

agenda:

- ✓ - Syntax, Semantics, pragmatics
- ✓ - intro to lambda calculus
- 5-10 minute break + quiz
- intro to Elsa

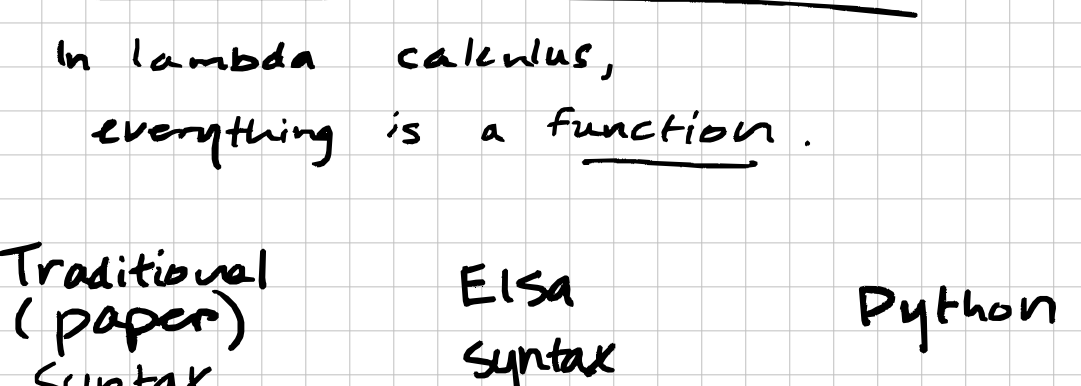
an analogy

human languages: Linguistics ::  
 programming languages: the field of PL  
 e.g. Python, Java, Rust, C, Haskell, etc.

Syntax - what languages and programs look like

**Semantics** - what languages and programs mean

pragmatics - how languages and programs are used



"a system of reasoning"

⊗ ⊗ ← Greek letter lambda

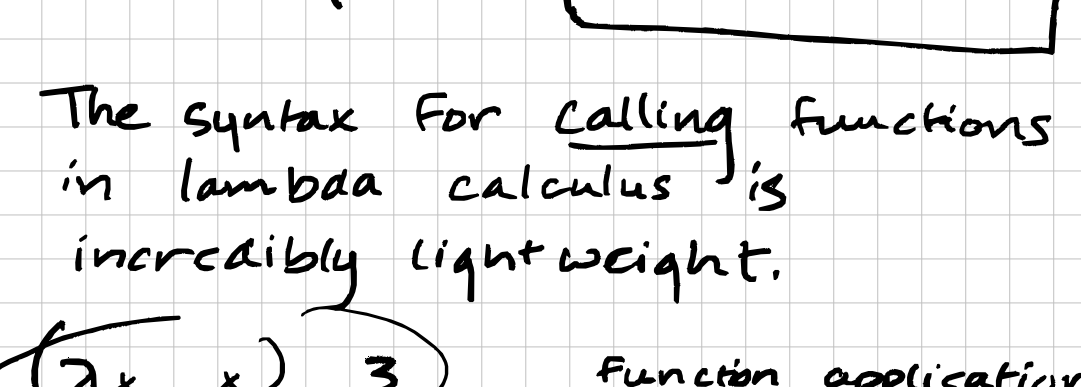
1936 - Turing machines were invented (Alan Turing)

1936 - lambda calculus was invented (Alonzo Church)

