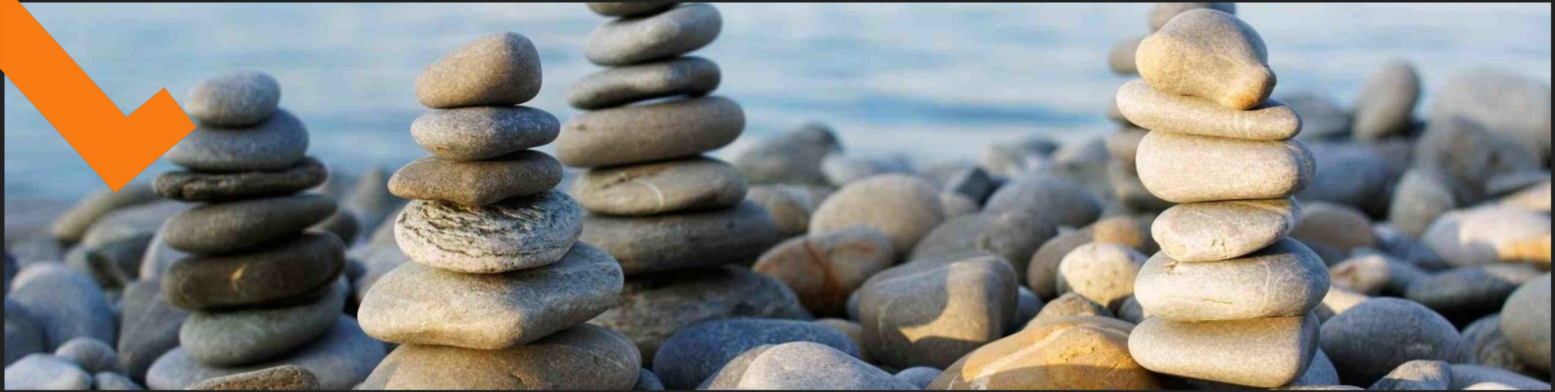
Lambdas & Pebbles



What is Lambda Calculus?

- Allows one to describe computation from a functional point of view.
- Invented by Alonzo Church in the late 1920s.

What is Lambda Calculus?

- Allows one to describe computation from a functional point of view.
- Invented by Alonzo Church in the late 1920s.

Historical Side Note

- Allows one to describe computation from state machine point of view.
- Invented by Alan Turing in the mid 1930s.

Grammar of Lambda Calculus

```
<Expression> ::= <identifier>

<Expression> ::= λ<identifier>.<Expression>

<Expression> ::= <Expression> <Expression>

<Expression> ::= (<Expression>)
```

What’s the point?

- Functional programming is all the rage.
  - Lambda calculus provides the foundation.

What's the point?

- Functional programming is all the rage.
  - Lambda calculus provides the foundation.
- Lambda calculus can encode any computable function.
  - Equivalent to a turing machine.

What’s the point?

- Functional programming is all the rage.
    - Lambda calculus provides the foundation.

- Lambda calculus can encode any computable function.
    - Equivalent to a turing machine.
- It’s simplicity and compositional power is **<u>awesomeness in its purest form</u>**.

