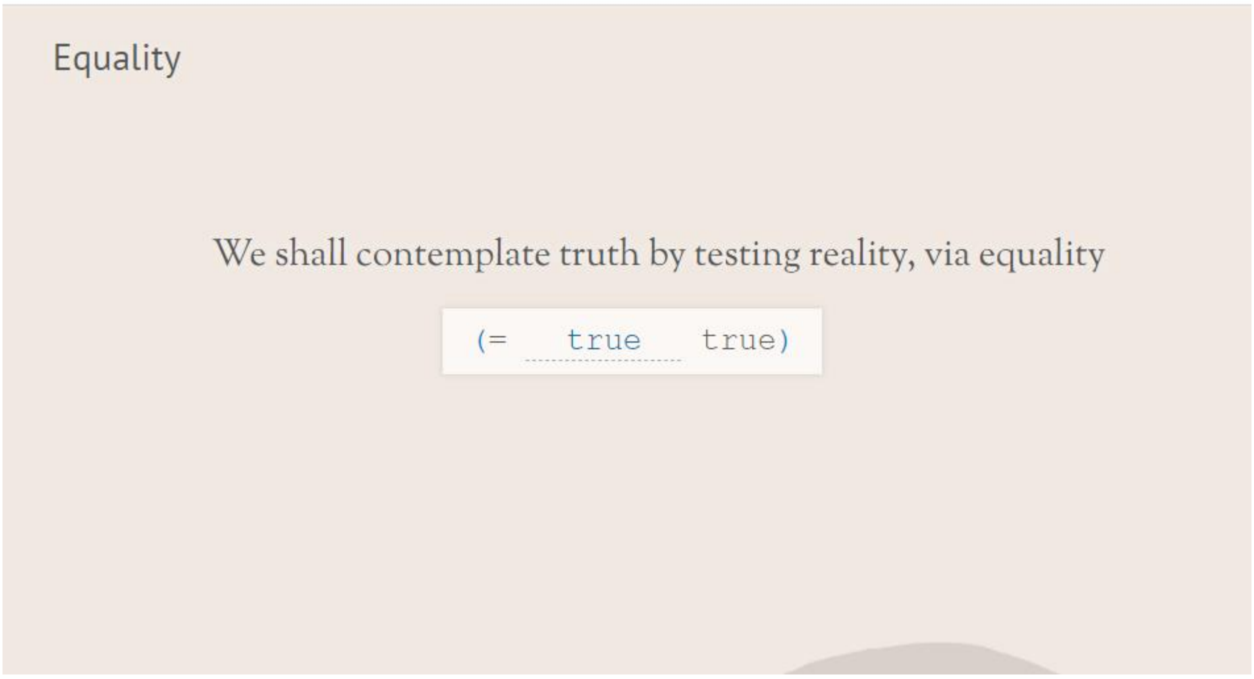
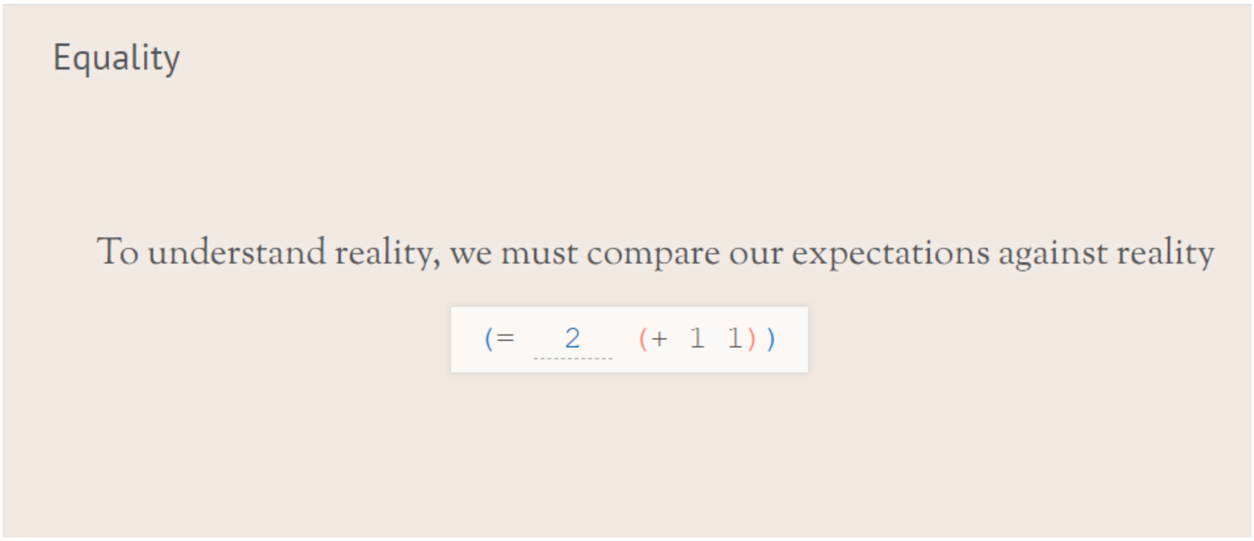


