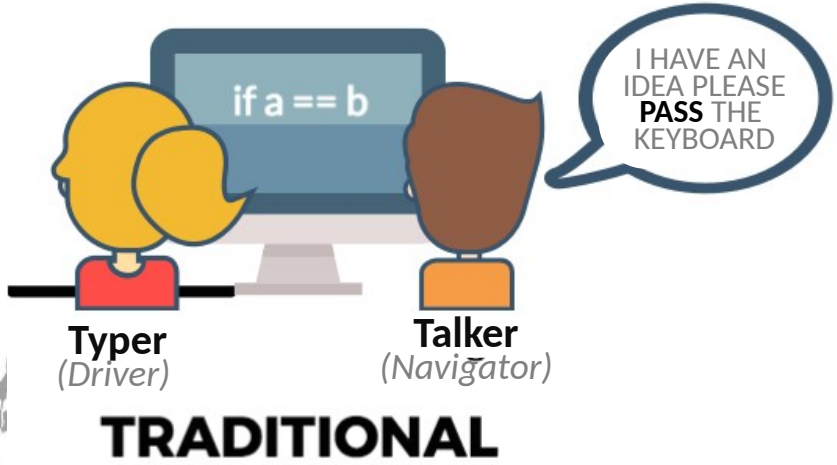
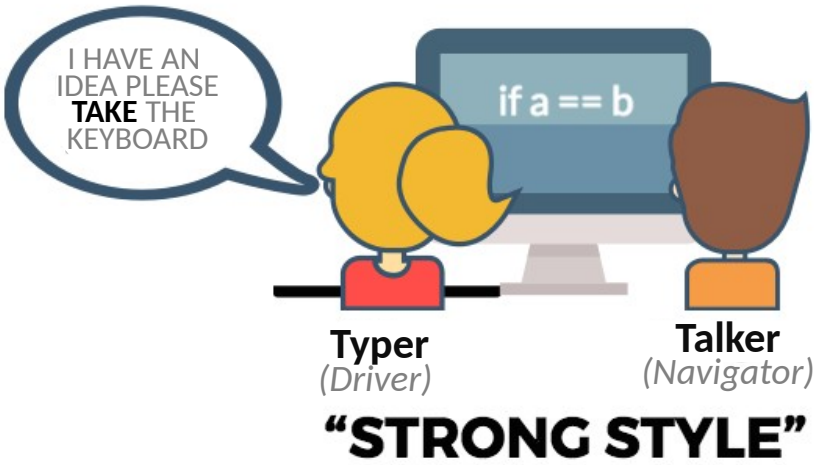




Pair Programming

" **Mob Programming** is all the brilliant people working on the **same thing**, at the **same time**, in the **same space**, on the **same computer**."

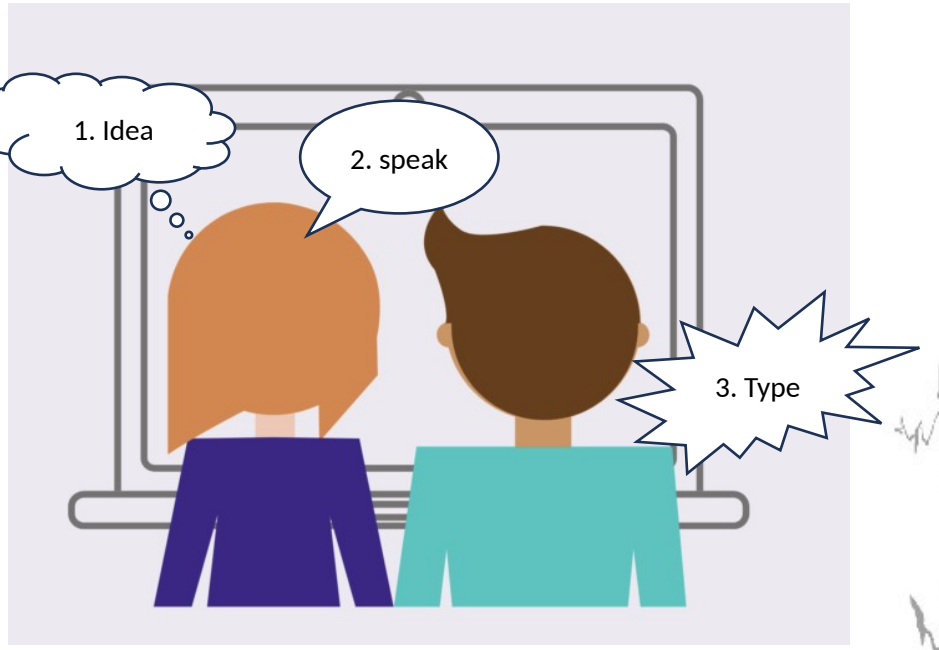
-- Woody Zuill

THE MOB PROGRAMMING GUIDEBOOK

Llewellyn Falco & Maaret Pyhäjärvi

art by Cary Polkovitz

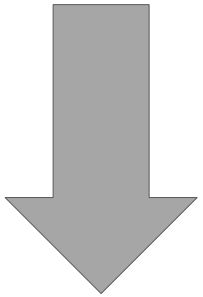
"For an idea to go from
your head to the computer
it must go through
someone else's hands"