A Lambda Expression that takes 2 Inputs

$$(\lambda x.\lambda y.\ x + y)$$

A Lambda Expression that takes 2 Inputs

$$(\lambda x.\lambda y.\ x + y)\ \ 13\ \ 9$$

A Lambda Expression that takes 2 Inputs

$$(\lambda x.\lambda y.\ x + y)\ \ 13\ \ 9$$

$$13 \longrightarrow \boxed{\lambda x.\lambda y.\ x + y}$$

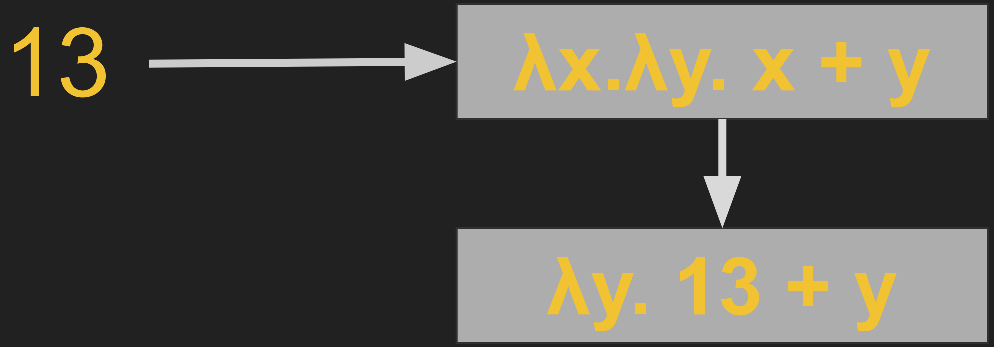
A Lambda Expression that takes 2 Inputs

$$(\lambda x.\lambda y.\ x + y)\ \ 13\ \ 9$$

$$13 \longrightarrow \boxed{\lambda x.\lambda y.\ x + y}$$

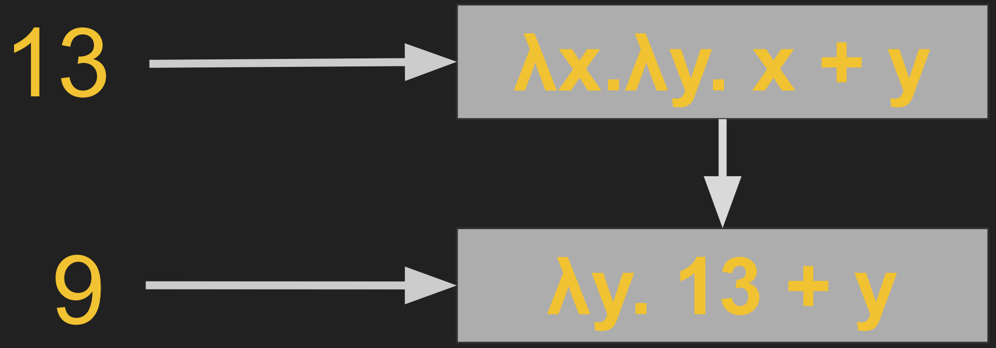
A Lambda Expression that takes 2 Inputs

$$(\lambda y.\ 13 + y)\ 9$$