Equality

We shall contemplate truth by testing reality, via equality

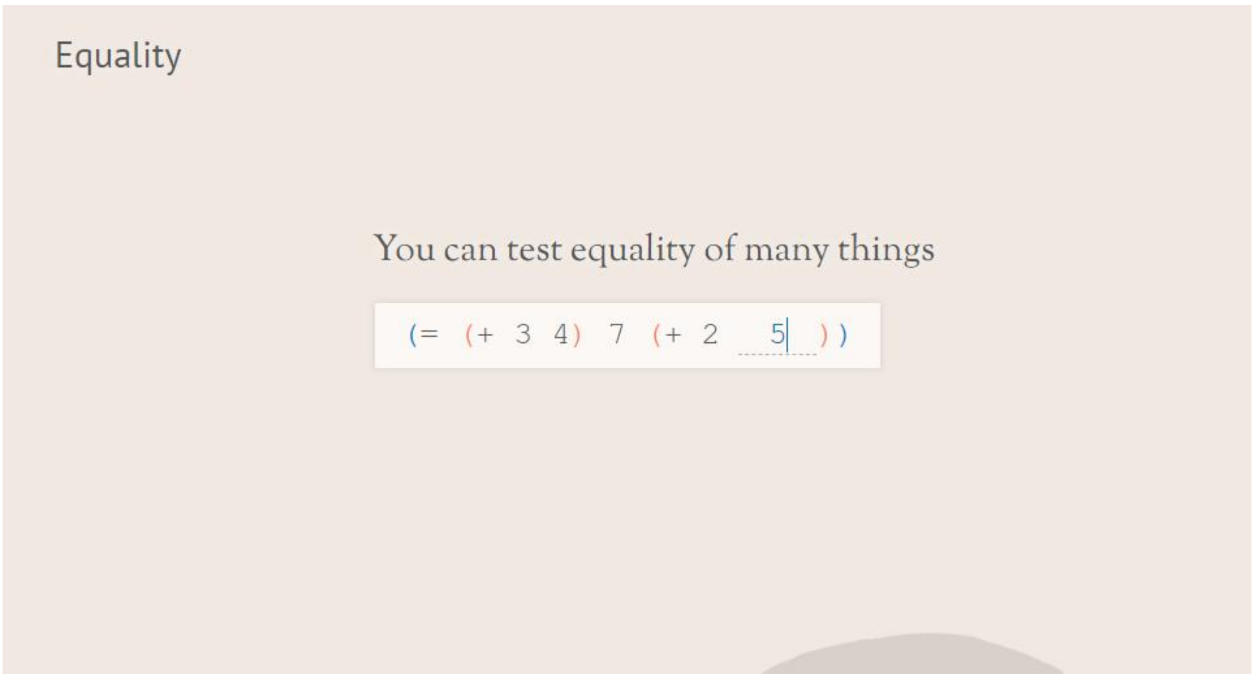
```
(=  true  true)
```



Equality

To understand reality, we must compare our expectations against reality