Strong Style
Pair Programming

Speak at the highest level of abstraction

go down as needed

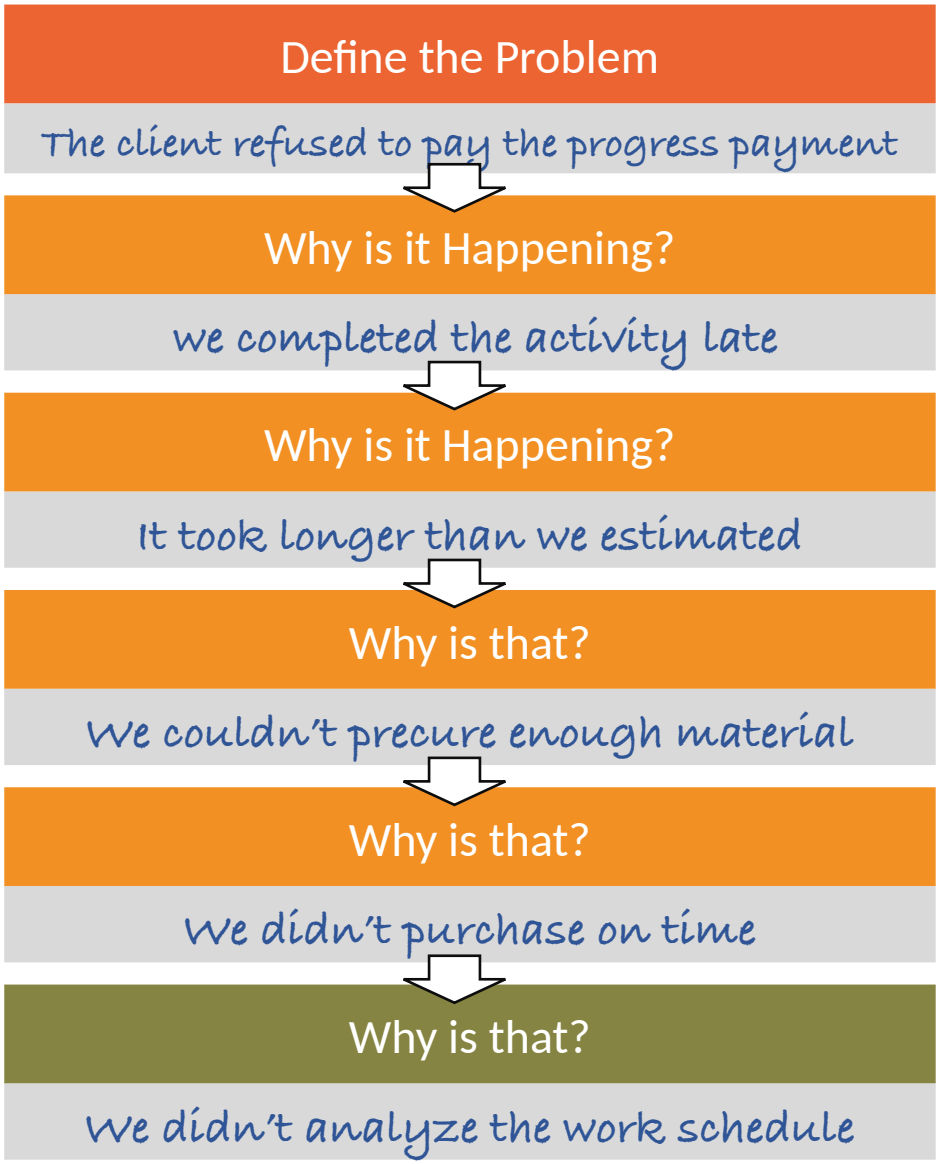


Intention – "write a for loop"

Location – "at line 34 and ½ "