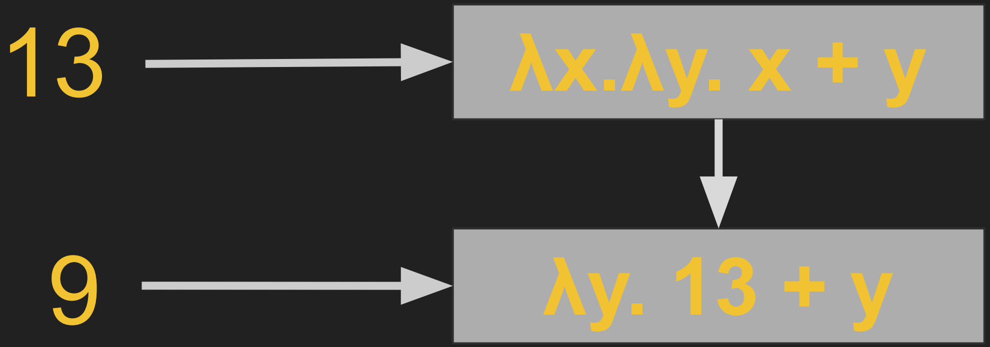
A Lambda Expression that takes 2 Inputs

$$(\lambda y.\ 13 + y)\ 9$$



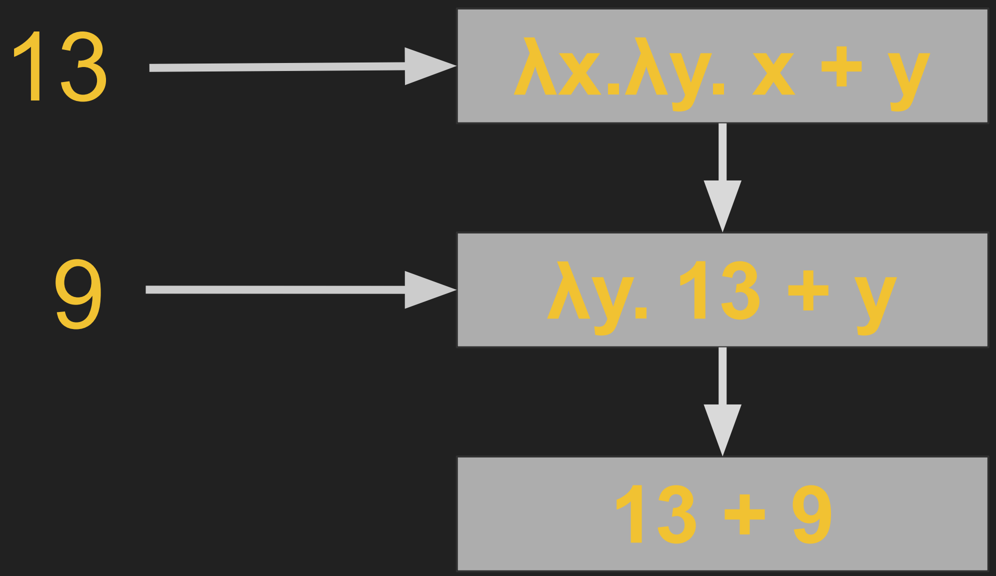
A Lambda Expression that takes 2 Inputs

$$(\lambda y.\ 13 + y)\ 9$$



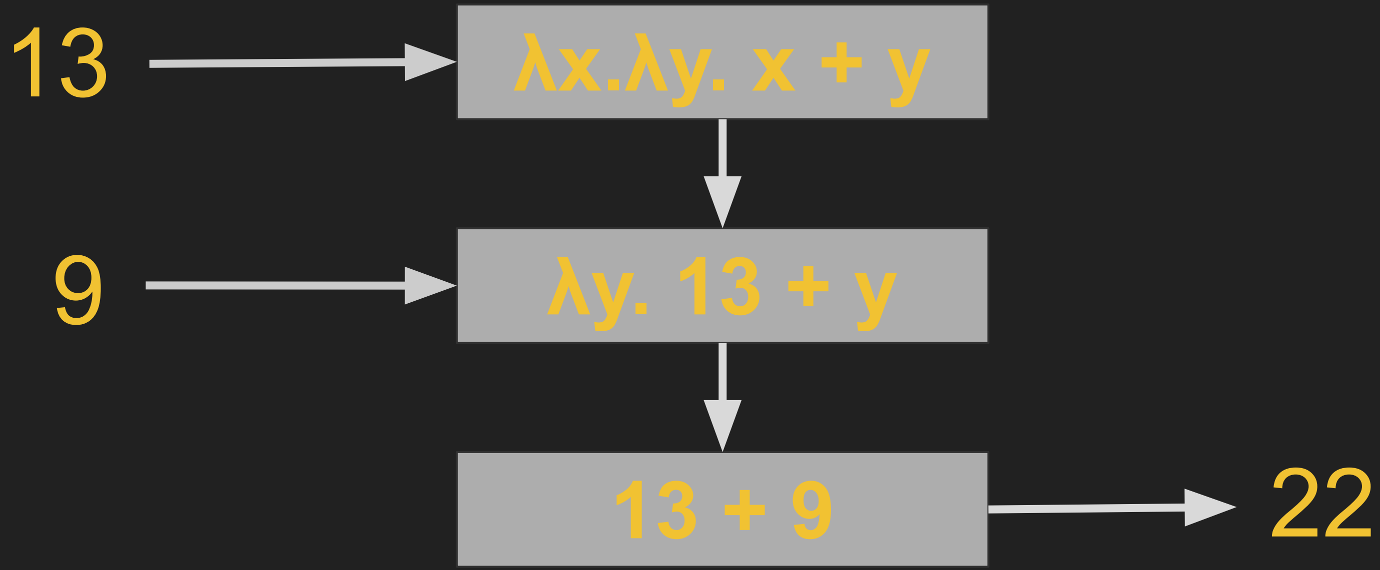
A Lambda Expression that takes 2 Inputs

$$(13 + 9)$$



A Lambda Expression that takes 2 Inputs

22



Church Numerals

- Given all we have are functions. How do we encode the notion of number?

Church Numerals

- Given all we have are functions. How do we encode the notion of number?
  - Really no choice… we have to use functions and function applications!