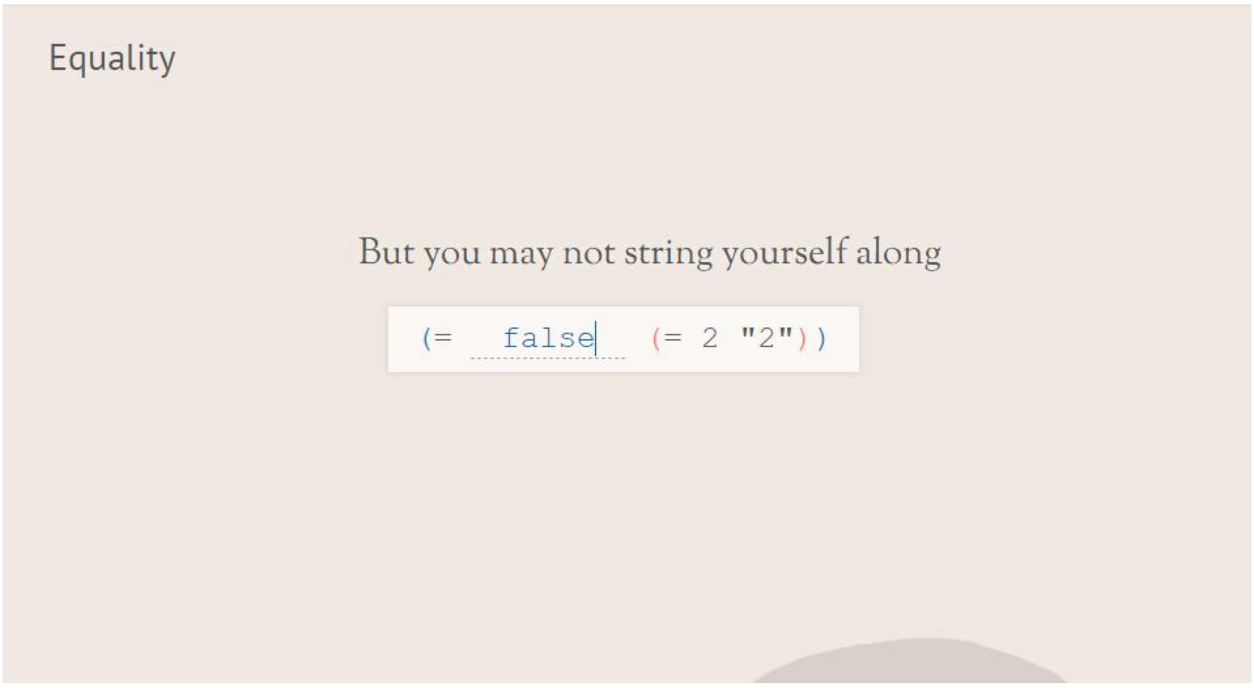
```
(=  2  (+ 1 1))
```



Equality

You can test equality of many things

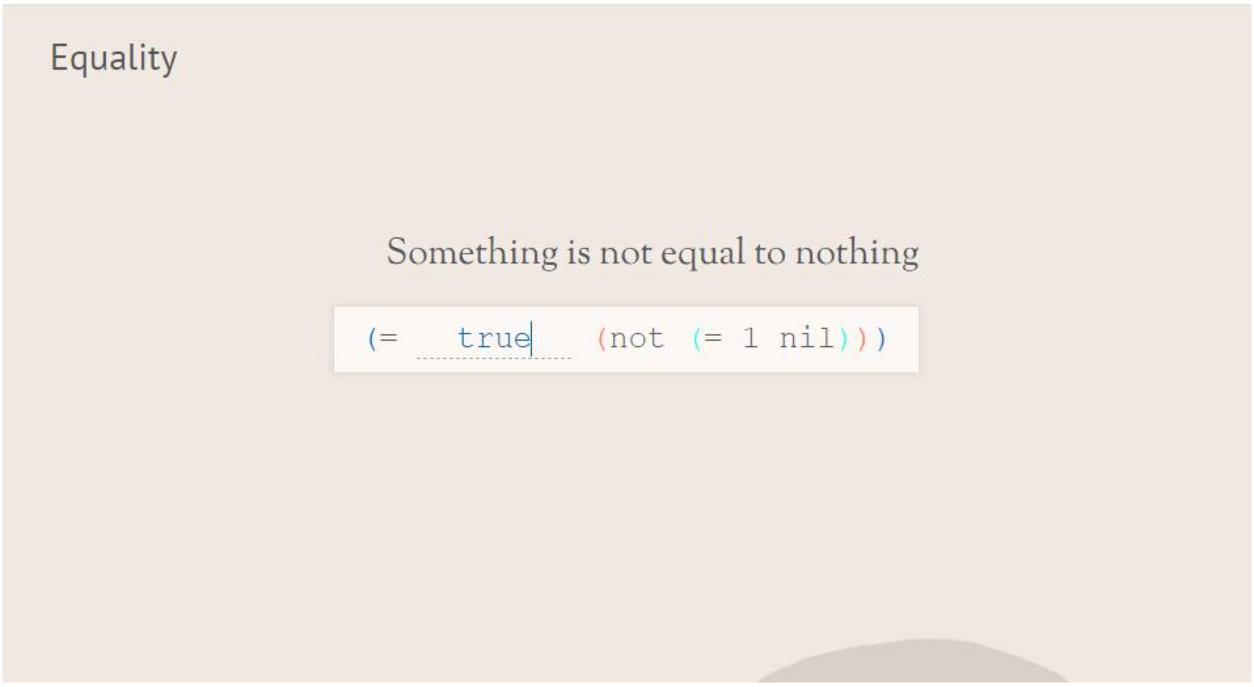
```
(= (+ 3 4) 7 (+ 2  5 ))
```



