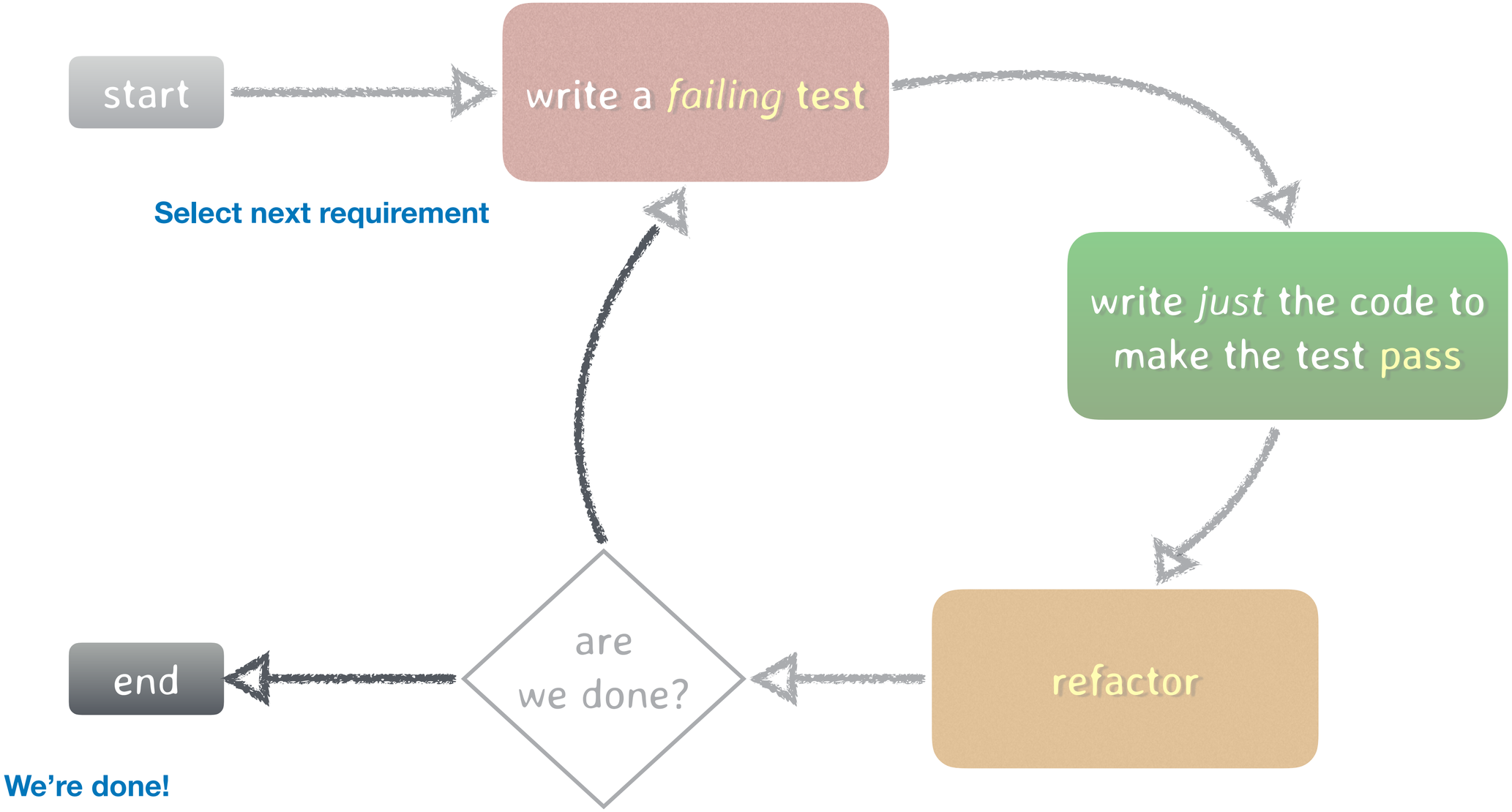

start
write a failing test
what does this tell you?
- need to generalise?
- hard due to coupling?
write just the code to make the test pass
Commit to VCS?
refactor
are we done?
end



Commit to VCS

This is where the design principles come in and are applied (maybe based on what we learnt at green)


start
write a failing test
write just the code to make the test pass
refactor
are we done?
Check requirements
end


start
write a failing test
Select next requirement
write just the code to make the test pass
refactor
are we done?
end
We're done!