First 5 Church Numerals

0: $\lambda f.\lambda x.\ x$

1: $\lambda f.\lambda x.\ f\ x$

2: $\lambda f.\lambda x.\ f\ (f\ x)$

3: $\lambda f.\lambda x.\ f\ (f\ (f\ x))$

4: $\lambda f.\lambda x.\ f\ (f\ (f\ (f\ x)))$

First 5 Church Numerals

0: $\lambda f.\lambda x.\ x$

1: $\lambda f.\lambda x.\ f\ x$

2: $\lambda f.\lambda x.\ f\ (f\ x)$

3: $\lambda f.\lambda x.\ f\ (f\ (f\ x))$

4: $\lambda f.\lambda x.\ f\ (f\ (f\ (f\ x)))$

First 5 Church Numerals

0: $\lambda f.\lambda x.\ x$

1: $\lambda f.\lambda x.\ f\ x$

2: $\lambda f.\lambda x.\ f\ (f\ x)$

3: $\lambda f.\lambda x.\ f\ (f\ (f\ x))$

4: $\lambda f.\lambda x.\ f\ (f\ (f\ (f\ x)))$

Let’s encode these in Scheme!

Arithmetic with Church Numerals

- How do we operate with these numerals?

Arithmetic with Church Numerals

- How do we operate with these numerals?
  - Lambda Functions!

Arithmetic with Church Numerals

- How do we operate with these numerals?
    - Lambda Functions!

SUCC: $\lambda n.\lambda f.\lambda x.\ f\ (n\ f\ x)$

Arithmetic with Church Numerals

- How do we operate with these numerals?
  - Lambda Functions!

SUCC: $\lambda n.\lambda f.\lambda x.\ f\ (n\ f\ x)$

$f\ \underbrace{fff.....fff}_{n}$

Arithmetic with Church Numerals

- How do we operate with these numerals?
  - Lambda Functions!

SUCC: $\lambda n.\lambda f.\lambda x.\ f\ (n\ f\ x)$ $\qquad f\underbrace{fff.....fff}_{n}$

ADD: $\lambda m.\lambda n.\lambda f.\lambda x.\ m\ f\ (n\ f\ x)$

Arithmetic with Church Numerals

- How do we operate with these numerals?
  - Lambda Functions!

SUCC: $\lambda n.\lambda f.\lambda x.\ f\ (n\ f\ x)$ $\qquad f\ \underbrace{fff.....fff}_{n}$

ADD: $\lambda m.\lambda n.\lambda f.\lambda x.\ m\ f\ (n\ f\ x)$ $\qquad \underbrace{ff..ff}_{m}\ \underbrace{ff..ff}_{n}$

Arithmetic with Church Numerals

- How do we operate with these numerals?
  - Lambda Functions!

SUCC: $\lambda n.\lambda f.\lambda x.\ f\ (n\ f\ x)$ $\quad f\,\{fff.....fff\}_n$

ADD: $\lambda m.\lambda n.\lambda f.\lambda x.\ m\ f\ (n\ f\ x)$ $\quad \{ff..ff\}_m\ \{ff..ff\}_n$

MULT: $\lambda m.\lambda n.\lambda f.\lambda x.\ m\ (n\ f)\ x$

Arithmetic with Church Numerals

- How do we operate with these numerals?
    - Lambda Functions!

SUCC: λn.λf.λx. f (n f x) $f\,\underbrace{fff.....fff}_{n}$

ADD: λm.λn.λf.λx. m f (n f x) $\underbrace{ff..ff}_{m}\ \underbrace{ff..ff}_{n}$

MULT: λm.λn.λf.λx. m (n f) x $\underbrace{\underbrace{ff..ff}_{n}..\underbrace{ff..ff}_{n}}_{m}$

Arithmetic with Church Num

- How do we operate with thes
  - Lambda Functions!

SUCC: λn.λf.λx. f (n f x)

ADD: λm.λn.λf.λx. m f (

MULT: λm.λn.λf.λx. m (n

Let’s encode these in Scheme!

And now….

