## Procedural programming

Explicit sequence of steps to compute

## Python syntax

Manipulate values

Assign names to keep track of them

### Assignment

name = expression

not equality

$n = m+5$

$n = n+5$

$n := m+5$ $\quad$ $n \leftarrow m+5$

Haskell : type    Int, Float, Bool, Char,

Python

Values have types

Names derive type from current value

n = 5 ← n is Int

n = False ← now n is Bool

In contrast to C, C++, Java ...

```
int n;
n = False;   ← error
```

Names must be assigned before they are used

$x = y + 1$ $y$ has no value, error

Dynamic type vs Static

Strong type vs Weak

$x = $ False

$y = x + 2$ X No + for False

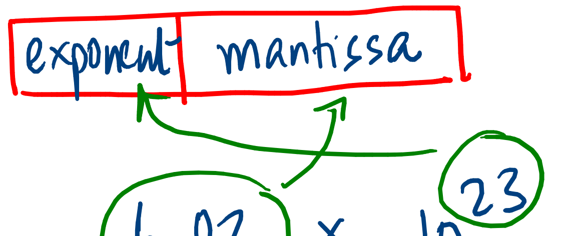
Not quite! False is 0.
Clarify later.

## Basic types Integer, Float

+, -, *, / always "real" division

% mod

// div



$6.02 \times 10^{23}$

$.602 \times 10^{24}$

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

Boolean: True, False

or

and

not( )

No Char type - only strings

String: "hello" 'world'

Triple quote ''' my ''' """ hi """

Behaves like a list, but is not a list of Char
(because there is no Char)

List: [ , , ] Need not have uniform underlying type

["hello", 2, [3, False]]

Confusion between lists & arrays

$$
\begin{array}{lccccc}
l = & [2, & 3, & 8, & 9, & 10] \\
 & 0 & 1 & 2 & 3 & 4 \\
 & -5 & -4 & -3 & -2 & -1
\end{array}
$$

$l[3] \rightsquigarrow 9$

$l[0]$ to $l[4]$ are valid positive

$l[-1]$ to $l[-5]$ are also valid

$l[5]$ — error

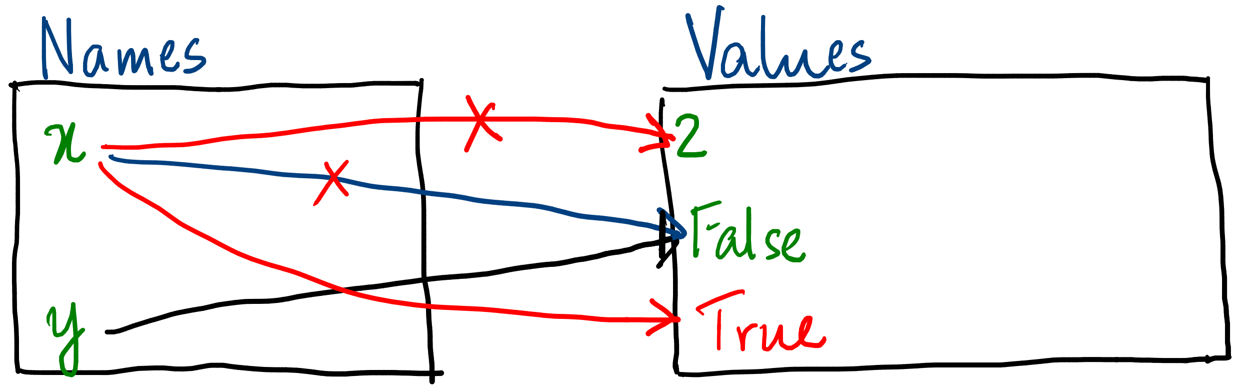
$l[-6]$ /

s = "hello"
0 1 2 3 4

s[3] = ? "l"

Lists

| | |
|---|---|
| len(l) | length |
| l1 + l2 | concatenation (++) |
| l1 * 3 | $\sim$ l1 + l1 + l1 |



$x = 2$

$x = \text{False}$

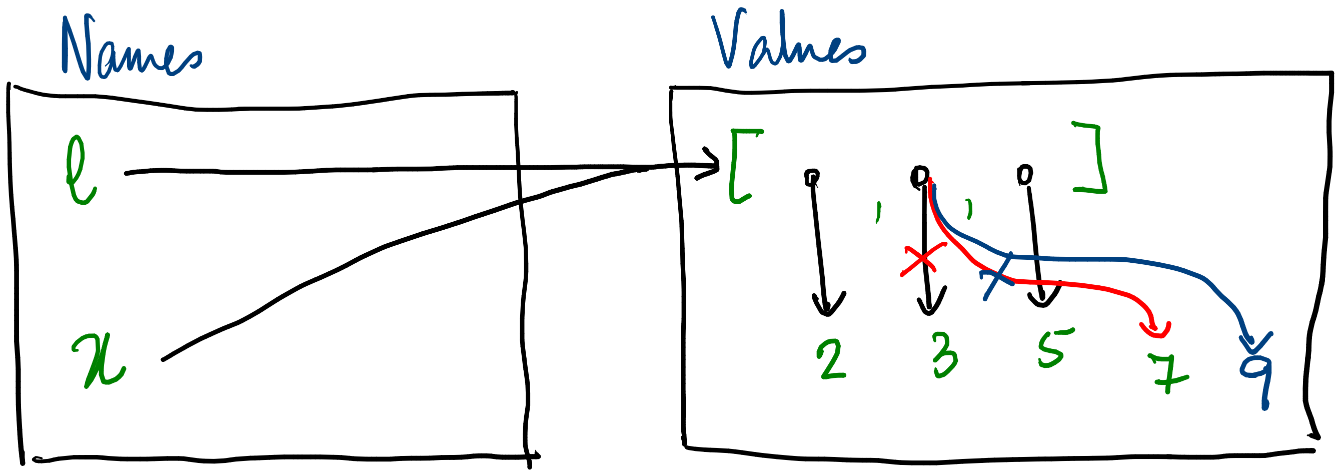
$y = x$

$x = x \text{ or True}$

"Immutable"

Does not update $y$

Lists are different "Mutable"



l = [2, 3, 10]

l[1] = 7

x = l

x[1] = 9

l[1] $\rightsquigarrow$ ? 9 !

Strings are <u>not</u> mutable

x = "hell"

x[3] = "p" ✗