HOROS

S
O
L
I
D

SOLID

The SOLID principles

Single responsibility
Open/ Closed
Liskov Substitution
Interface Segregation
Dependency Inversion

Single responsibility

Dependency Inversion

Seperation of Concerns

Dependency Inversion

Seperation of Concerns

Dependency Inversion

Seperation of Concerns

Program to an Interface

Functional

Loose

Unit Testable

Introspective

‘**D**empotent

The Cat Flap Principle

Who invented the cat flap?

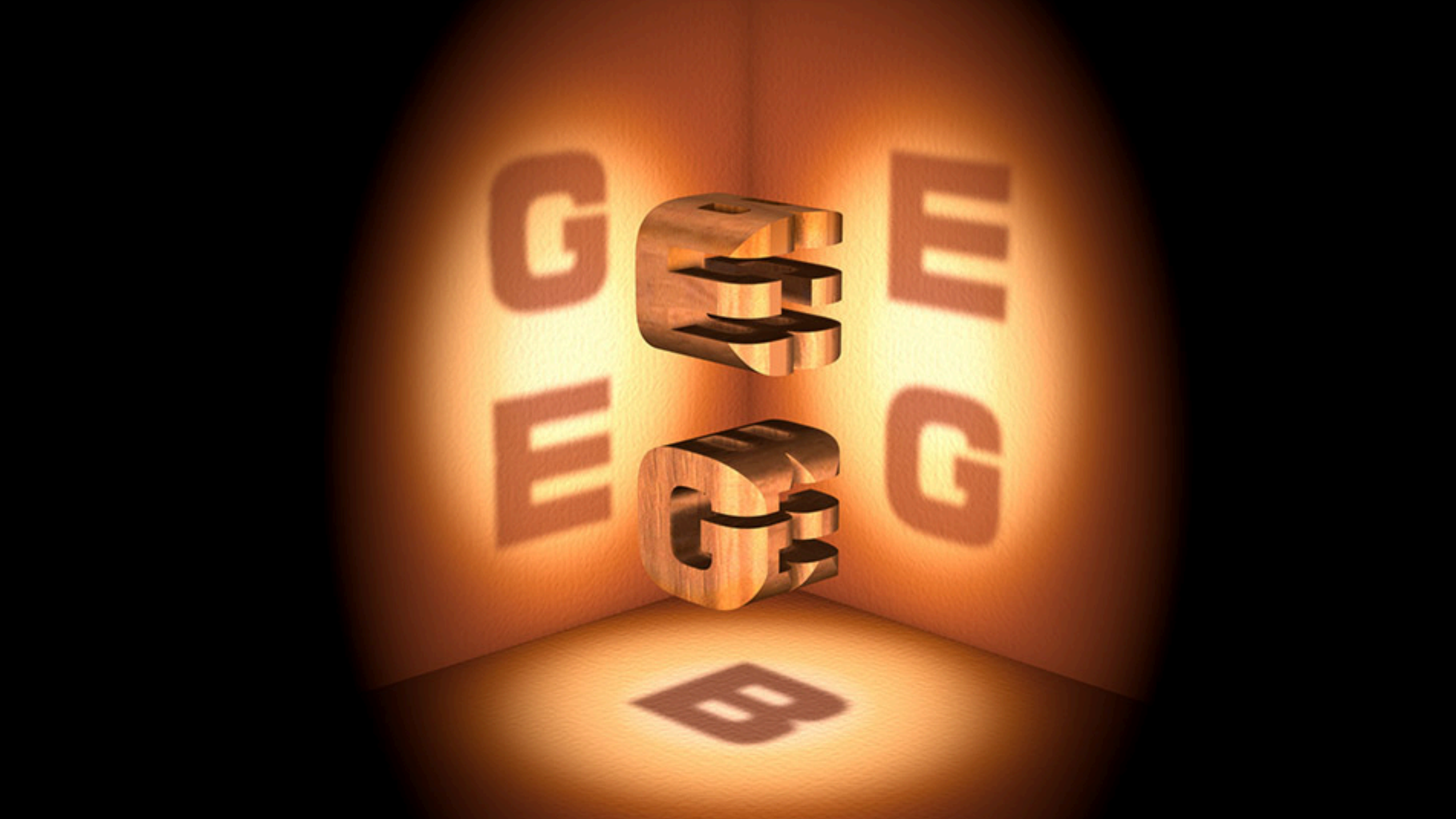
It is a rare mind indeed that can render the hitherto nonexistent blindingly obvious