Y Combinator

$$\lambda f.(\lambda x.f(x\ x))(\lambda x.f(x\ x))$$

Y Combinator

$$\lambda f.(\lambda x.f(x\ x))(\lambda x.f(x\ x))$$

Y Combinator

- Also called a fixed point combinator
    - $x = f(x)$; x is a fixed point of f

Y Combinator

- Also called a fixed point combinator
    - $x = f(x)$; x is a fixed point of f

$$Y = \lambda f.(\lambda x.f(x\ x))(\lambda x.f(x\ x))$$

Y Combinator

- Also called a fixed point combinator
  - $x = f(x)$; x is a fixed point of f

Y g = (λf.(λx.f(x x))(λx.f(x x))) g

Y Combinator

- Also called a fixed point combinator
  - $x = f(x)$; x is a fixed point of f

$$Y\ g = (\lambda f.(\lambda x.f(x\ x))(\lambda x.f(x\ x)))\ g$$

$$\longrightarrow \quad (\lambda x.g(x\ x))(\lambda x.g(x\ x))$$

Y Combinator

- Also called a fixed point combinator
  - $x = f(x)$; x is a fixed point of f

$$Y\ g = (\lambda f.(\lambda x.f(x\ x))(\lambda x.f(x\ x)))\ g$$

$$\longrightarrow\ (\lambda x.g(x\ x))(\lambda x.g(x\ x))$$

$$\longrightarrow\ g((\lambda x.g(x\ x))(\lambda x.g(x\ x)))$$

Y Combinator

- Also called a fixed point combinator
  - $x = f(x)$; x is a fixed point of f

Y g = (λf.(λx.f(x x))(λx.f(x x))) g

⟶ (λx.g(x x))(λx.g(x x))

⟶ g((λx.g(x x))(λx.g(x x)))

Y Combinator

- Also called a fixed point combinator
  - $x = f(x)$; x is a fixed point of f

$$Y\ g = (\lambda f.(\lambda x.f(x\ x))(\lambda x.f(x\ x)))\ g$$

$$\longrightarrow (\lambda x.g(x\ x))(\lambda x.g(x\ x))$$

$$\longrightarrow g((\lambda x.g(x\ x))(\lambda x.g(x\ x))) \longrightarrow g(g((\lambda x.g(x\ x))(\lambda x.g(x\ x))))$$

Y Combinator

- Also called a fixed point combinator
  - $x = f(x)$; x is a fixed point of f

Y g = (λf.(λx.f(x x))(λx.f(x x))) g

⟶ (λx.g(x x))(λx.g(x x))

⟶ g((λx.g(x x))(λx.g(x x))) ⟶ g(g((λx.g(x x))(λx.g(x x))))

⟶ g(g(g((λx.g(x x))(λx.g(x x))))) ⟶ ……

Y Combinator

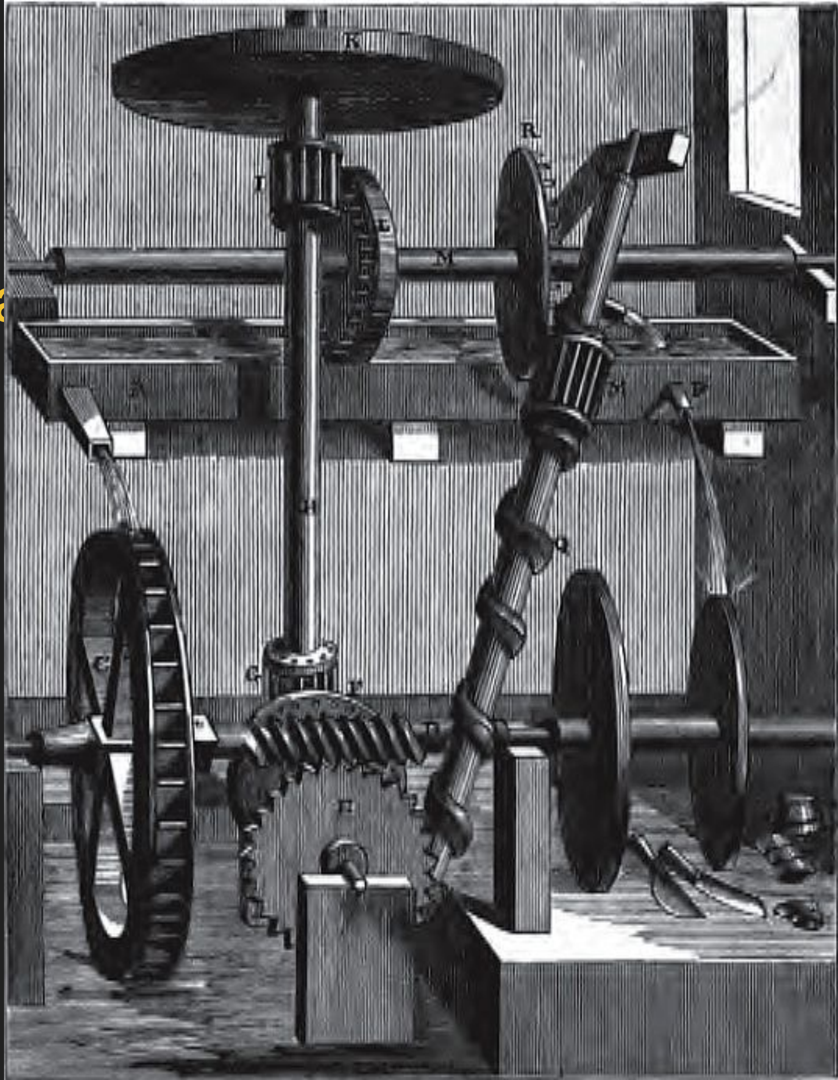
- Also called a fixed point combina
  - *x* = *f*(*x*); *x* is a fixed point of *f*

