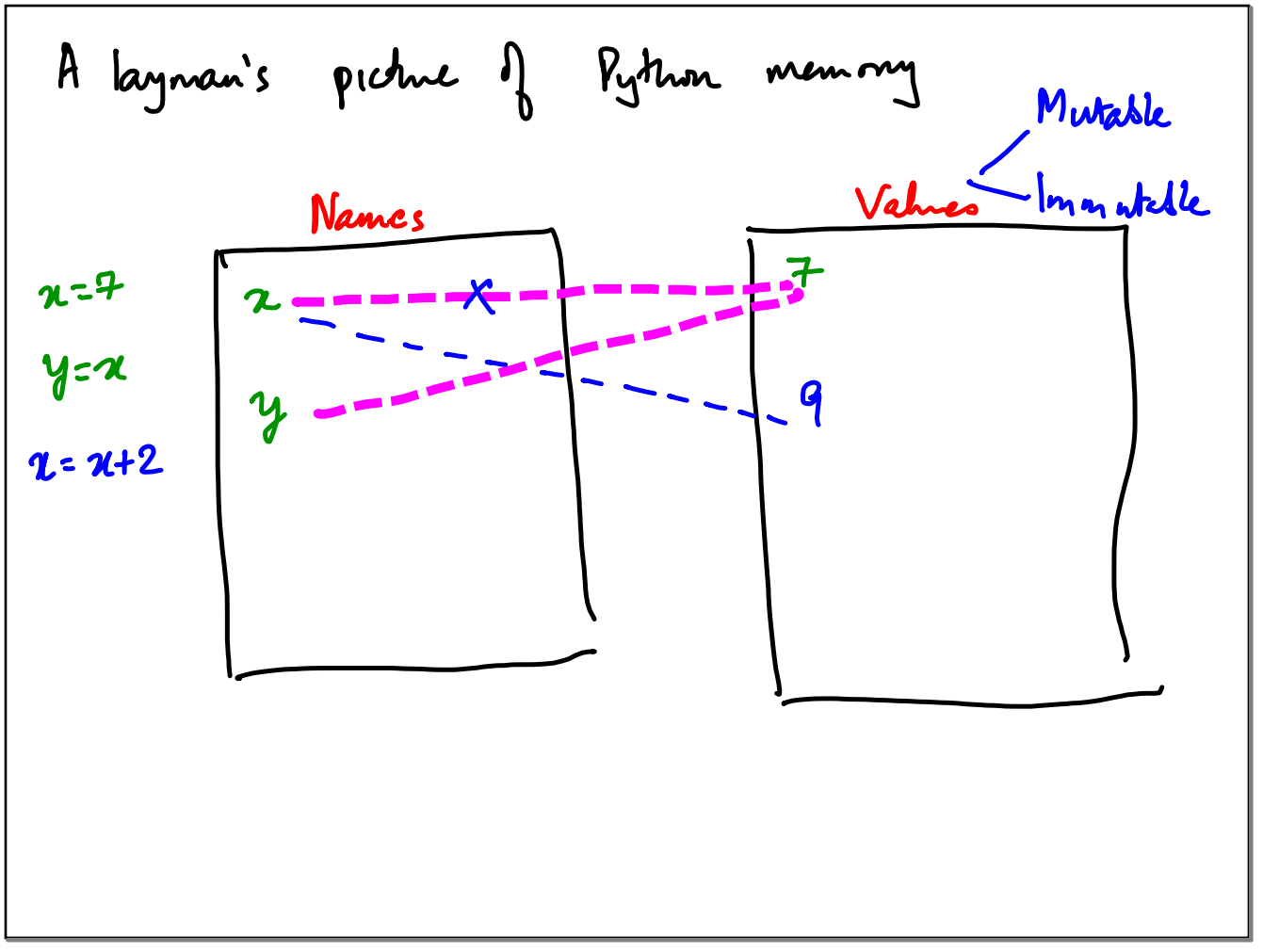
Persistent variables in classes

Names, values, types etc

No declarations ⇒ names don't carry type

Type of name is given by current value

Dynamic typing But strict


A layman's picture of Python memory
Mutable
Immutable
Names
Values
x=7
y=x
x=x+2
x
y
7
9

