

Welcome  
Please  
Come In

Sthlm



elojure  
Bridge

# Who am I ?

Maja Kontrec

maja@kontrec.com

*Erlang*



# What is programming?

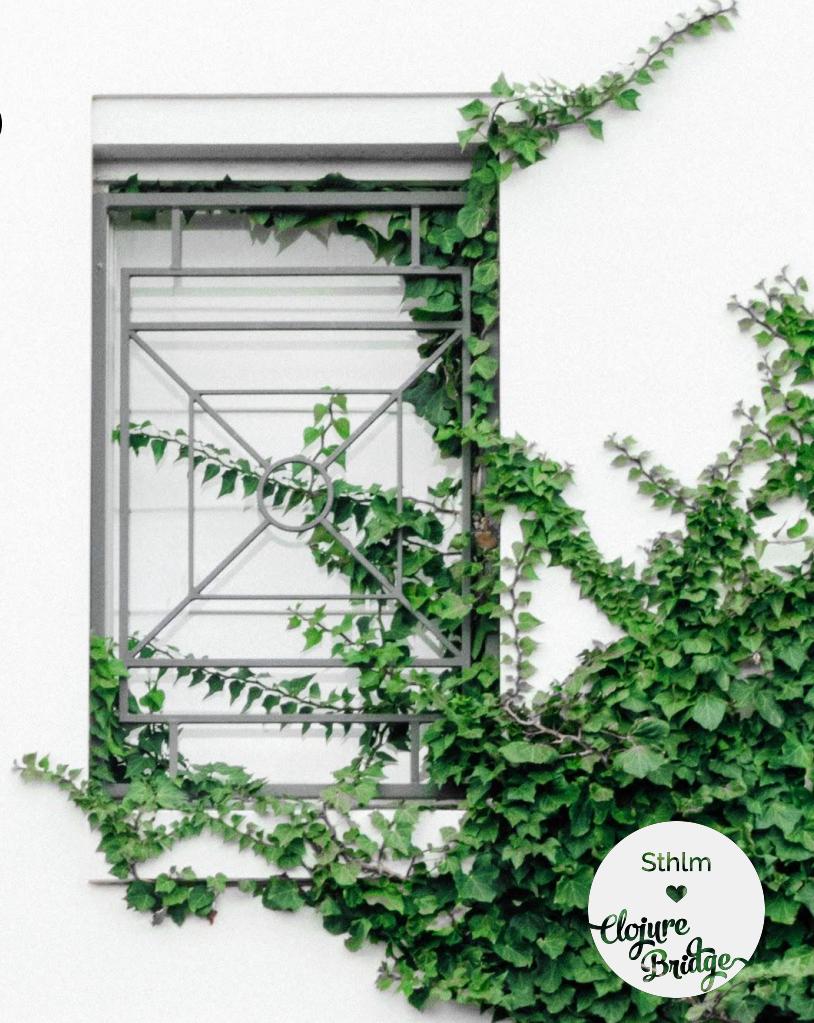


# Why Clojure?

simple

all-purpose

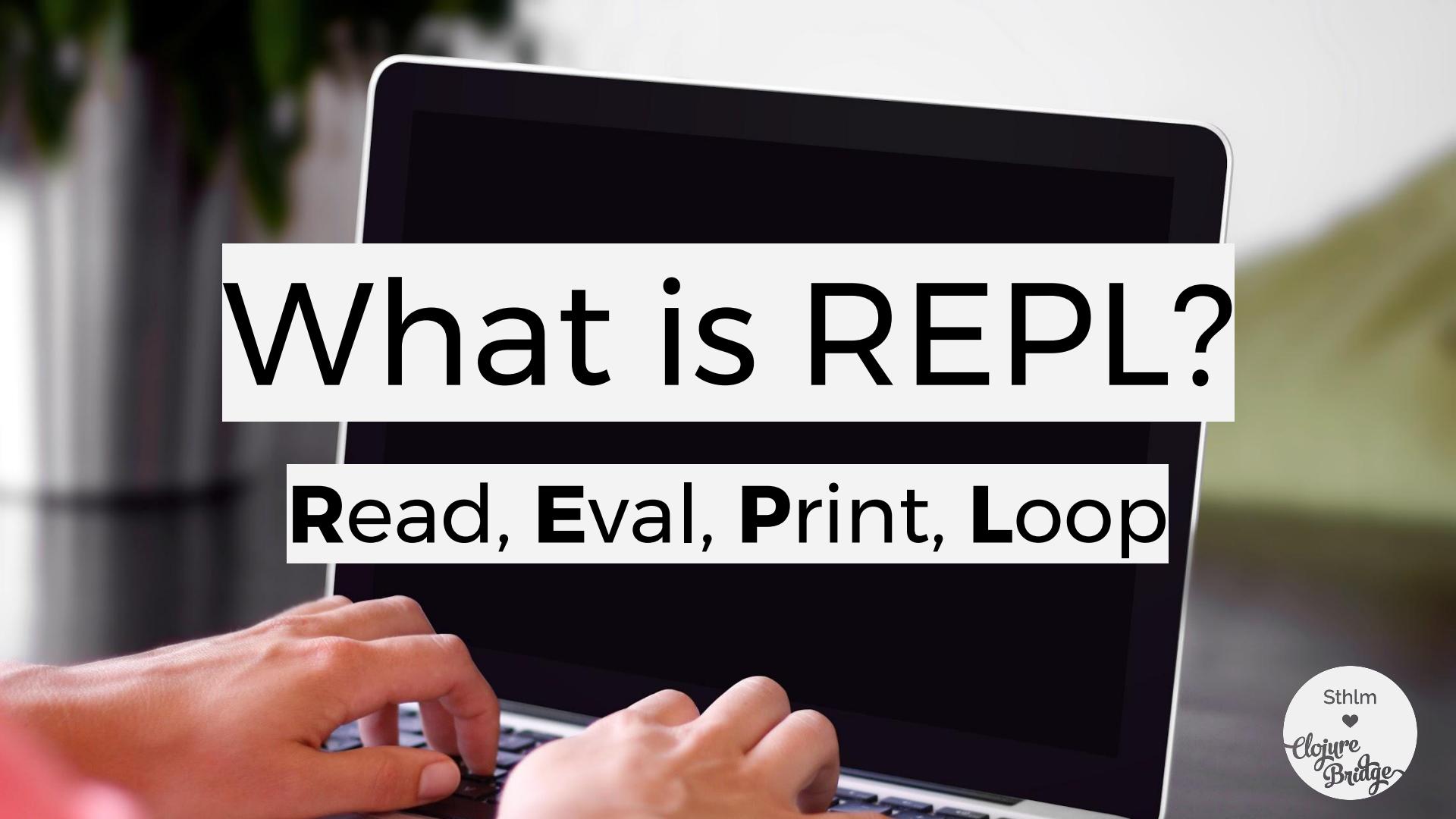
fun



# What does Clojure look like?

```
;; this is a comment  
  
(print-str "Hello, World!") ;prints Hello, World!  
  
(+ 3 4)  
  
(forward :trinity 40)
```



A close-up photograph of a person's hands typing on a dark laptop keyboard. The background is blurred, showing what appears to be an indoor setting with some greenery.

# **What is REPL?**

## **Read, Eval, Print, Loop**



# Try Nightcode InstaREPL

## Exercise 1

Exercises



# What are turtles?

Exercise 2 - Evaluate file and line

Exercises



# Look at Clojure docs

Exercise 3

Exercises



# Simple values

FIRE  
EXIT



# Strings = Text

"Hello, World!"

"This is a longer string that I wrote for purposes of an example."



# Booleans and nil

false

true

nil



# :Keywords

:TRIPS

:RECIPES

:MOVIES

# Numbers

0

12

-42

0.0000072725

10.5

-99.9

1/2

-7/3

Integers

Decimal  
numbers

Ratios



# Arithmetic

```
;;2+1  
(+ 2 1)
```



# Assignment: def



# Assigning names to values

```
(def mangoes 3)  
(def oranges 5)  
(+ mangoes oranges)  
;=> 8
```



# Assigning names to results

```
(def fruit-amount (+ mangoes oranges))  
(def average-fruit-amount (/ fruit-amount 2))  
average-fruit-amount  
;=> 4
```



# Basic arithmetic

## Exercise 4

Exercises



# Data Structures

Vectors/Maps



# Vectors

0

1

2

3

4

5

"a"	"b"	"c"	"d"	"e"	"f"
-----	-----	-----	-----	-----	-----



# Usage

```
(count [5 10 15])  
;=> 3  
(nth [5 10 15] 1)  
;=> 10
```

```
(first [5 10 15])  
;=> 5  
(rest [5 10 15])  
;=> (10 15)
```

```
(conj [5 10] 15)  
;=> [5 10 15]
```



