

Evaluation

Assignments	50	at least 6
Quiz	20	
Final Exam	30	

COPYING IS FATAL

Haskell

Functional Programming language

DECLARATIVE

What to do

vs

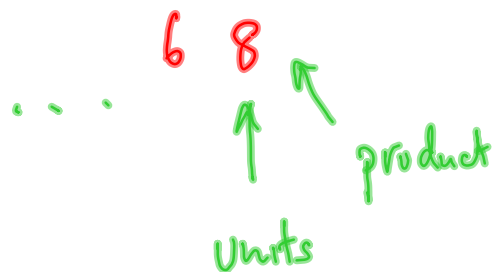
IMPERATIVE

How to do it

C, C++, Java, Python, ...

$$\begin{array}{r}
 \overset{2}{2} \overset{\text{Carry}}{\textcircled{2}} 3 6 7 \\
 \times \quad 4 9 4 \\
 \hline
 \end{array}$$

← this process is imperative programming



Need a language to describe such

Intermediate values generated processes
 Significance/interpretation is determined by position on page

We can't use positional clues in a programming language!

Need names to refer to values

e.g. use "carry" to store current carry value when multiplying

Assignment

Repeatedly assign fresh values to the same name

Basic step in an imperative prog language

Assignment statement

y Name \leftarrow Value \leftarrow an expression
 $6 * x + 3$

In Python

$y = 6 * x + 3$

$x = x + 1$
Reassign to x its
current value + 1

$$x = 2 * x + 4 * x$$

Both refer to current value of x

Data type

Int, Float, Bool, Char, ..

In Haskell, every name/variable has a fixed type

In Python

Values have types

Names have no type

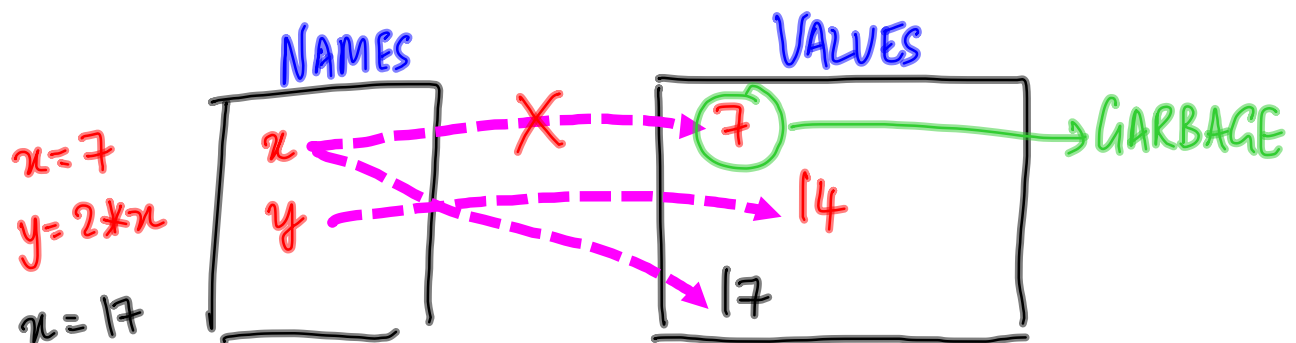
Type of a name is derived from its value

$x = 6$
 $\therefore y = 2 * x$ ✓

$x = \text{True}$
 $y = 2 * x$ ✗ x is not of the right type

Can freely assign values to names,

but, when a name appears in an expression
(i.e. the name is "used") it must
have a "sensible" value



Basic scalar values in Python are similar to Haskell

int, float → arbitrary precision

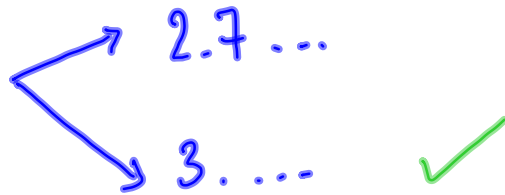
bool True, False

No Char type, only String

Write $f(x)$ not $f\ x$ as in Haskell

$\sin(x)$

Python



`print x` OK in 2.7, not in 3..

`print (x)` Must have () in 3...

Operations:

Arithmetic

+, *, -, /

↙ Float division
even if both args
are int

 $3/2 \rightarrow 1.5$

// div of Haskell

% mod of Haskell

** exponentiation $x \uparrow y \rightsquigarrow x^y$

Comparison

<, >, <=, >=, ==, !=

Boolean

and (&), or (!), not

Collective values

List Like Haskell [6,7,4,3]
 Unlike Haskell [6,7.2,True,3.5]
 [6, [2,4], 3.6]

$l = [6, 7, 7.5]$

Access elements by position $l[0], l[1], l[2]$