$$Y\ g = (\lambda f.(\lambda x.f(x\ x))(\lambda x.f(x\ x)))\ g$$

$$\longrightarrow\ (\lambda x.g(x\ x))(\lambda x.g(x\ x))$$

$$\longrightarrow\ g((\lambda x.g(x\ x))(\lambda x.g(x\ x)))\ \longrightarrow$$

$$\longrightarrow\ g(g(g((\lambda x.g(x\ x))(\lambda x.g(x\ x)))))$$



Y Combinator

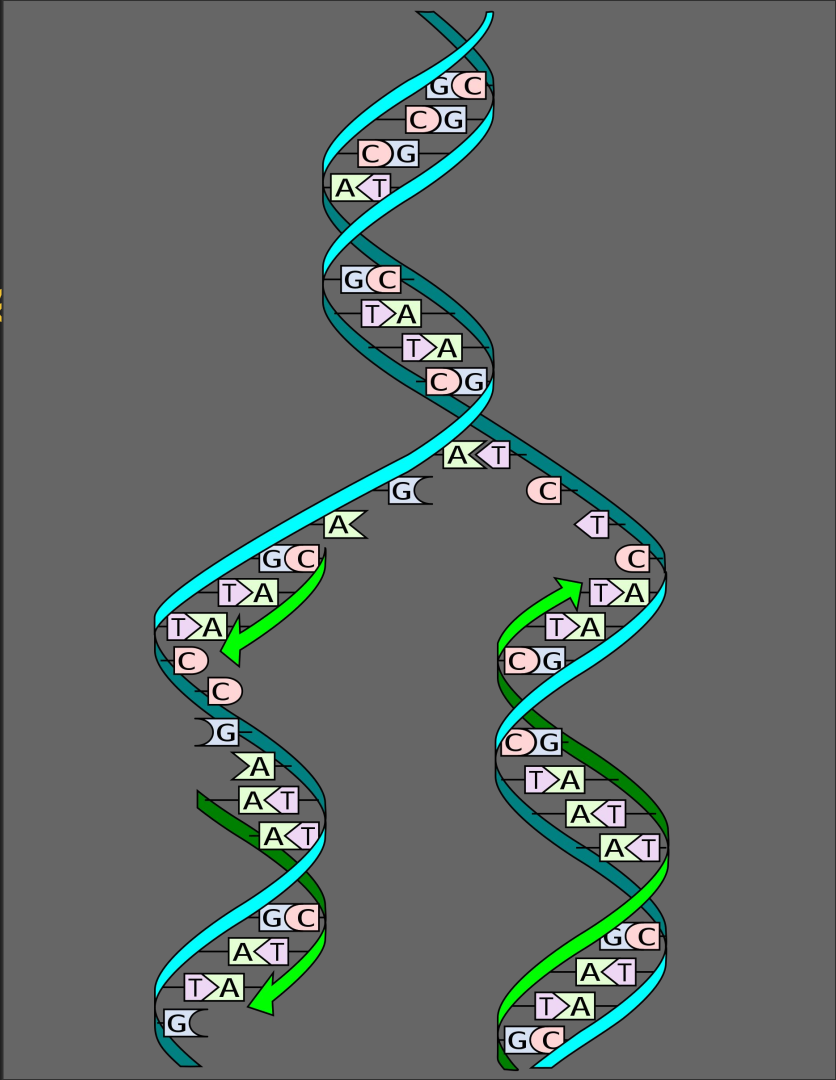
- Also called a fixed point combina
    - $x = f(x)$; x is a fixed point of f

