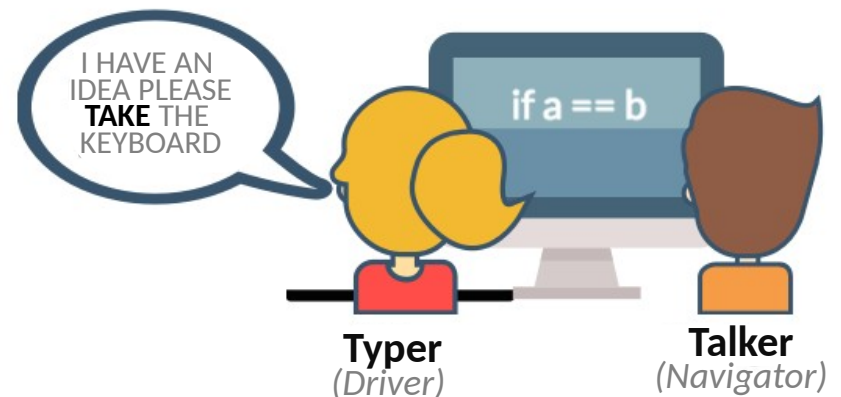


TRADITIONAL



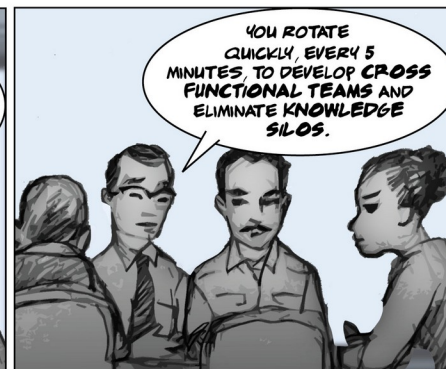
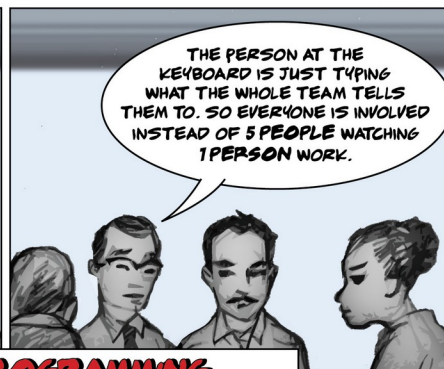
"STRONG STYLE"

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



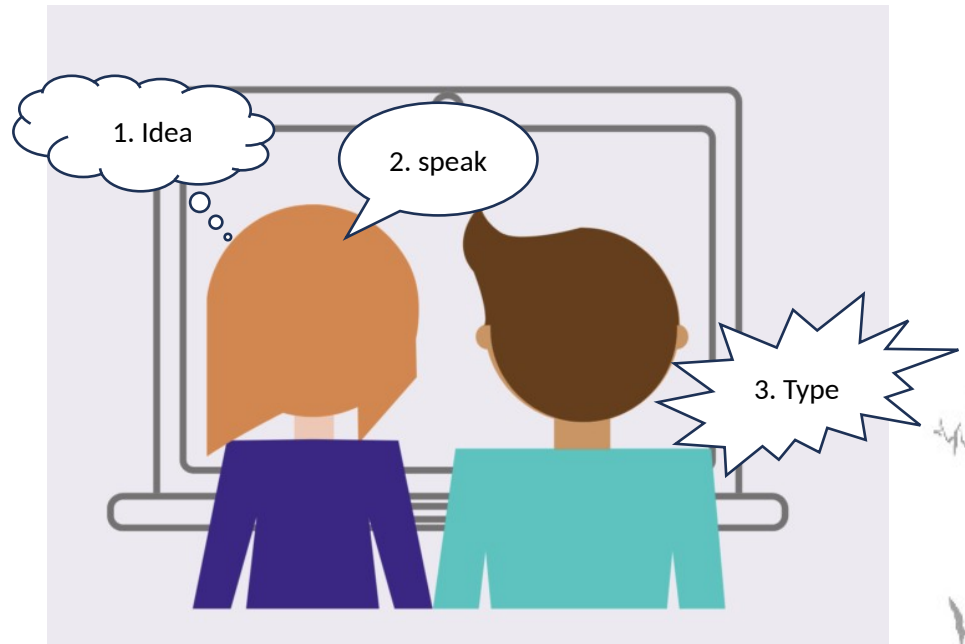


THE MOB PROGRAMMING GUIDEBOOK

Llewellyn Falco & Maaret Pyhäjärvi



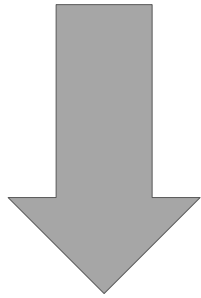
“For an idea to go from
your head to the **computer**
it must go through
someone else’s hands”



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 1/2 "

Details

- "for(int i = 0;"

The 5 Whys

Define the Problem

The client refused to pay the progress payment



Why is it Happening?

we completed the activity late



Why is it Happening?