# Remember Turtles?

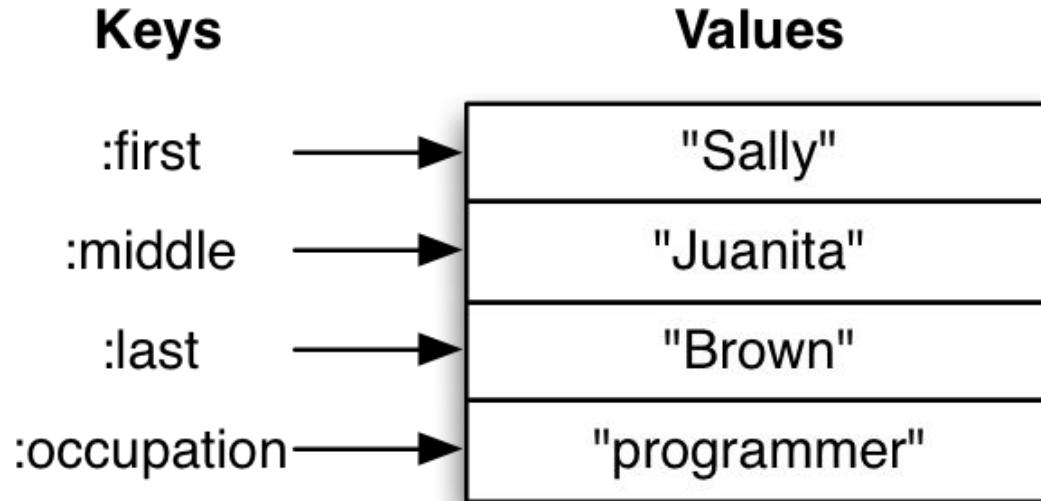
Exercise 5 - Turtle name

Exercise 6 - Create a vector

Exercises



# Maps



# Maps

```
{:first "Sally" :last "Brown"}  
{:a 1 :b "two"}  
{}
```



# Usage

```
(count {:first "Sally" :last "Brown"})  
;=> 2
```

```
(get {:first "Sally" :last "Brown"} :first)  
;=> "Sally"  
(get {:first "Sally"}) :last)  
;=> nil
```



# Usage

```
(assoc {:first "Sally"} :last "Brown")
;=> {:first "Sally", :last "Brown"}
```

```
(dissoc {:first "Sally" :last "Brown"} :last)
;=> {:first "Sally"}
```

```
(merge {:first "Sally"} {:last "Brown"})
;=> {:first "Sally", :last "Brown"}
```



# Usage

```
(keys {:first "Sally" :last "Brown"})  
;=> (:first :last)
```

```
(vals {:first "Sally" :last "Brown"})  
;=> ("Sally" "Brown")
```