$Y\ g = (\lambda f.(\lambda x.f(x\ x))(\lambda x.f(x\ x)))\ g$

$\longrightarrow\ (\lambda x.g(x\ x))(\lambda x.g(x\ x))$

$\longrightarrow\ g((\lambda x.g(x\ x))(\lambda x.g(x\ x)))\ \longrightarrow$

$\longrightarrow\ g(g(g((\lambda x.g(x\ x))(\lambda x.g(x\ x)))))$



Y Combinator

- Also called a fixed point combinator
    - $x = f(x)$; x is a fixed point of f

Y(g) = g(Y(g)) = g(g(Y(g)) = …

Y Combinator

- Also called a fixed point combinator
  - $x = f(x)$; x is a fixed point of f

Y(g) = g(Y(g)) = …

Y Combinator

- Also called a fixed point com
  - $x = f(x)$; x is a fixed point

$Y(g) = g(Y(g)) = \ldots$

Let's do factorial in Scheme!

(caveat: using the Z combinator)

Y Combinator

- Also called a fixed point com
  - $x = f(x)$; x is a fixed point

$$Y(g) = g(Y(g)) = \ldots$$

Historical Side Note

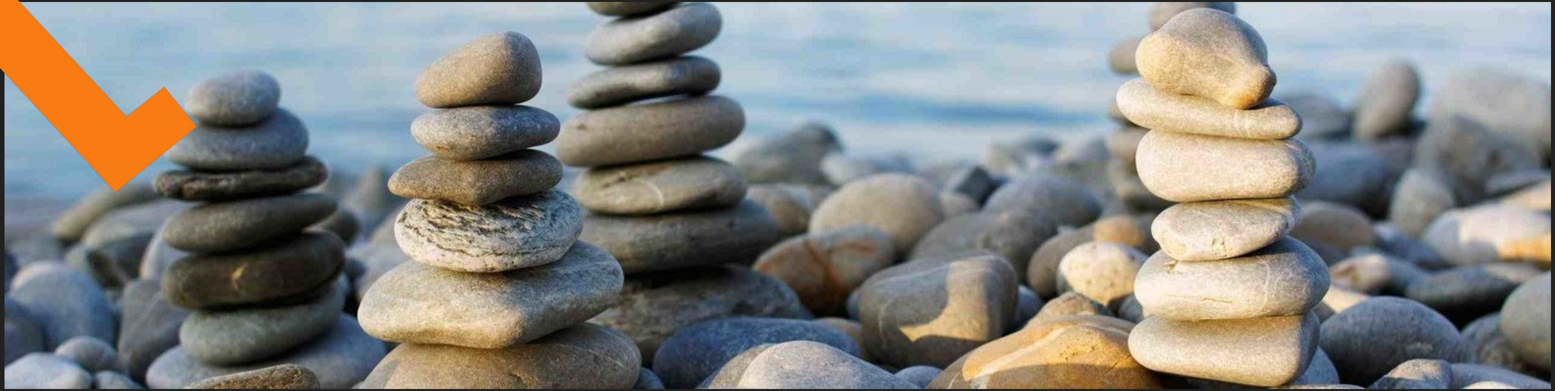
Y Combinator is an American seed accelerator, started in March 2005. Y Combinator is consistently ranked at the top of U.S. accelerators

https://www.ycombinator.com

Why did you choose the name "Y Combinator?"

The Y combinator is one of the coolest ideas in computer science. It's also a metaphor for what we do. It's a program that runs programs; we're a company that helps start companies.

Lambdas & Pebbles



$$\lambda f.(\lambda x.f(x\ x))(\lambda x.f(x\ x))$$