- Dirk Gently

Everything should
be made as simple
as possible,
but no simpler

Einstein's Razor

You're Not as Smart as You Think You Are


G
E
E
G
B

It always takes longer than you expect, even when you take into account Hofstadter's Law

- Hofstadter's Law

You're not as smart as you think you are, even when you take into account Phil's Law

- Phil's Law

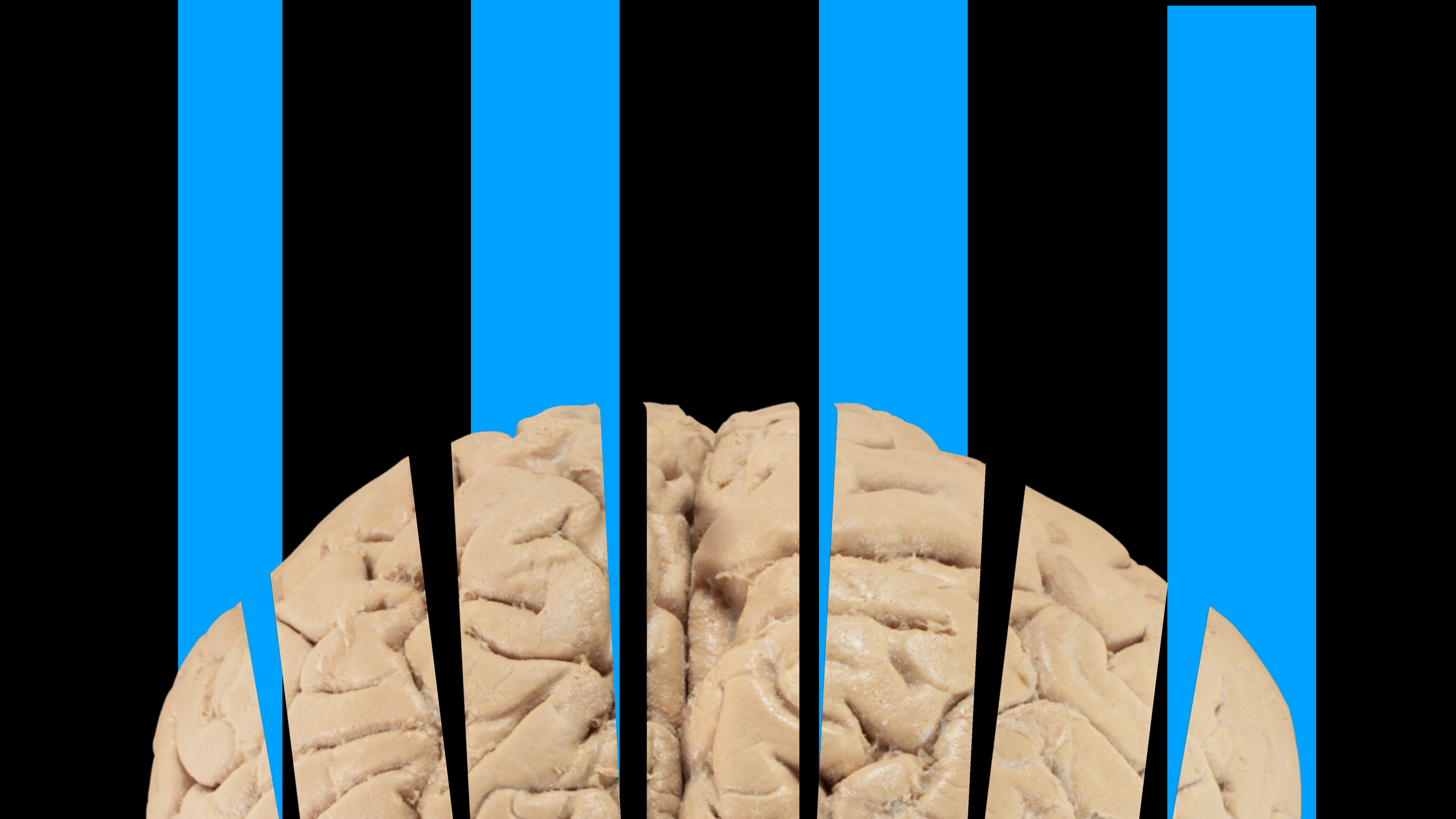
So how bad is it?