Equality

But you may not string yourself along

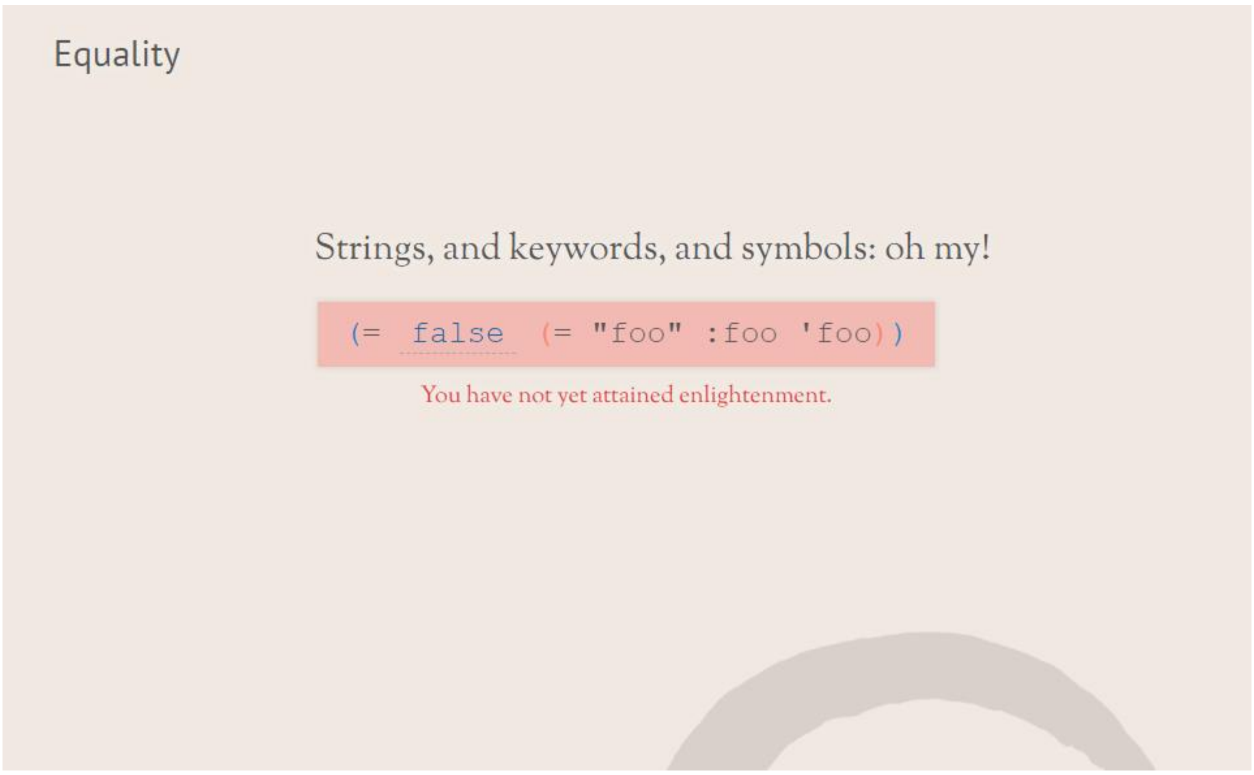
```
(= false (= 2 "2"))
```



Equality

Something is not equal to nothing

```
(= true (not (= 1 nil)))
```