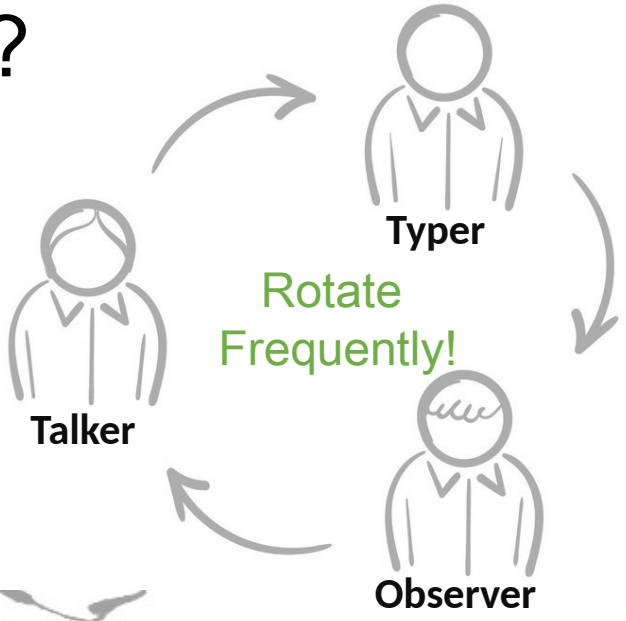
Details – "for(int i = 0; …."

The 5 Whys



Root Cause

When is the last time you rotated ?



Rotation Styles

On Task

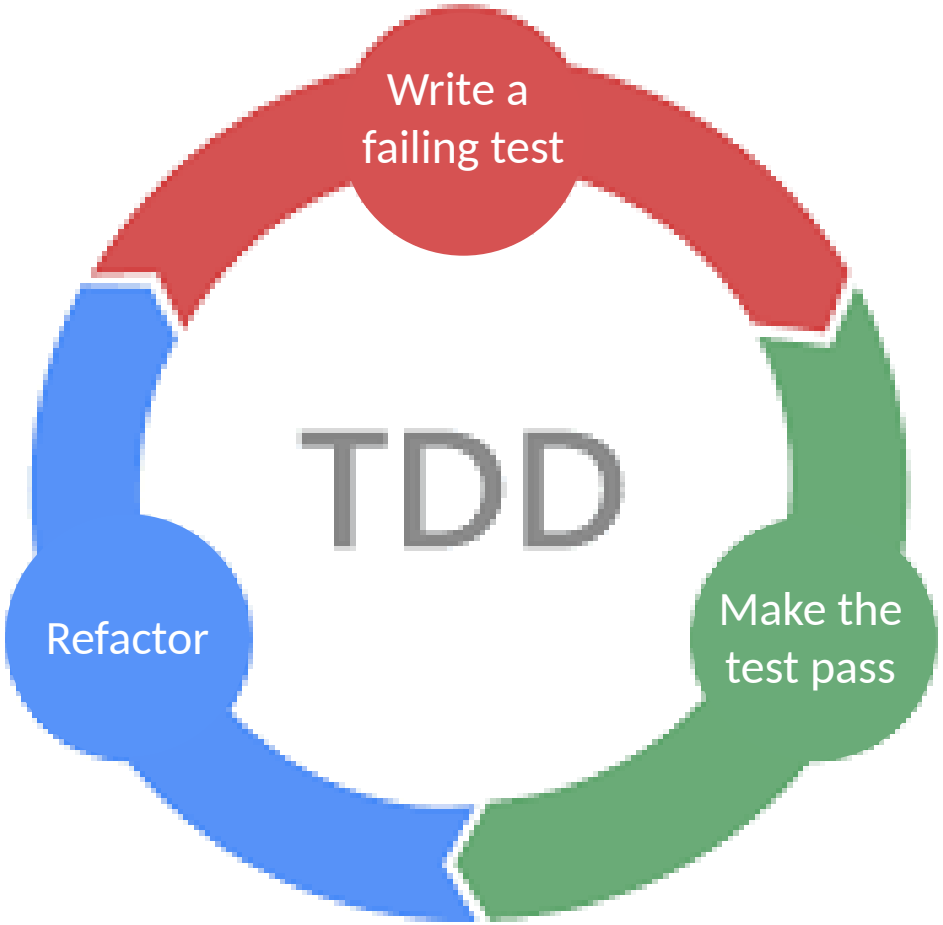
On Time

On Idea

Kindness
Consideration
and
Respect

Working Agreements

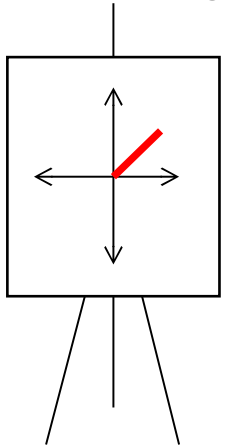
Make for better pairs & mobs


Write a
failing test
TDD
Make the
test pass
Refactor

Start Here!

with **drawing**

and natural **language**



// Create side (0,0) – (3,4)
// Verify length

Whiteboard

English

Testing Circle

Result

Code

Side (0,0) – (3,4) length = 5

```
Side s = new Side(0,0,3,4);
Approvals.Verify(s + " length = " +s.Length);
```

When was the last time you ran your tests?