It took longer than we estimated



Why is that?

We couldn't procure enough material



Why is that?

We didn't purchase on time



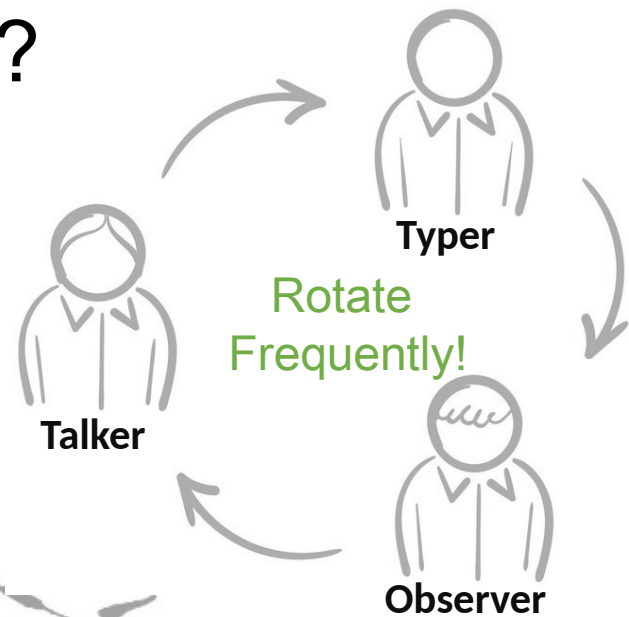
Why is that?

We didn't analyze the work schedule



Root Cause

When is the last time you
rotated ?