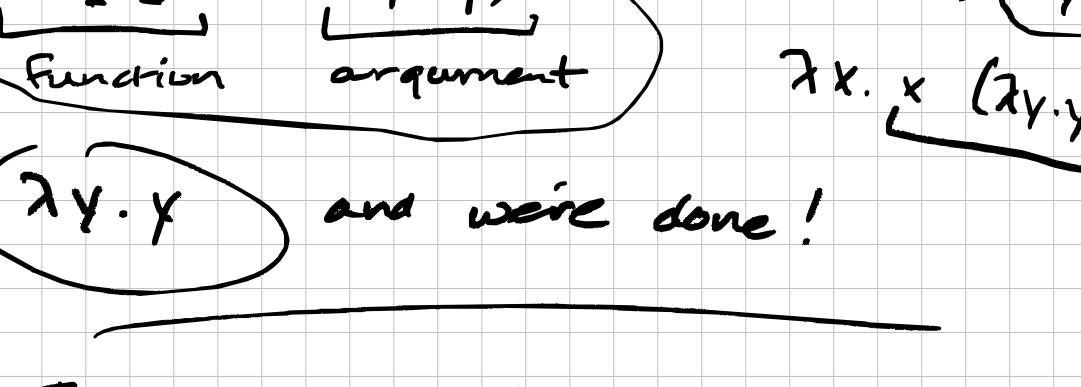
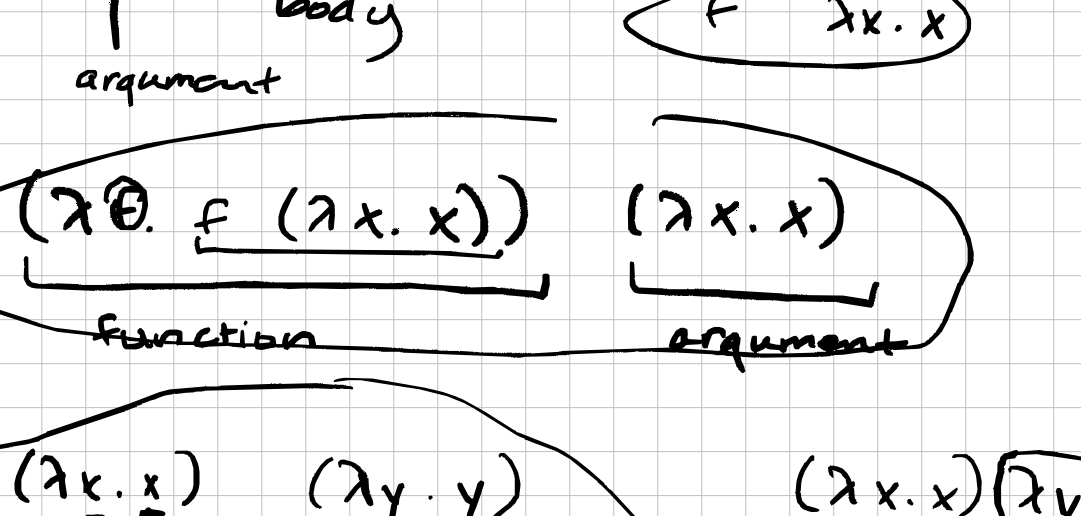
Both universal models of Computation!

In lambda calculus, everything is a function.

Traditional (paper) Syntax      Elsa Syntax      Python



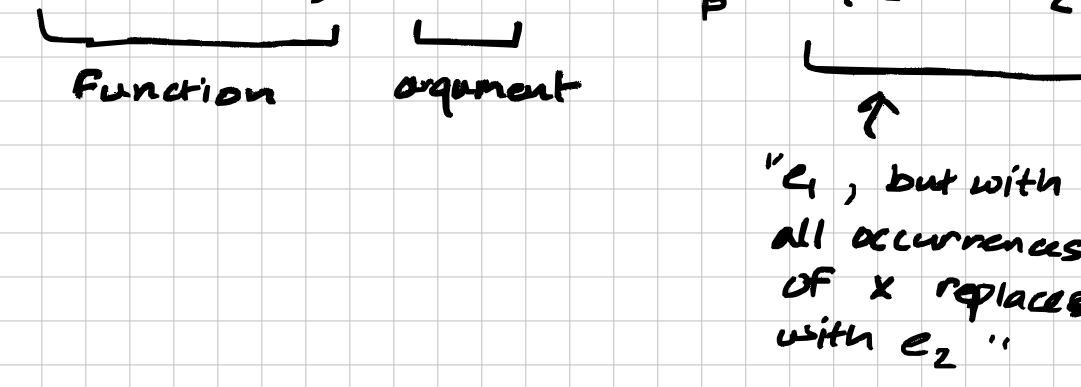
What does this mean?



$\lambda y. y$  is the same as  $\lambda x. x$

$\lambda z. (\lambda x. x)$  A function that takes an argument, ignores it, and returns the identity function.

