Type Systems.

What is the purpose of types?

→ Ensure programs are "good".

e.g. – don't do nonsensical operations eg. 3 + True.

– don't crash.

– don't access memory they are not supposed to.

– don't leak secrets.

– always terminate.

– etc.

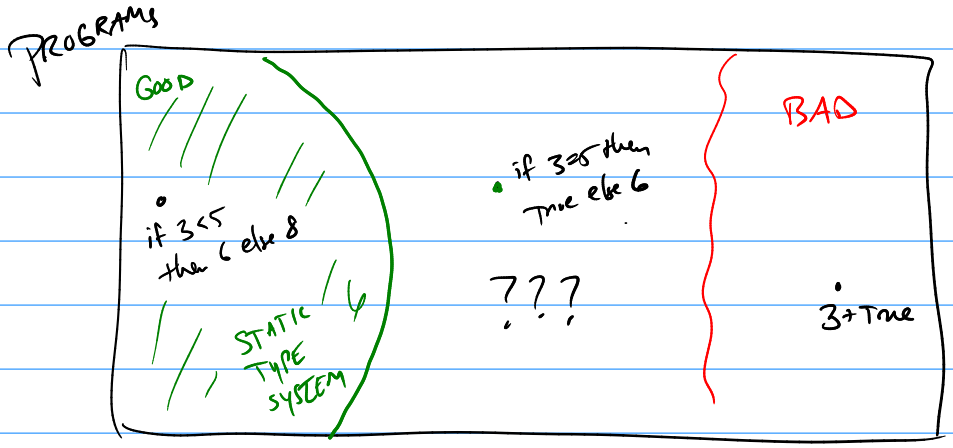
Static vs. dynamic types.

– In a dynamic type system, values have types and are checked at runtime.

– In a static type system, program expressions have types and are checked before runtime.

e.g. (if 3 = 5 then True else 6) + 8

↑ Eg of a program that is ok in a dynamic type system but rejected w/ static type system.



Arith language

numbers ↓ variables ↓

<e> ::= n | x | False | True
| <e> <op> <e>
| let x = <e> in <e>
| if <e> then <e> else <e>

<op> ::= + | − | * | < | =

<type> ::= Int | Bool.

inference rules

$$\frac{A \quad B \quad C}{D}$$

"if A and B and C, then D".

Goal: write down rules for deciding what types our programs have

<e> : <type> means "if you run <e>, it will result in a value of type <type>."

$$\frac{}{n : \text{Int}}$$ "Any number literal has type Int."

$$\frac{}{\text{False} : \text{Bool}} \qquad \frac{}{\text{True} : \text{Bool}}$$

$$\frac{e_1 : \text{Int} \quad e_2 : \text{Int}}{e_1 + e_2 : \text{Int}}$$ + similar for −, *.

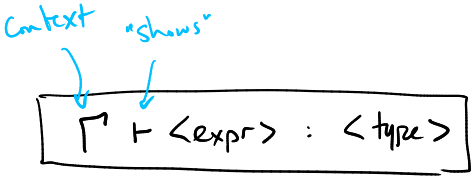
$$\frac{e_1 : \text{Int} \quad e_2 : \text{Int}}{e_1 < e_2 : \text{Bool}}$$

could also add a rule for comparing Booleans if we want.

Similar rules for =.

$$\frac{e_1 : \text{Bool} \quad e_2 : t \quad e_3 : t}{\text{if } e_1 \text{ then } e_2 \text{ else } e_3 : t}$$

To give types for variables, we must add a context which is a map from variable names to types.
We use the symbol Γ.



"Under the assumptions Γ, <expr> has type <type>".

e.g. $\{x : int, y : bool\} \vdash x + 3 : int$.

$$\frac{\Gamma[x] = t}{\Gamma \vdash x : t} \qquad \frac{\Gamma \vdash e_1 : Int \quad \Gamma \vdash e_2 : Int}{\Gamma \vdash e_1 + e_2 : Int.}$$

$$\frac{\Gamma \vdash e_1 : t_1 \quad \Gamma, x : t_1 \vdash e_2 : t_2}{\Gamma \vdash \text{let } x = e_1 \text{ in } e_2 : t_2}$$

let x = 5 in (x + (let x = 12 in x + 9))