Rotation Styles



On Task



On Time



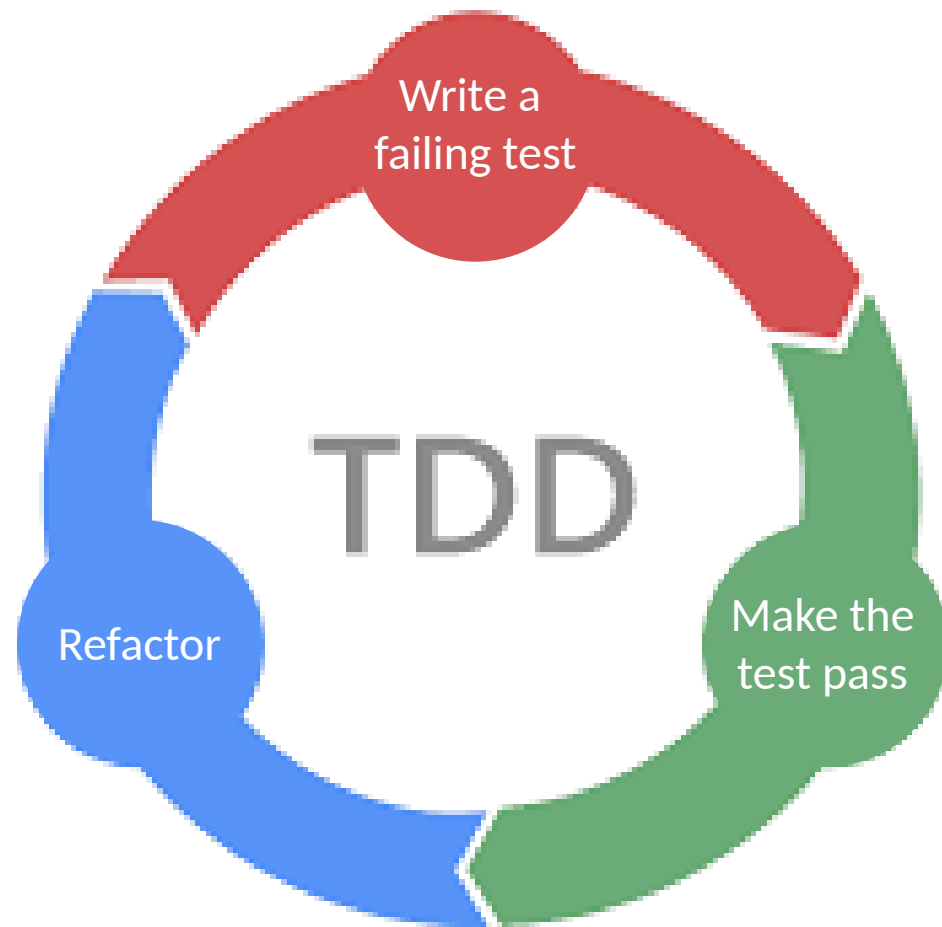
On Idea



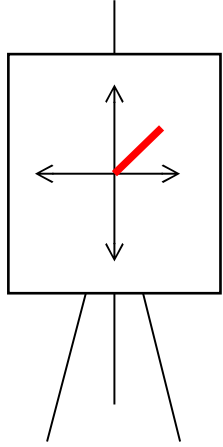
*Kindness
Consideration
and
Respect*

Working Agreements

Make for better pairs & mobs



Start Here!
with **drawing**
and natural language



Whiteboard

```
// Create side (0,0) - (3,4)  
// Verify length
```

English

Testing Circle

Code

Result

```
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](https://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

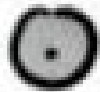
1. Specification - what am I going to write?
2. Feedback - did it work?
3. Regression - does it still work?
4. Granularity - why did it stop working?

The 4 Benefits of Tests

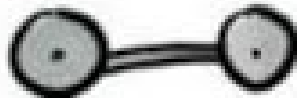
Blog



Not like this....



1



2



3



4

Like this!



1



2



3



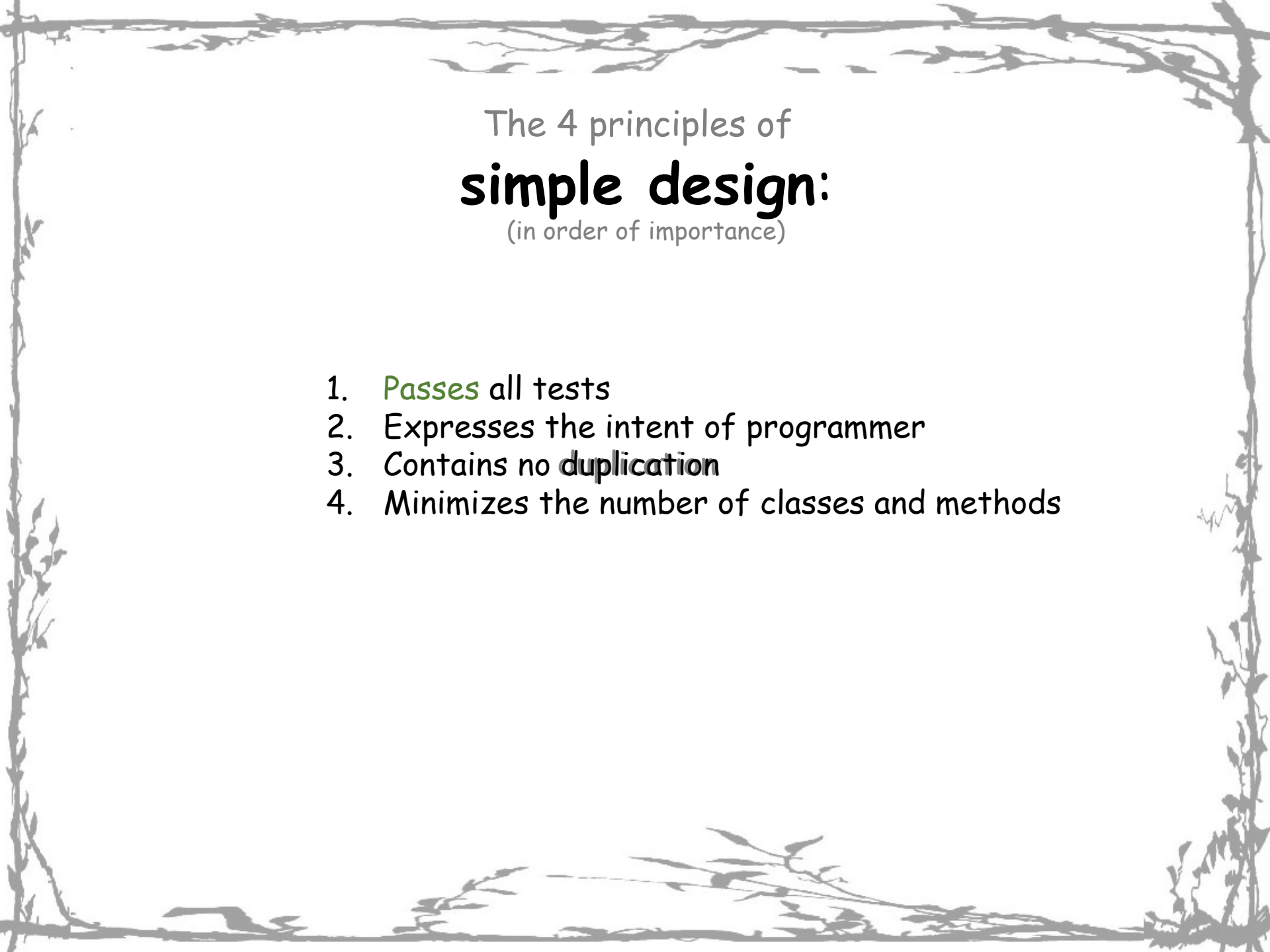
4



5

Hervik Kriberg

Building an MVP
(minimal viable product)

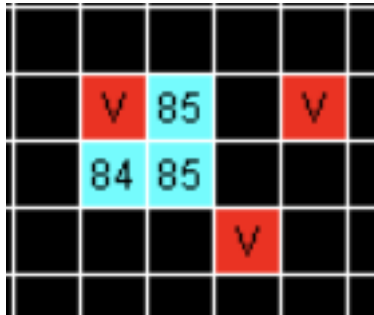


The 4 principles of **simple design:** (in order of importance)

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

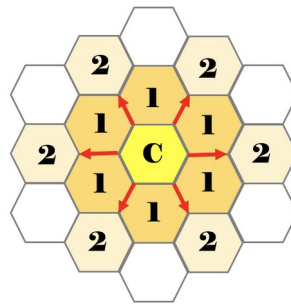
Variations to Game of Life:

Vampires



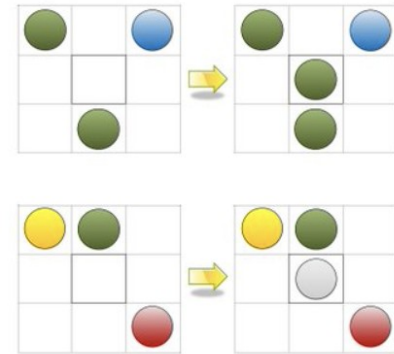
*Cells age,
turn into undying vampires,
that kill each other*

Hex



The board isn't square

Cells with Friends



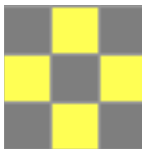
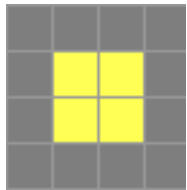
*Cells have color
which is passed onto their
children*

More info at:

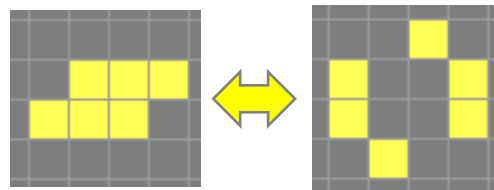
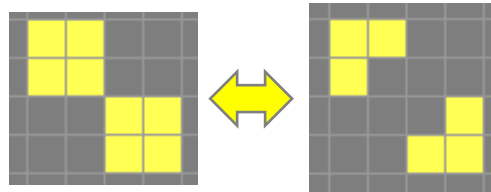
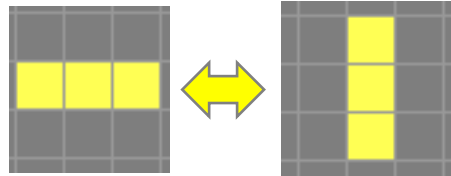


Interesting Game of Life Structures

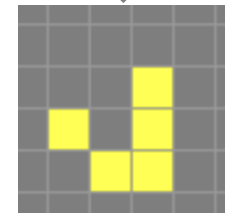
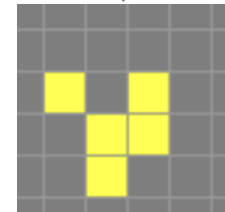
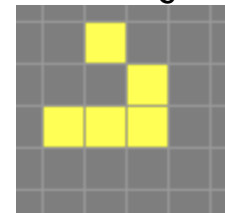
Unchanging



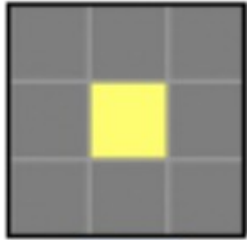