1 minute?

3 minutes?

more?

Test && Commit || Revert (TCR)

Advanced Technique

What it is

Every time you run your tests, you will either:

1. Commit everything, because **all tests passed**
2. Revert everything *(or everything except the test)* because **it failed**

Kent Beck
Creator of TCR
& Extreme
Programming

Smaller steps
Faster Feedback
Better safety net

Java starter project

github.com/LarsEckart/tcr-extension.starterproject

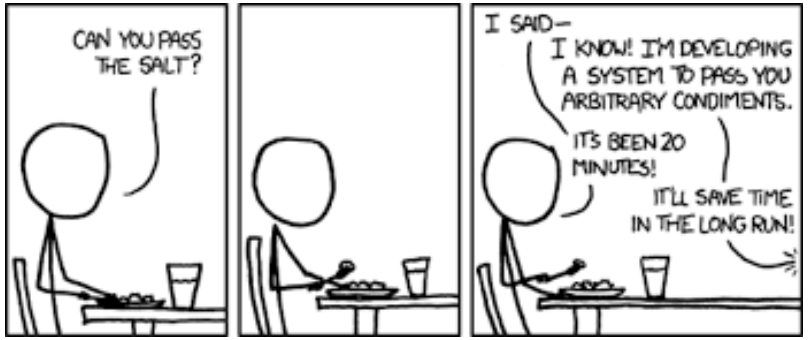
You

Ain't

Gonna

Need

It



xkcd.com

Write the simplest thing that could possibly work

Zero

One

Many

Boundaries

Interfaces

Exceptions

Scenarios

Guide your tests by
*looking at these **7** cases*

Blog

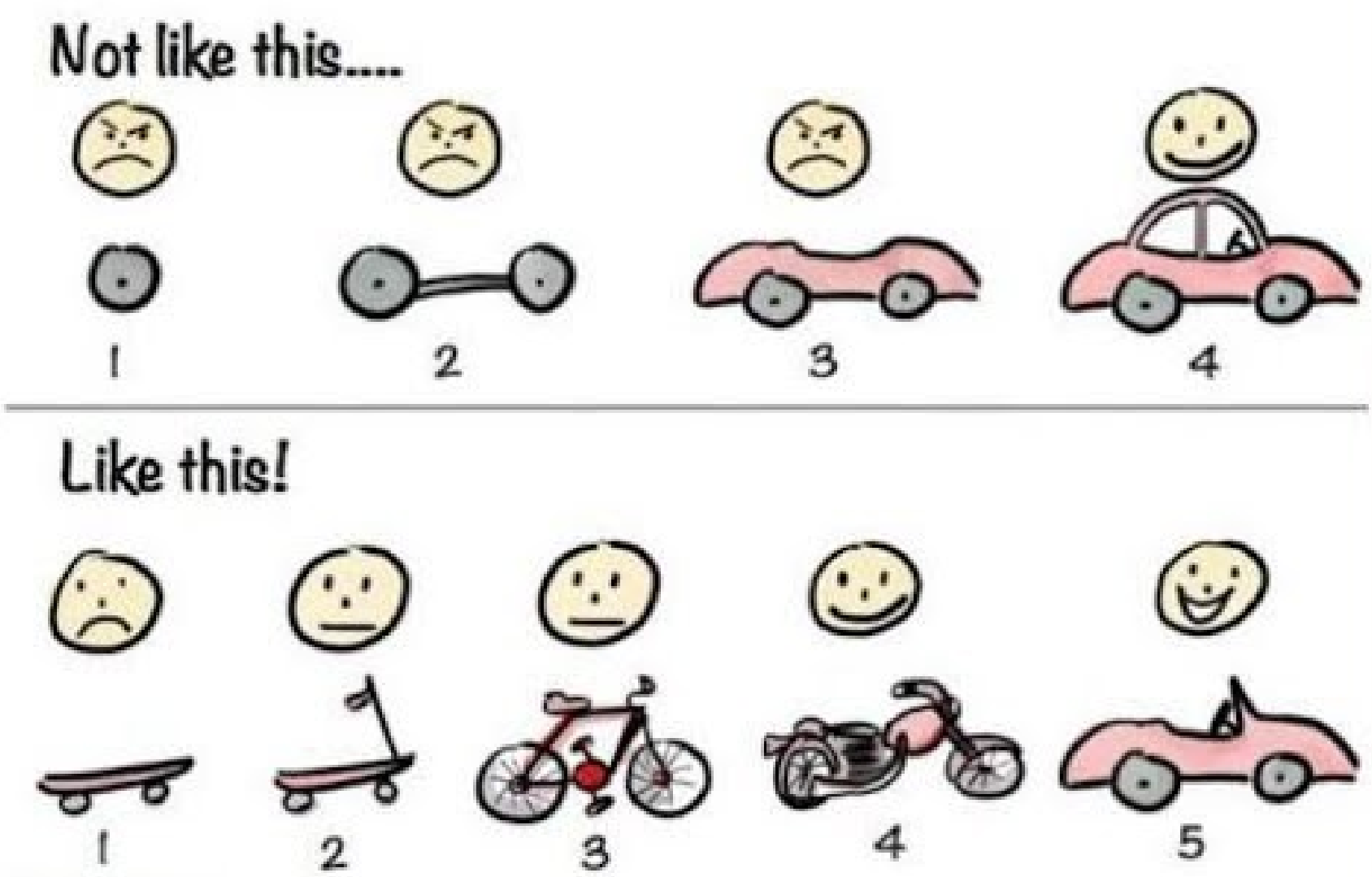
blog.wingman-sw.com/tdd-guided-by-zombies