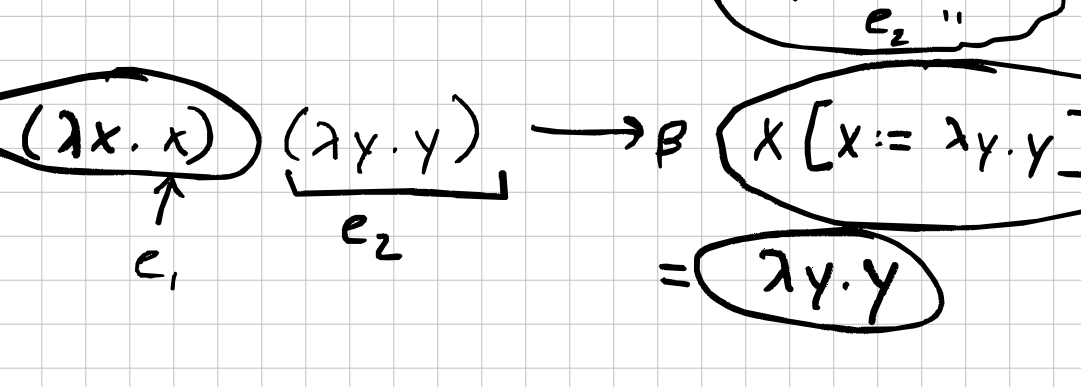
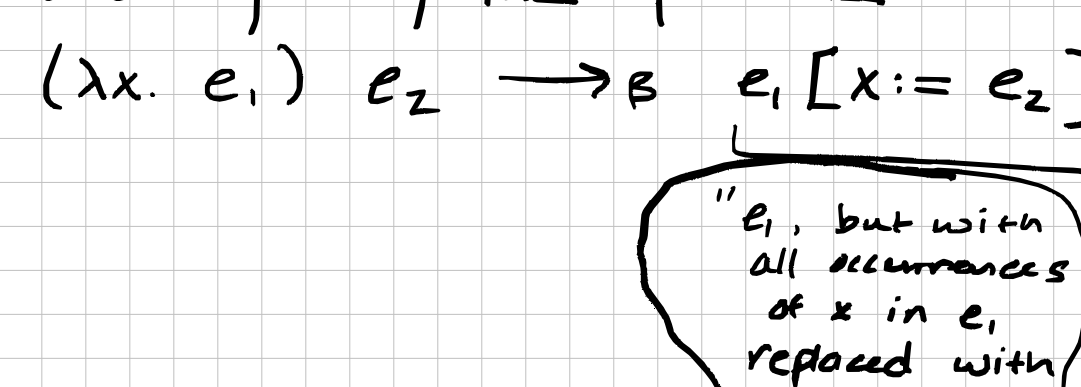
$\lambda y. (\lambda x. x)$  Same.



The syntax for calling functions in lambda calculus is incredibly lightweight.

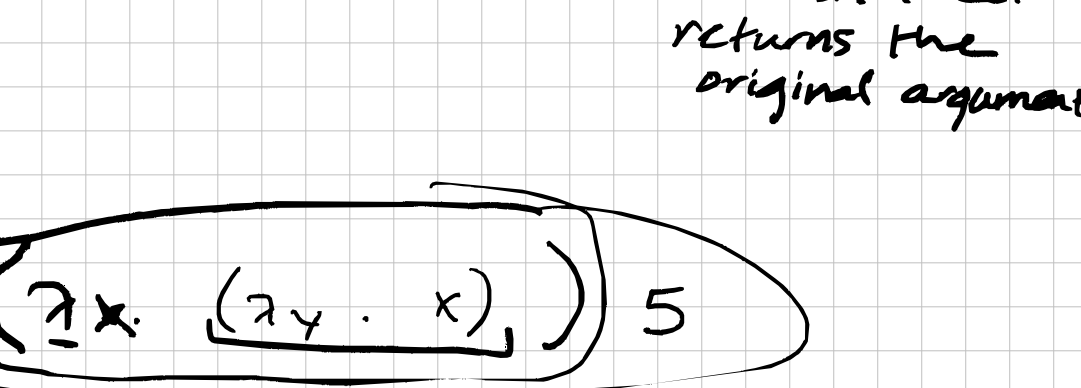
$(\lambda x. x) 3$  Function application

$\lambda F. (F (\lambda x. x))$  What does this do?  
 argument      body       $F \lambda x. x$



$\lambda y. y$  and were done!

The essence of computation in lambda calculus is substitution.



The substitution rule aka the  $\beta$  rule

$(\lambda x. e_1) e_2 \xrightarrow{\beta} e_1[x := e_2]$       Greek letter beta

"steps to"      "e<sub>1</sub>, but with all occurrences of x in e<sub>1</sub> replaced with e<sub>2</sub>"

In lambda calculus, function application is left-associative

$F g x$  means "apply F to g, get back a value (which is a function), and then apply that function to x"

in other words:  $F g x$  is shorthand for  $(F g) x$  (it's not  $f(g x)$ !)

Let's try using the  $\beta$ -rule

$(\lambda x. e_1) e_2 \xrightarrow{\beta} e_1[x := e_2]$



What if we wanted functions that take more than one argument?

$\lambda x y. x$  (A function that takes two arguments and returns the first one)

$\lambda x. (\lambda y. x)$  (A function that takes an argument and returns a function that returns the original argument)



Remember the  $\beta$ -rule:

$(\lambda x. e_1) e_2 \xrightarrow{\beta} e_1[x := e_2]$



← the constant function that returns 5!

So  $\lambda x. (\lambda y. x)$  is a machine for creating constant functions. Cool!



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

$(F g) x$

$F (g x)$