Quote assignment

Catchup: globals in Python

Lambdas in other languages

What quote does // (define a 3)

'a ↩ (quote a)

↓

a

a

↓

3

(quote ——————) → ——————

does not evaluate ↑

(quote (+ 1 2 3))

→ (+ 1 2 3)

Normally, w/ a Scheme expression

(define x (+ 1 2))

↓ eval

3

the does define

(if #t (+ 1 7) 2)

↑ eval

(quote (define 3 car))

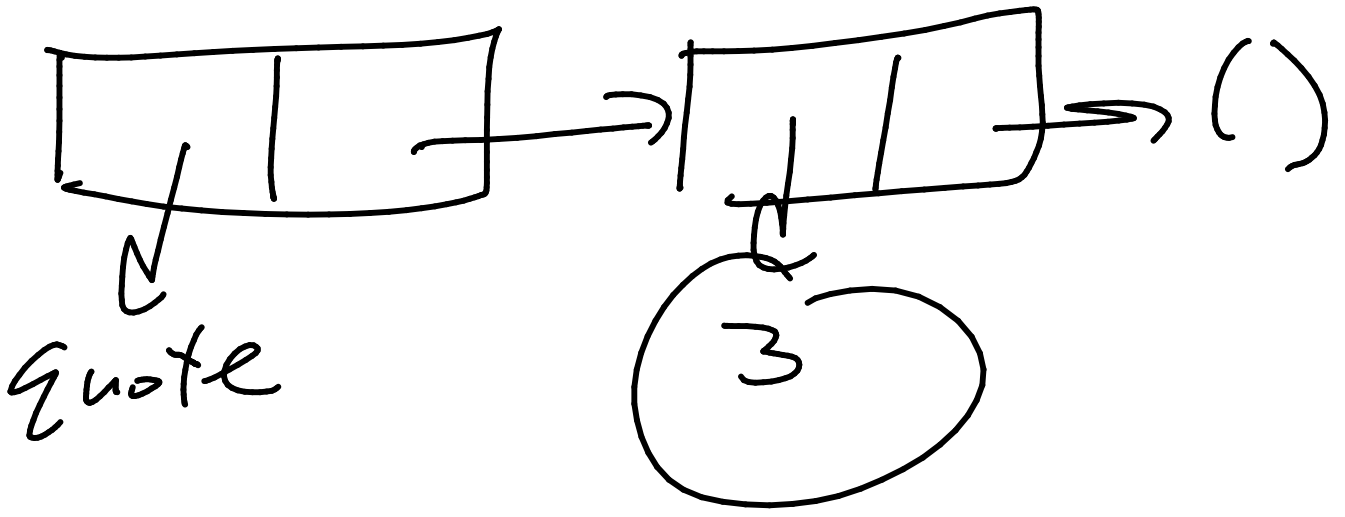
↳ (define 3 car)

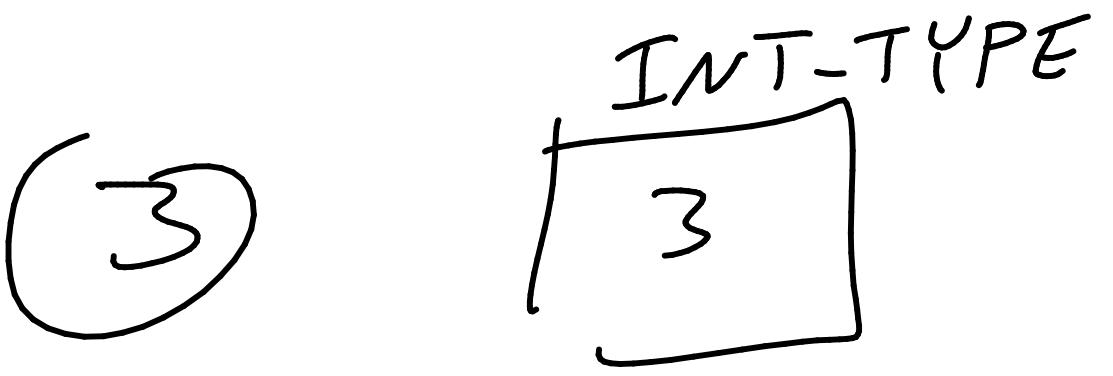
Assignment: implement quote

The good news:
quote assignment takes very little code
to accomplish

The "watch out" news
It's easy to get it wrong and
compensate in your printing code

(quote 3)





Complexity is your enemy.
You will write code on easy cases,
and fails on hard cases.

```
(if #t (let ((x 3)) x)
       (quote 7))
```

What if
if works w/ basic examples?
let - - - - - -
quote - - - -
but the one above doesn't?
Bad idea is to carve out exceptions
e.g. "Oh, let's do a special case
for let inside if"

```python
x = 1

def doit():
    x = 2

doit()
print(x)
```

This creates a new local variable.

```python
x = 1

def doit():
    print(x)

doit()
```

This refers to whichever x I have access to (Python does static scoping, so I look outside function, and find the global)

In Python, if you read (see, look at) a variable in a function, you find the var wherever it may exist subject to scoping rules.

- if you write to it, get a new local variable
  - unless you indicate it to be global

---

most other langs use a declaration

```
int x;
val x:Int
(let ((x ...)))
```

(explicitly create a variable w/ a keyword to indicate)

```
x = 1

def doit():
    global x
    x = 2

doit()
print(x)
```