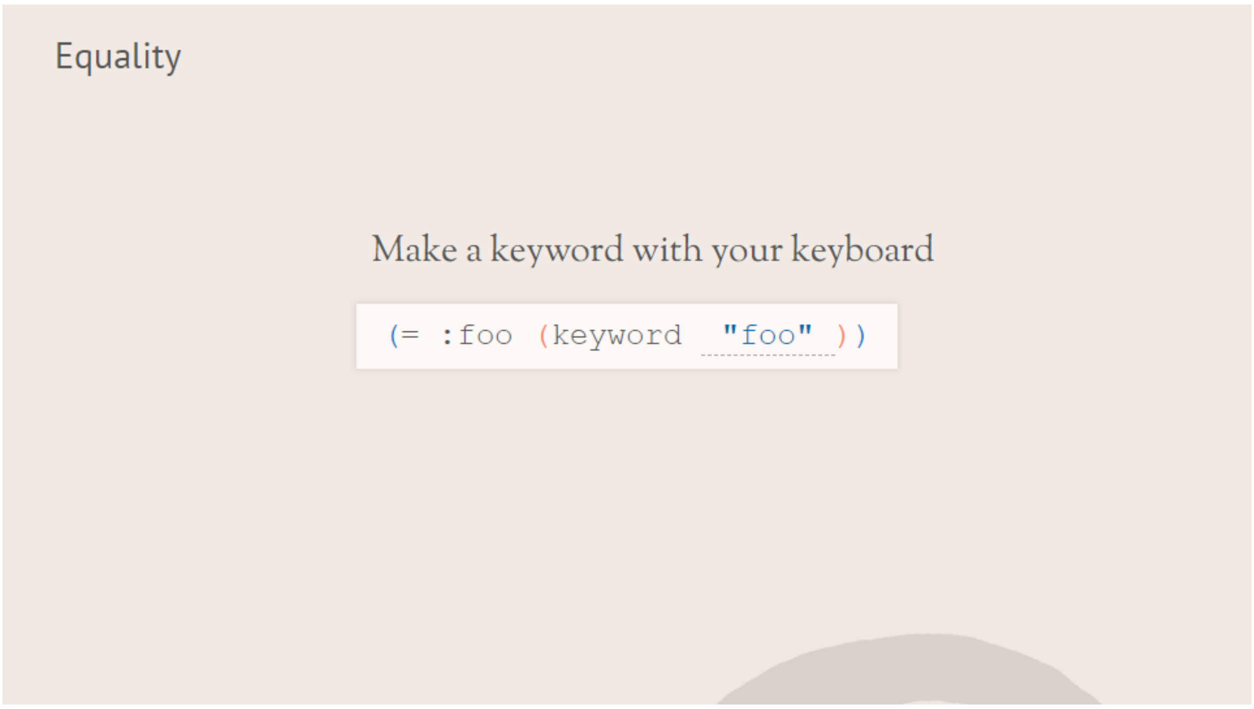


Equality

Strings, and keywords, and symbols: oh my!

```
(= false (= "foo" :foo 'foo))
```

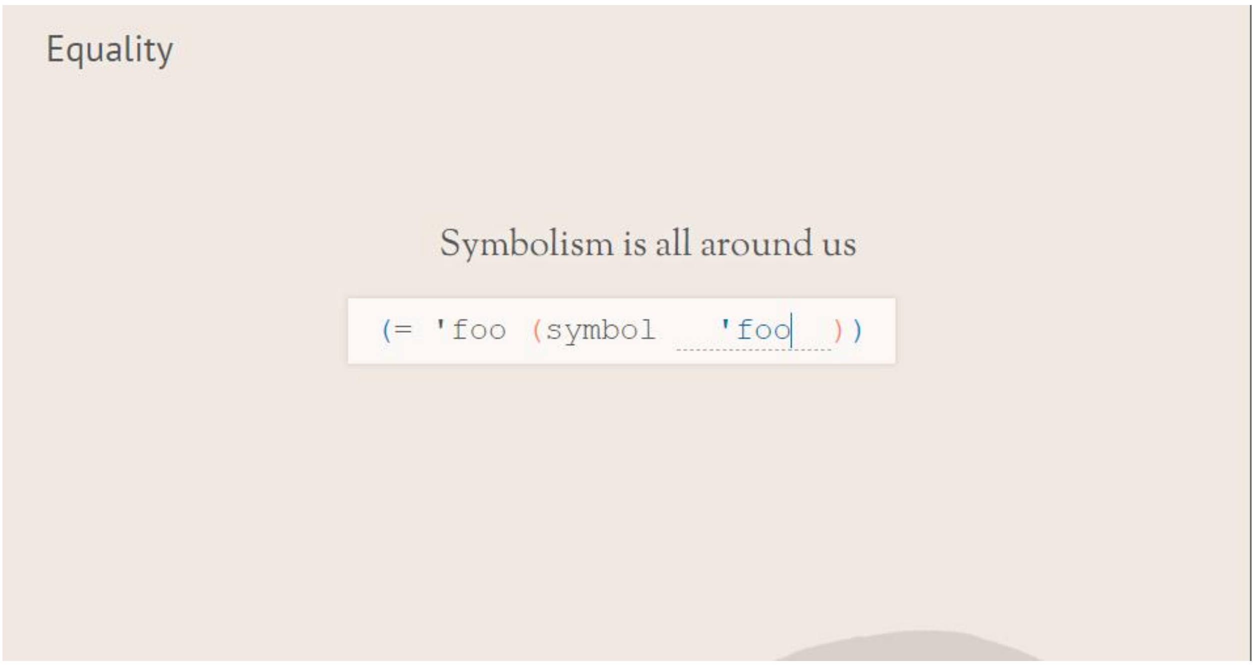
You have not yet attained enlightenment.