# More on Turtles

Exercise 7 - See turtles states

Exercise 8 - Modeling yourself

Exercises





# Flow control

# if

```
(if (= traffic-light :red)
    (stop)
    (go))
```



# Y value within a frame (part 1)

Exercise 9

Exercises



# cond

```
(cond  
  (= traffic-light :red) (stop)  
  (= traffic-light :yellow) (slow-down)  
  :else (go))
```



# Y value within a frame (part 2)

Exercise 10

Exercises



# and, or, not

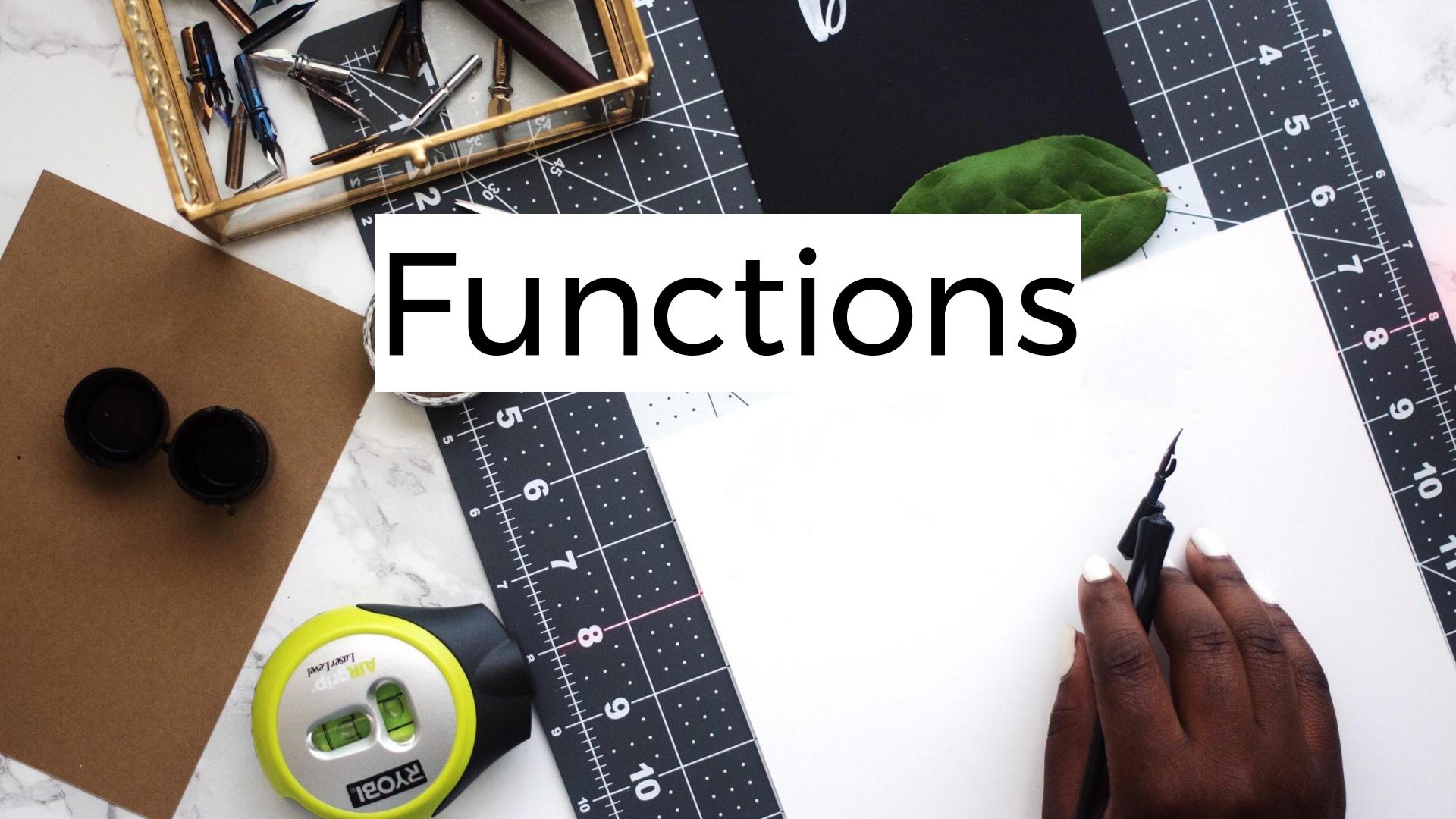
```
(and true false) ;=> false
```

```
(or true false) ;=> true
```

```
(not true) ;=> false
```



# Functions



value(s)



function



result

# Definition

```
(defn forward-right
  "Moves specified turtle forward and tilts its head"
  [turtle]
  (forward turtle 60)
  (right turtle 135))
```

# Usage

```
(forward-right :trinity) ;=> {:trinity {:angle 135}}
(forward-right :neo) ;=> {:neo {:angle 135}}
```



# Multiple arguments

```
(defn forward-right-with-len
  "Given turtle and length, forward the turtle and tilts its head"
  [turtle len]
  (forward turtle len)
  (right turtle 135))

(forward-right-with-len :trinity 90) ;=> {:trinity {:angle 135}}
(forward-right-with-len :neo 80) ;=> {:neo {:angle 135}}
```



# Move turtles using function

Exercise 11

Exercises



# Move turtles using function with parameters

Exercise 12

Exercises



# Your Turn!



# Drawing Lines

## j.mp/2qLQKIM



# Turtles Walk

(more function study) <http://j.mp/2pkMR60>

# Snowflakes

(animation) <http://j.mp/2pMs7qM>

# Twinkle Twinkle Little Star -

(making sounds) <http://bit.ly/2pgEvvi>

# Global Growth

(web app with REST api) <http://bit.ly/2qNPRs9>

# Caesar Cipher

(mini exercise of Strings and Characters) <http://bit.ly/2pMHiQE>