Pattern by
James Grenning

1. Specification - what am I going to write?
2. Feedback - did it work?
3. Regression - does it still work?
4. Granularity - why did it stop working?

Blog

The 4 Benefits of Tests



Building an MVP

(minimal viable product)

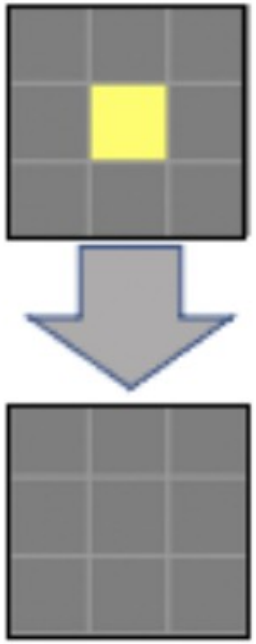
The 4 principles of

simple design:

(in order of importance)

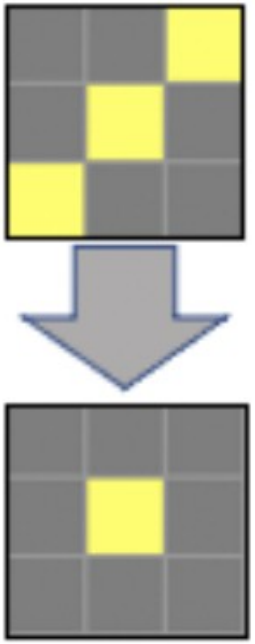
1. Passes all tests
2. Expresses the intent of programmer
3. Contains no duplication
4. Minimizes the number of classes and methods

1 neighbor or less



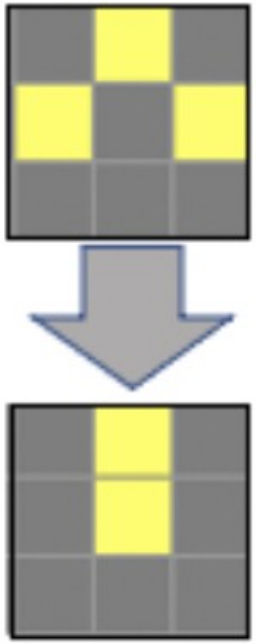
Middle square
dies
from **Starvation**

2–3 neighbors



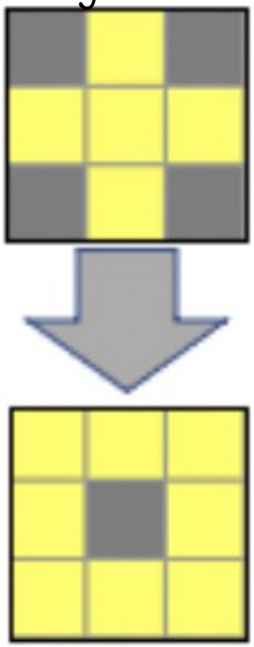
Middle square
survives
from **Subsistence**