Equality

Make a keyword with your keyboard

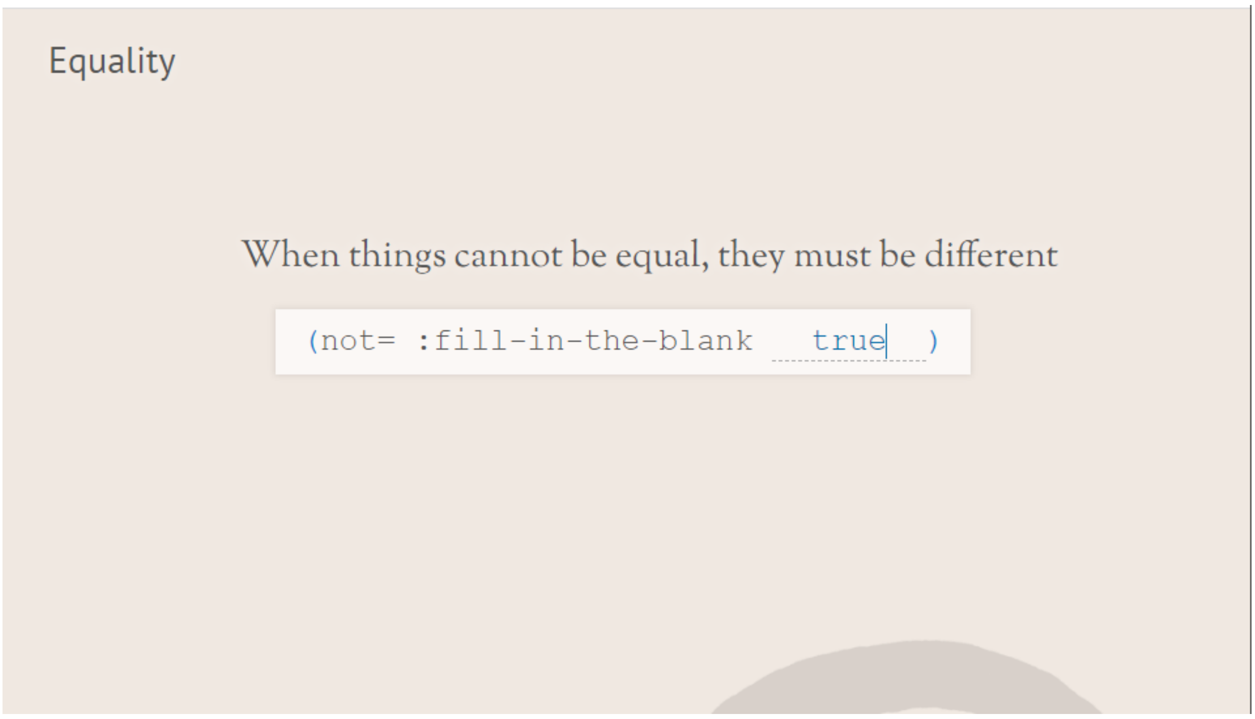
```
(= :foo (keyword "foo" ))
```



Equality

Symbolism is all around us

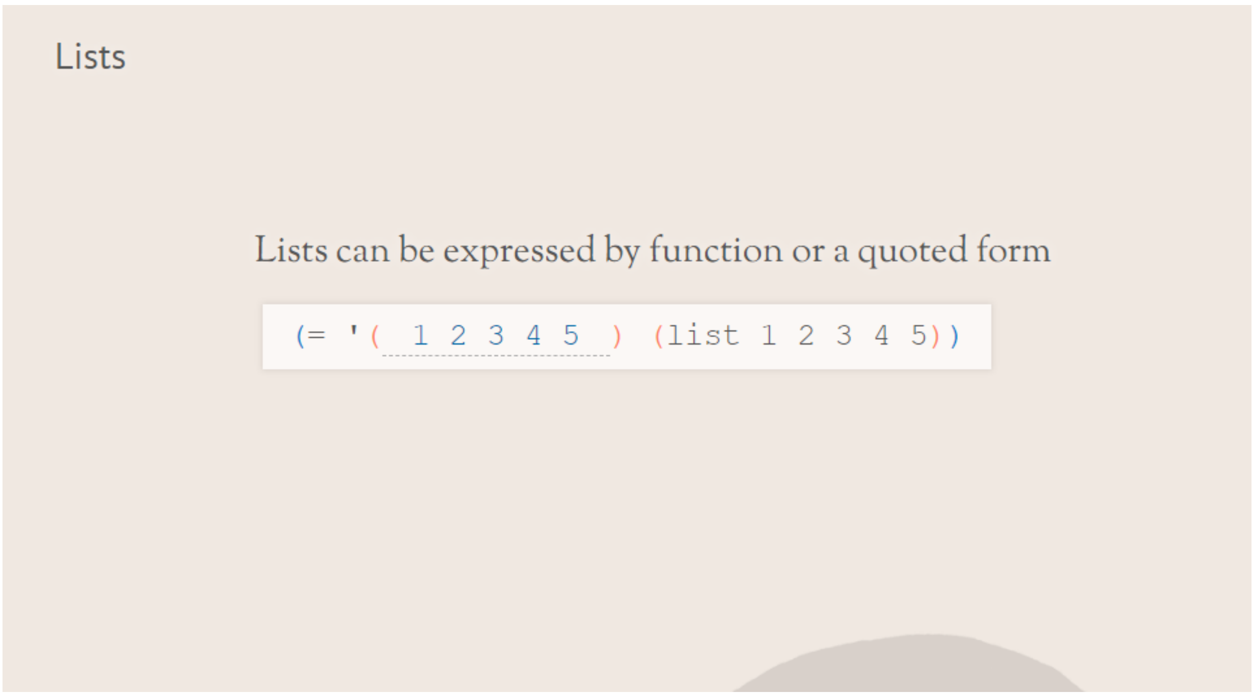
```
(= 'foo (symbol 'foo))
```



Equality