Immutable values

- Numbers
- Boolean
- Strings
- Tuples

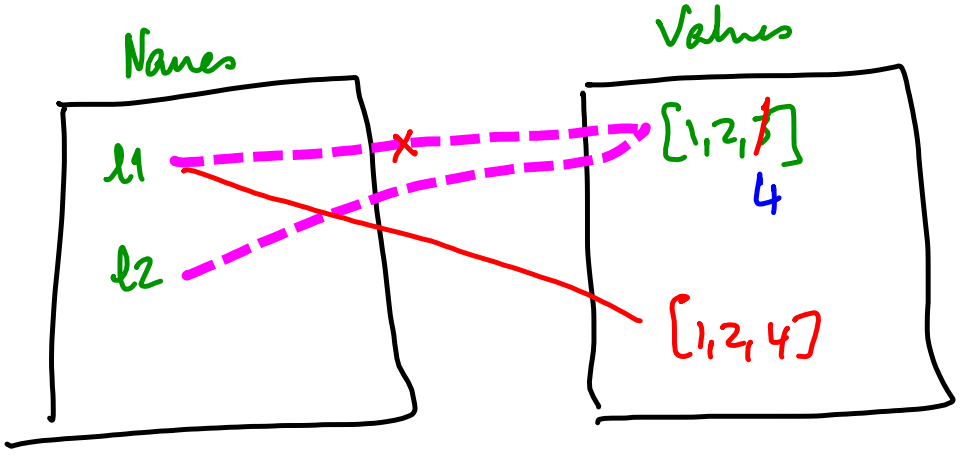
s = "hello" s[4] => "o"

hell! x s[4] = "!"

s = s[0:4] + "!"

Mutable

- Lists
- Dictionaries
- Objects via classes

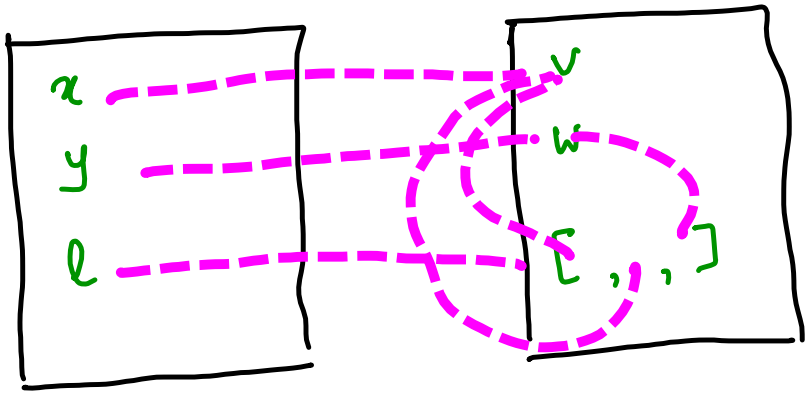



```
l1 = [1,2,3]
l2 = l1
l1[2] = 4
```

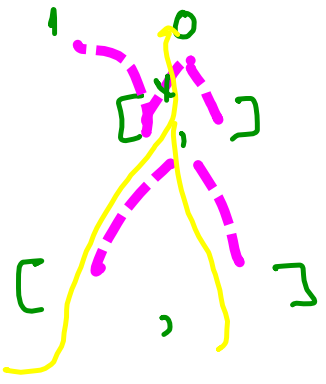
But

```
l1 = l1[0:2] + [4]
```

l = [x, x, y]