3 neighbors



Middle square
is born

4 or more neighbors

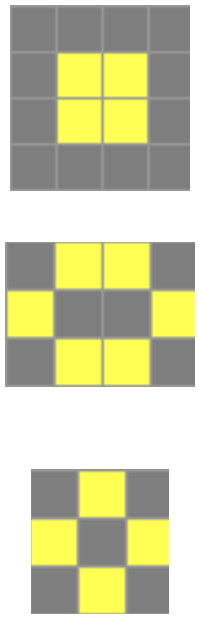


Middle square
dies
from **Overpopulation**

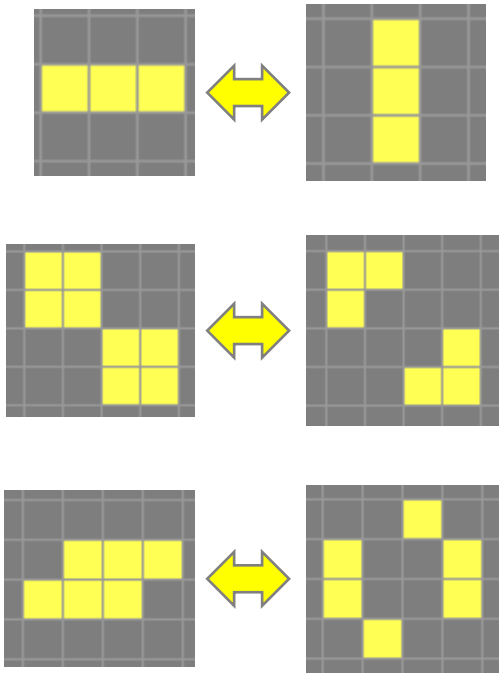
Game of Life Rules

Interesting Game of Life Structures

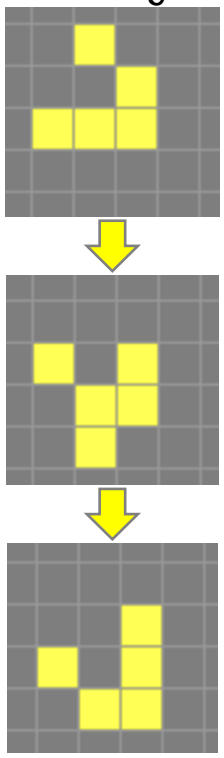
Unchanging



Oscillating



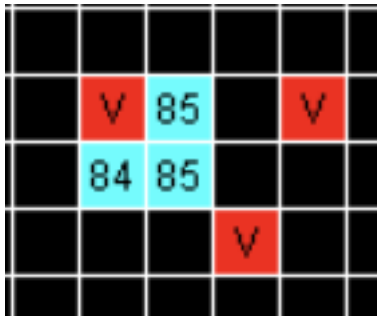
Traveling



https://playgameoflife.com

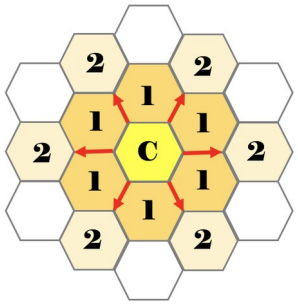
Variations to Game of Life:

Vampires



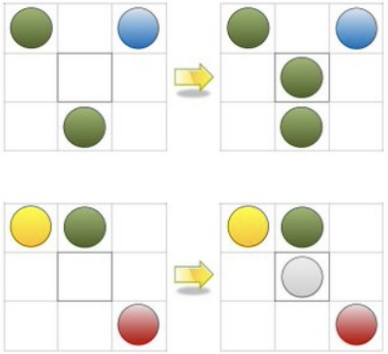
Cells age,
turn into undying vampires,
that kill each other

Hex



The board isn't square

Cells with Friends

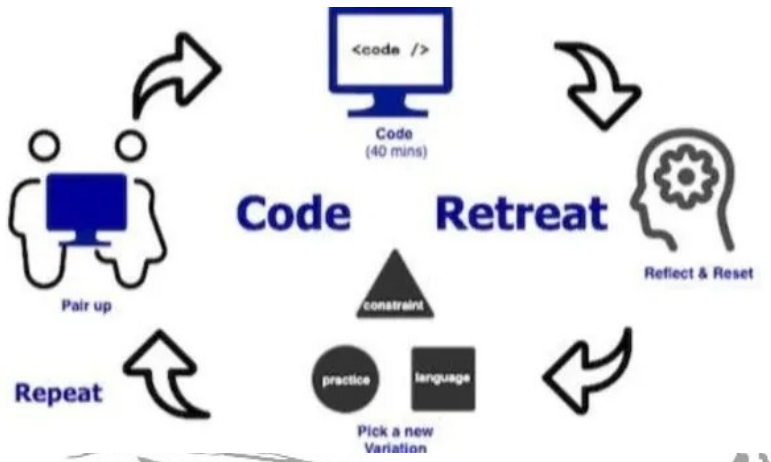


Cells have color
which is passed onto their
children

More info at:

CODERETREAT

Welcome!