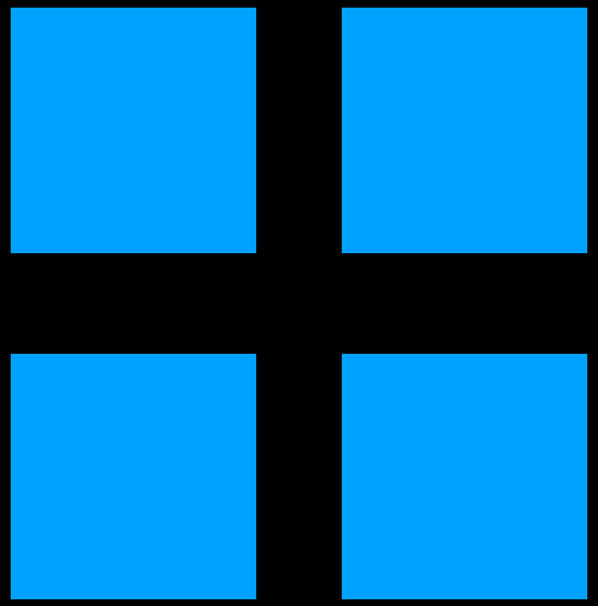
Magic numbers

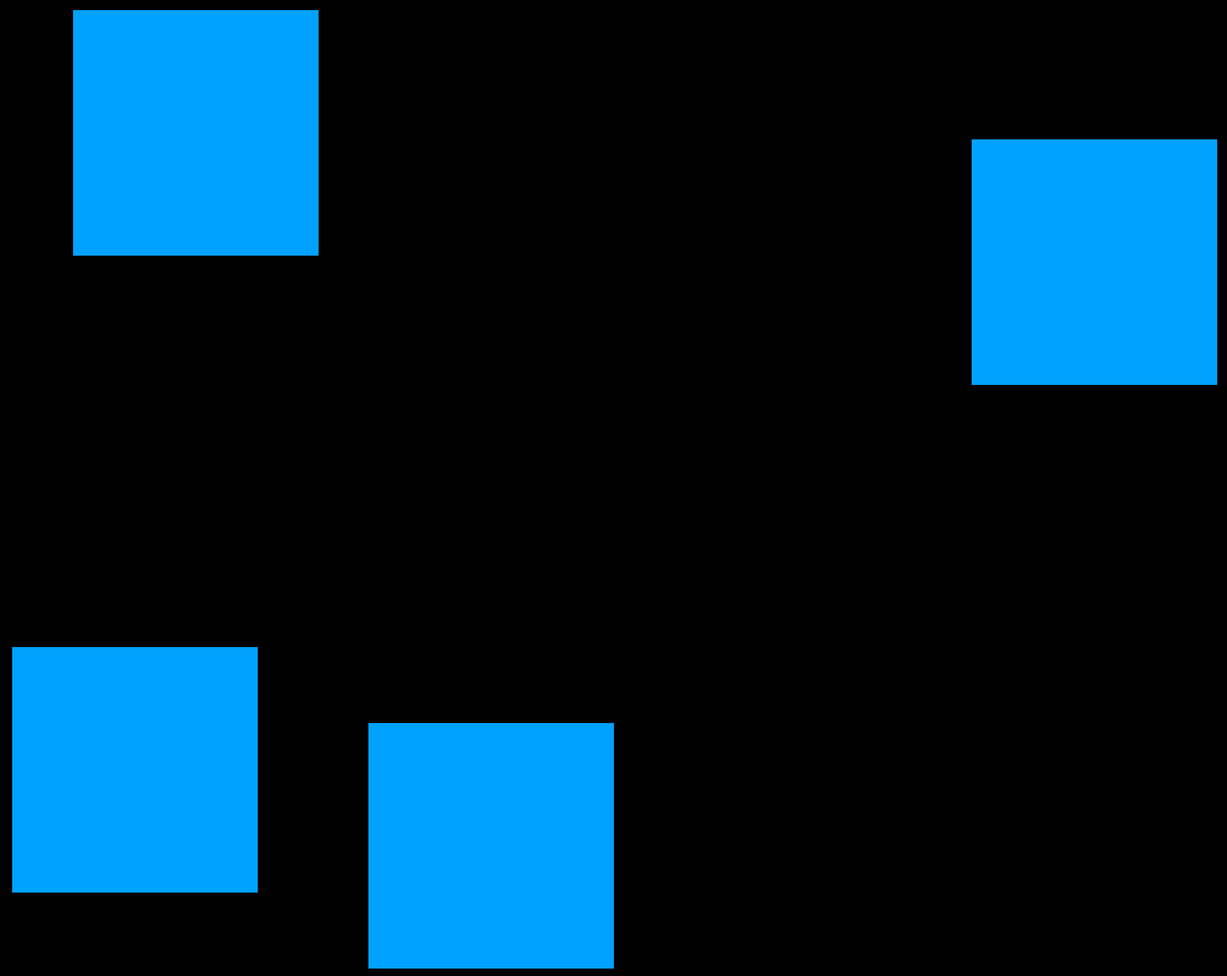
Magic number 7 (+/- 2)