When things cannot be equal, they must be different

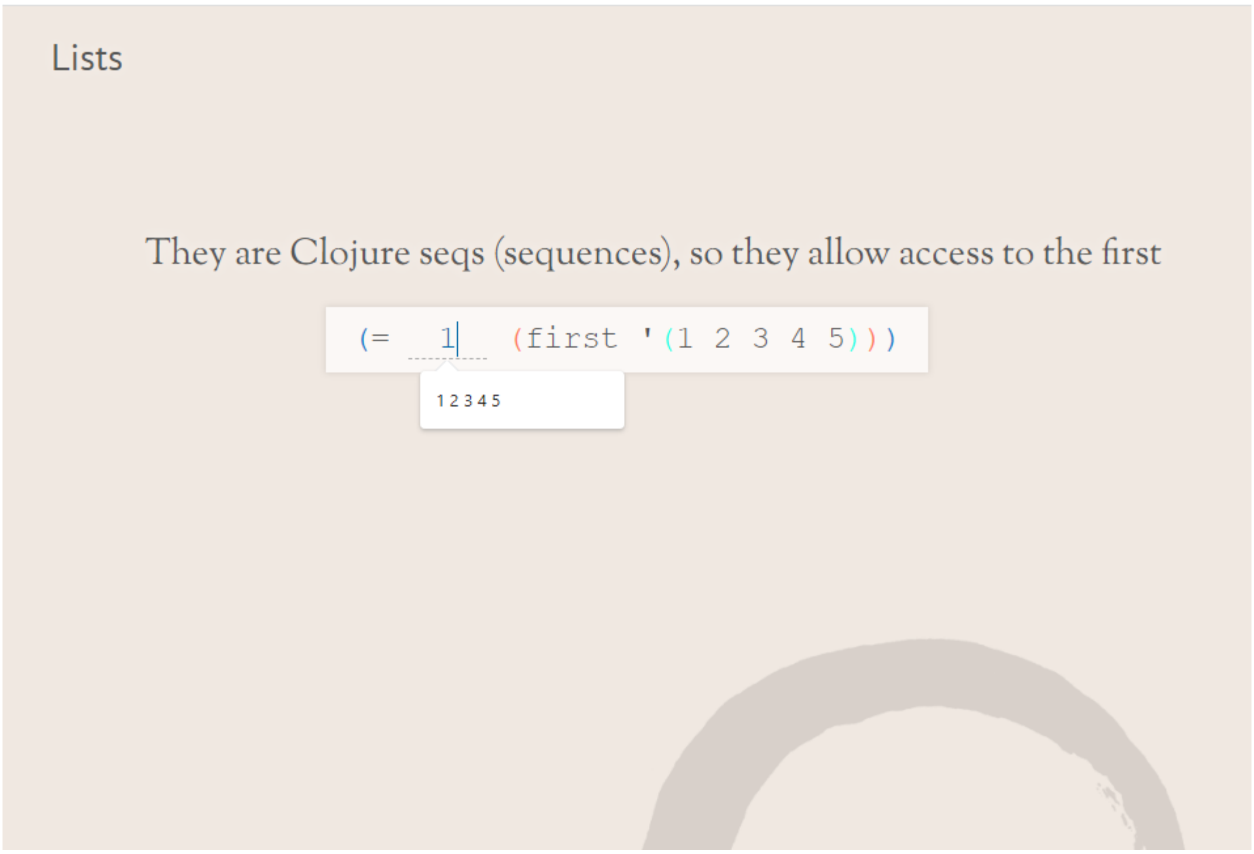
```
(not= :fill-in-the-blank true)
```



Lists

Lists can be expressed by function or a quoted form

```clojure
(= '( 1 2 3 4 5 ) (list 1 2 3 4 5))
```