Schedule

9:30 – 10:00 Introduction

10:00-11:00 Session 1

11:00-12:00 Session 2

11:00-12:00 Lunch

1:00 - 2:00 Session 3

2:00 - 3:00 Session 4

3:00 - 4:00 Session 5

4:00 - 4:30 Closing Circle

Session

40 mins - Code

10 mins - Retro

10 mins - Break

Need an environment?

CyberDojo

cyber-dojo.org

Online coding and testing in many languages

Starter Projects

github.com/LearnWithLlew/StarterProjects

Clone and go

exercism

exercism.org

64 languages with setup instructions

Gitpod.io

gitpod.io/#<http://your_github_url>

VS Code online with linux
Can install and run via the terminal

elixir

```elixir
def hello_world do
 "Hello World!"
end
```

Ruby

```ruby
def hello_world
  "Hello World!"
end
```

clojure

```clojure
(defn hello-world []
 "Hello World!")
```

Elm

```elm
helloWorld : String
helloWorld = "Hello World!"
```

Haskell

```haskell
helloWorld :: String
helloWorld = "Hello World!"
```

```cpp
string helloWorld() {
  return "Hello World!";
}
```

Lua

```lua
function helloWorld()
  return "Hello World!"
end
```

```php
function helloWorld() {
  return "Hello World!";
}
```

```javascript
function helloWorld() {
  return "Hello World!";
}
```

TypeScript

```typescript
function helloWorld(): string {
 return "Hello World!";
}
```

```csharp
public static string Hello() {
 return "Hello World!";
}
```

Go

```go
func helloWorld() string {
 return "Hello World!"
}
```

Swift

```swift
func helloWorld() -> String {
 return "Hello World!"
}
```

[OBJ-C]

```objc
NSString* helloWorld() {
 return @"Hello World!";
}
```

Perl

```perl
sub hello_world {
 return "Hello World!";
}
```

Kotlin

```kotlin
fun helloWorld(): String {
 return "Hello World!"
}
```

python

```python
def hello_world():
  return "Hello World!"
```

julia

```julia
function helloWorld()
  return "Hello World!"
end
```

Java

```java
public static String helloWorld()
{
 return "Hello World!";
}
```

Rust

```rust
fn hello_world() -> &'static str
{
 "Hello World!"
}
```

OCaml

```ocaml
let hello_world () = "Hello
World!"
```

Prolog

```prolog
hello_world :- write('Hello World!').
```

REASON ML

```reasonml
let helloWorld = () => "Hello World!";
```

./jq

```jq
def helloWorld: "Hello World!";
```

CoffeeScript

```coffeescript
helloWorld = -> "Hello World!"
```

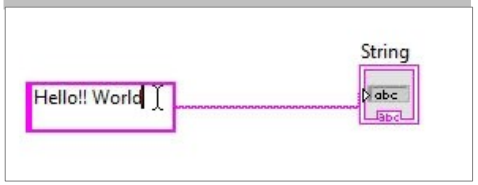
ERLANG

```erlang
hello_world() -> "Hello World!".
```