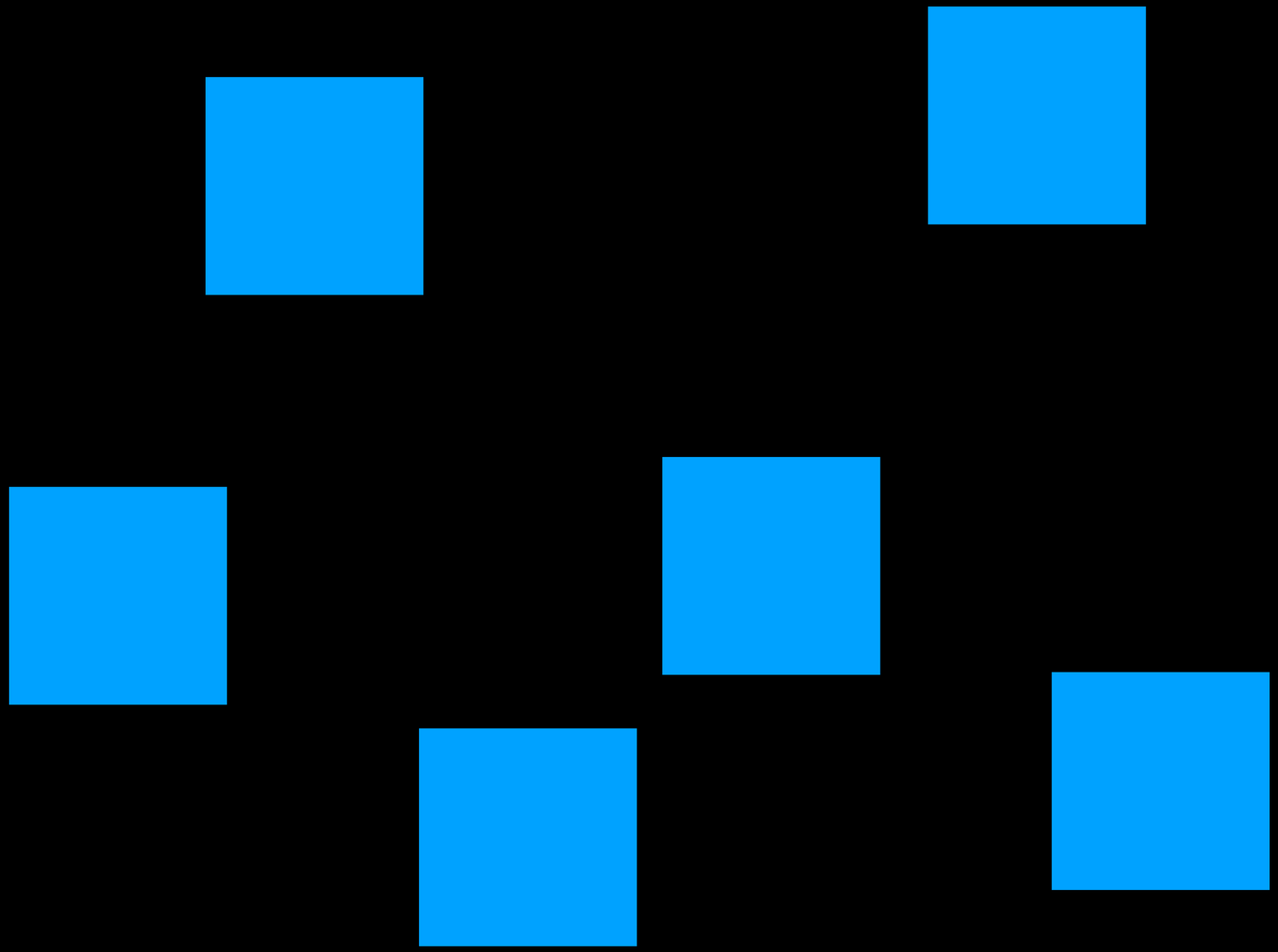
Magic number 7

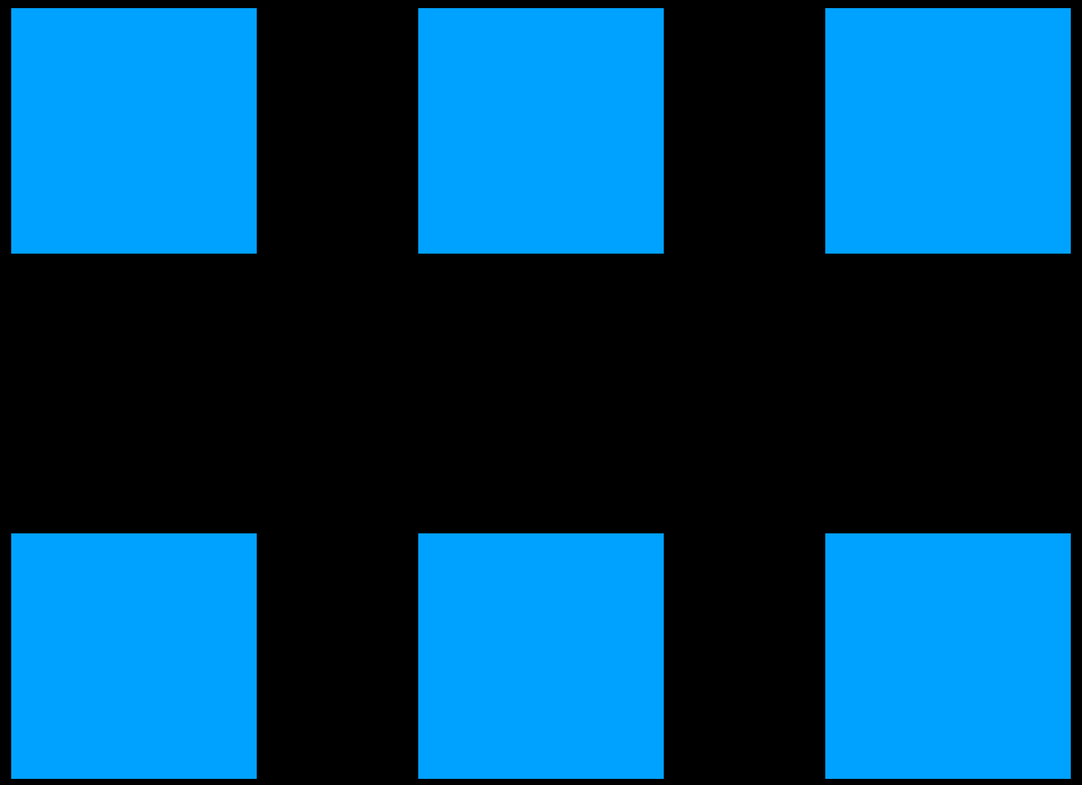
(+/- 2)

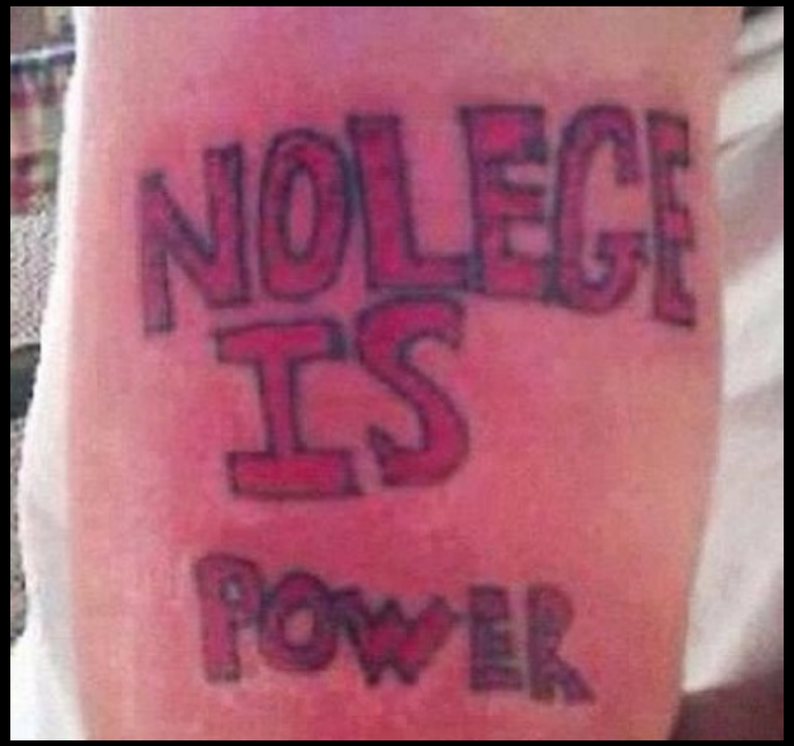











NOLEGE
IS
POWER

NO REGRETS

me
marry
you
will
Brittany

plex

complex

complex

many folds/
braids

simplex

one fold/ braid

complex

many folds/
braids

simple

one fold/ braid

complex

many folds/
braids

Fractal TDD

atdd.dev