Oscillating



Traveling

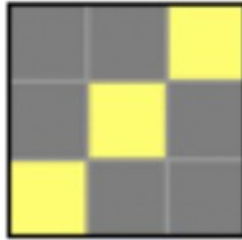


1 neighbor or less



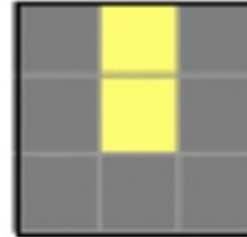
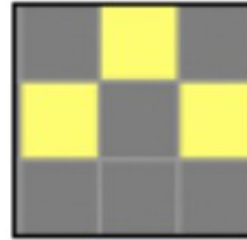
Middle square
dies
from **Starvation**

2-3 neighbors



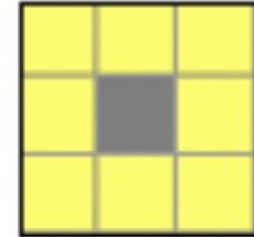
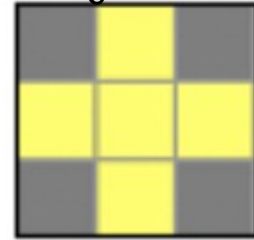
Middle square
survives
from **Subsistence**

3 neighbors



Middle square
is born

4 or more neighbors



Middle square
dies
from **Overpopulation**

Game of Life Rules

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*



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



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



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



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



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



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



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



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



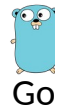
```
function helloWorld() {
  return "Hello World!";
}
```



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



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



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



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



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



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



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



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



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



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



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



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



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



