CSE114A lecture 2

agenda: Intro to lambda $\boxed{\text{calculus}}$

Everything is a function!

$$2 + 3$$

↑ ↑ ↑

functions!

a system of reasoning

1936 - Turing machines were invented! (Alan Turing)
and,
lambda calculus was invented! (Alonzo Church)

input → $\boxed{}$ → output

identity function: $\boxed{\lambda x.\ x}$

λ lambda

λx binder

delimiter

body

formal parameter

(in Elsa: $\boxed{\text{\x -> x}}$)

$\boxed{\text{\y -> y}}$ also the identity function

Python:

```
lambda x : x
```

```
def identity(y):
    return y
```

Quiz question 1:
What does this function do?

$$\lambda z.\ (\lambda x.\ x)$$

\z -> (\x -> x)

function application, aka calling a function

(\x -> x) "yuki"

Put the function next to its argument!

Python:

```
> identity("yuki")
"yuki"
> (lambda x: x)("yuki")
"yuki"
```

Convention:
the body of a function extends as far to the right ⟶ as possible

\x -> $\boxed{\text{x "yuki"}}$

This is a function call!

Python

```
lambda x : x("yuki")
```

(\x -> x "yuki") len

or sorted, or whatever

Quiz question 2: what does this function do?

$\boxed{\text{\f -> f (\x -> x)}}$

let's apply this function to \y -> y ...

How to evaluate this?

$\underbrace{\text{(\f -> f (\x -> x))}}_{\text{function}}\ \underbrace{\text{(\y -> y)}}_{\text{argument}}$

plug in the argument for occurrences of the formal parameter in the function's body.

So we get:

=> (\y -> y) (\x -> x)

(this is the body of the original function, but with the argument (\y -> y) substituted in for f.

Now what?

(\y -> y) (\x -> x)

=> \x -> x

The β-rule
aka the substitution rule:

(\x -> e1) e2 =b> e1 [x := e2]

This means:
e1, but with occurrences of x replaced by e2. *