F#

```fsharp
let helloWorld() = "Hello
World!"
```

LabVIEW

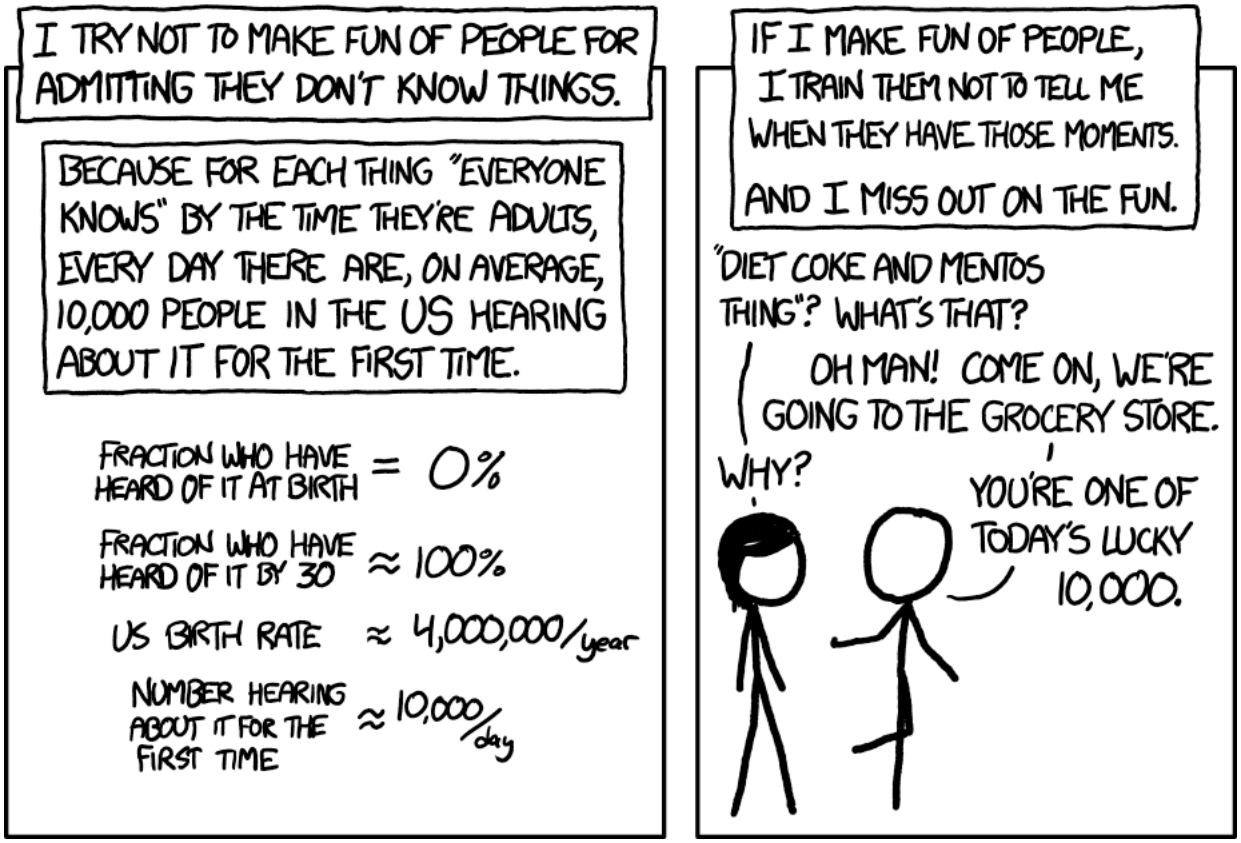


The Federation of the Functional
Elm
Haskell
F#
[OBJ-C]
ERLANG
elixir
clojure
OCaml
C#
Java
The Enterprise Empire
RE
REASON ML
TypeScript
JS
Ruby
python
Perl
Lua
The Land of Scripting
./jq
Isle of Misfits
julia
Prolog
LabVIEW
Frontend Frontier
CoffeeScript
Swift
Kotlin
C++
Performance Central
Go
Rust

What Surprised you today?

What would you like to try today?

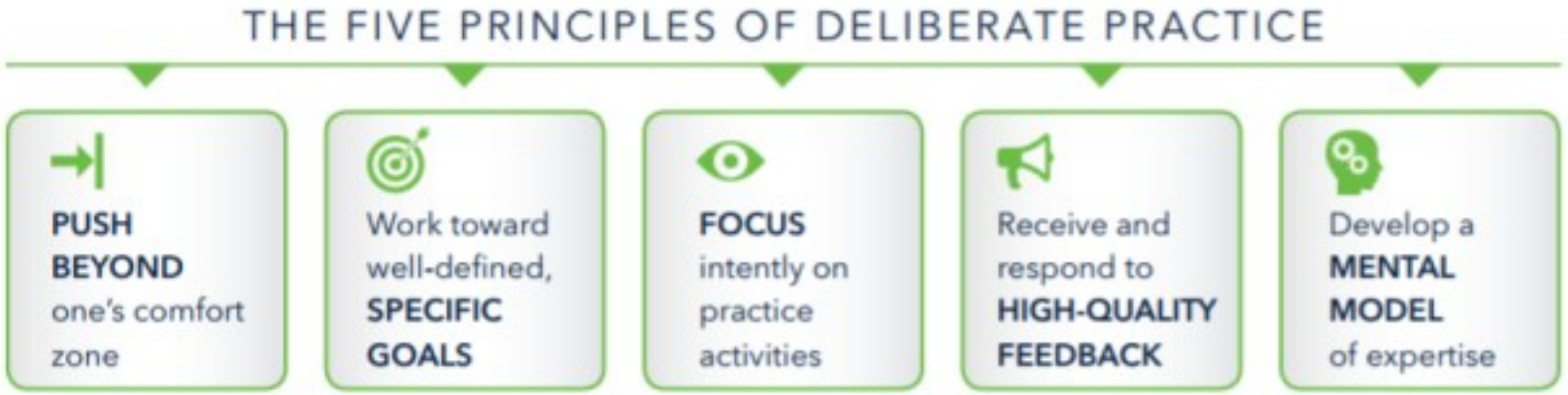
What did you learn today?



xkcd.com

*Many people have **strong opinions***
*about things they have **not experienced***

Hate it with data

*Don't make up **your mind***
*about something without **trying it.***

Deliberate Practice