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



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



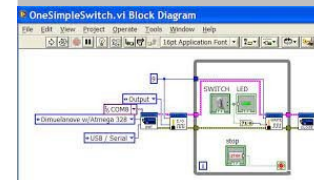
```
helloWorld = -> "Hello World!"
```



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



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





The Federation of the Functional



Elm



Haskell



ERLANG



elixir



clojure



Perl



The Land of Scripting



Java

The Enterprise Empire



./jq

Isle of Misfits



julia



Prolog



LabVIEW



REASON ML



OCaml



TypeScript



JS



Ruby



python

Frontend Frontier



CoffeeScript



Swift



Kotlin



C++

Performance Central




Rust



Go



What **Surprised** you today?



What would you like to **try**
today?

What did you **learn** today?

I TRY NOT TO MAKE FUN OF PEOPLE FOR ADMITTING THEY DON'T KNOW THINGS.

BECAUSE FOR EACH THING "EVERYONE KNOWS" BY THE TIME THEY'RE ADULTS, EVERY DAY THERE ARE, ON AVERAGE, 10,000 PEOPLE IN THE US HEARING ABOUT IT FOR THE FIRST TIME.

FRACTION WHO HAVE HEARD OF IT AT BIRTH = 0%

FRACTION WHO HAVE HEARD OF IT BY 30 $\approx 100\%$

US BIRTH RATE $\approx 4,000,000/\text{year}$

NUMBER HEARING ABOUT IT FOR THE FIRST TIME $\approx 10,000/\text{day}$

IF I MAKE FUN OF PEOPLE, I TRAIN THEM NOT TO TELL ME WHEN THEY HAVE THOSE MOMENTS. AND I MISS OUT ON THE FUN.

"DIET COKE AND MENTOS THING"? WHAT'S THAT?

OH MAN! COME ON, WE'RE GOING TO THE GROCERY STORE.

WHY?

YOU'RE ONE OF TODAY'S LUCKY 10,000.



*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

THE FIVE PRINCIPLES OF DELIBERATE PRACTICE



**PUSH
BEYOND**
one's comfort
zone



Work toward
well-defined,
**SPECIFIC
GOALS**



FOCUS
intently on
practice
activities



Receive and
respond to
**HIGH-QUALITY
FEEDBACK**



Develop a
**MENTAL
MODEL**
of expertise