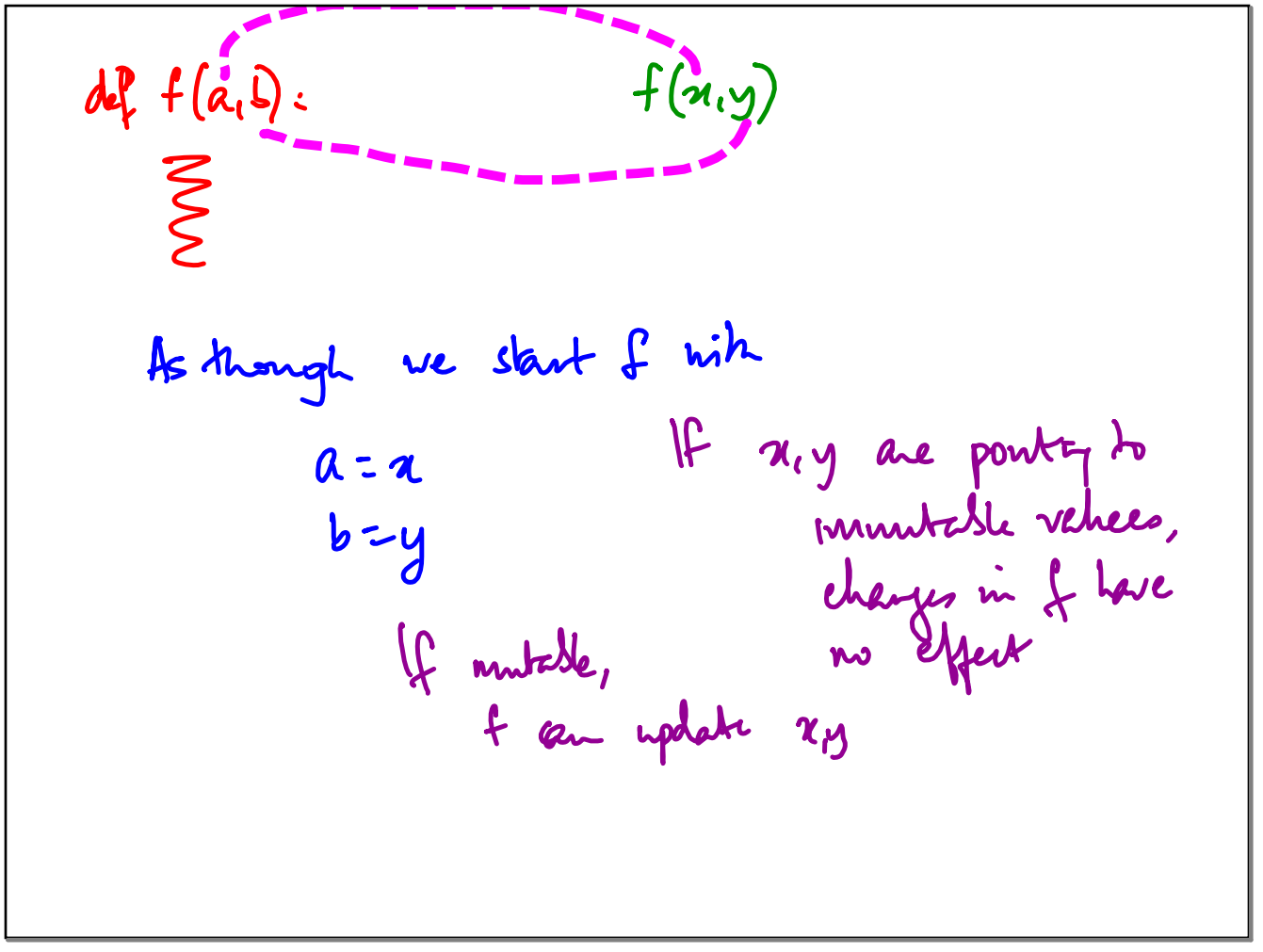


l = [[0]*2]*2



l[0][0] = 1

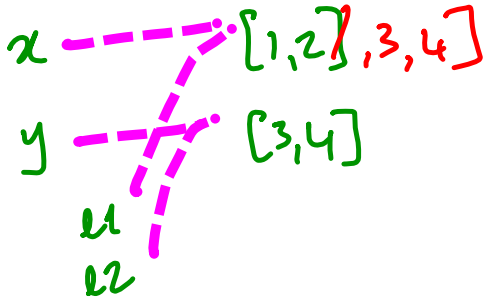
=> l[1][0] is the same 1
l[0][1] is still (immutable) 0


def f(a,b):
f(x,y)
As though we start f with
a = x
b = y
If mutable,
f can update x,y
If x,y are pointers to
immutable values,
changes in f have
no effect

```
def myextend(l1,l2):
    l1.extend(l2)
```

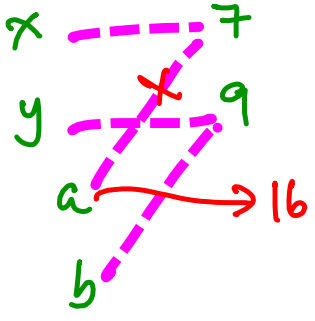
```
x = [1,2]
y = [3,4]
myextend(x,y)
```

x ⇝ [1,2,3,4]



```
def f(a,b):
    a = a+b
```

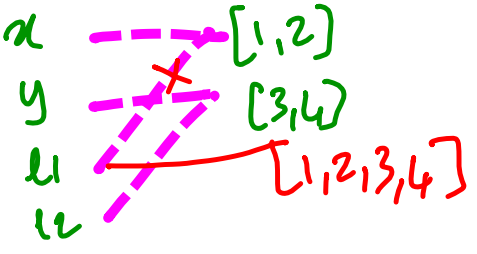
```
x = 7
y = 9
f(x,y)
```



Can change contents of a mutable object but not what it points to via a function

f(x,y)

```
def extendplus(l1,l2)
    l1 = l1 + l2
```



Updating lists

list[i] = new in place

list[i:j] = new in place

l[len(l):] = newlist l.extend(newlist)

l.append(v) l.extend(list)

l += l+ . . new object