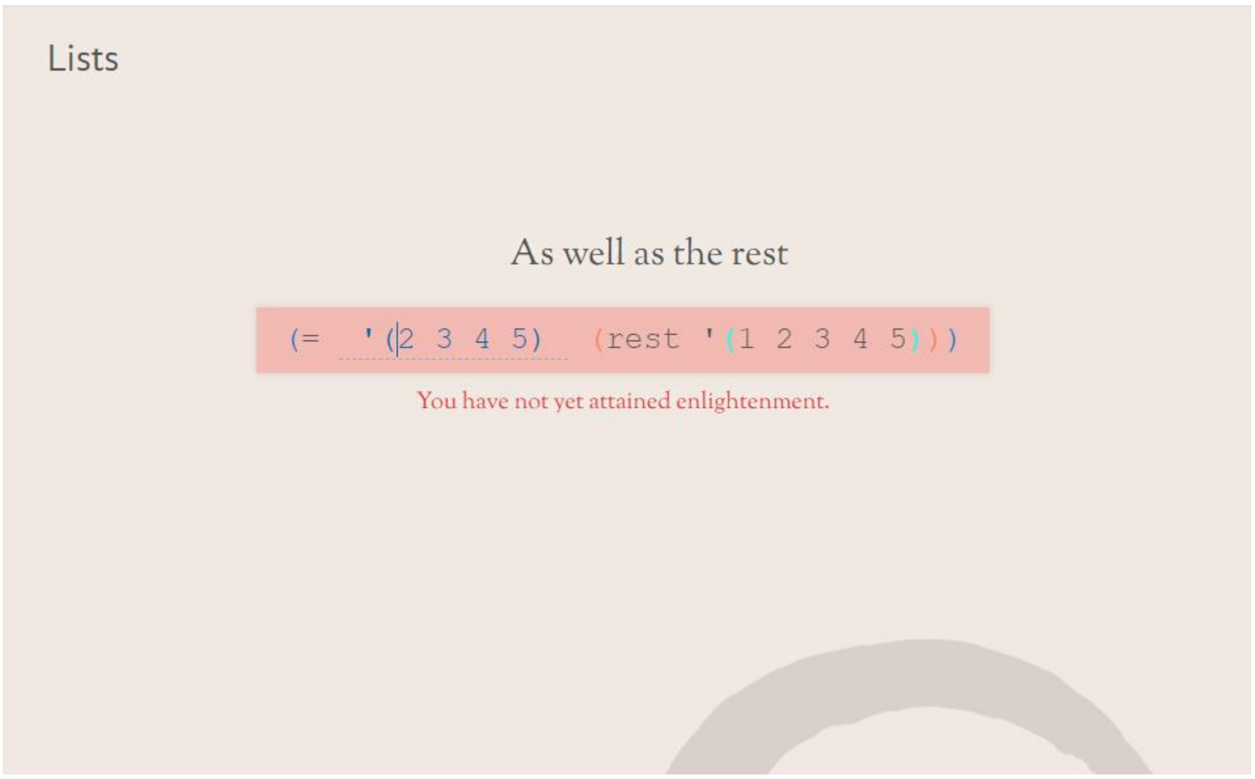


Lists

They are Clojure seqs (sequences), so they allow access to the first

```clojure
(= 1 (first '(1 2 3 4 5)))
```

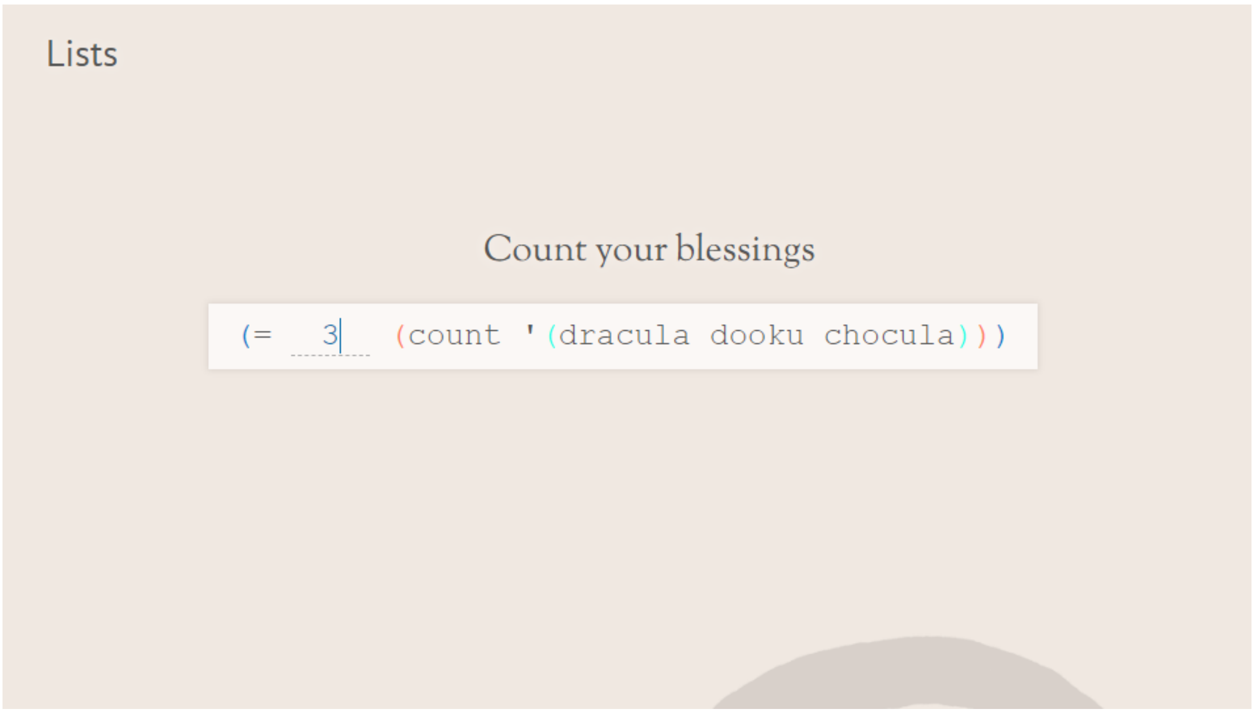
```
1 2 3 4 5
```


Lists

As well as the rest

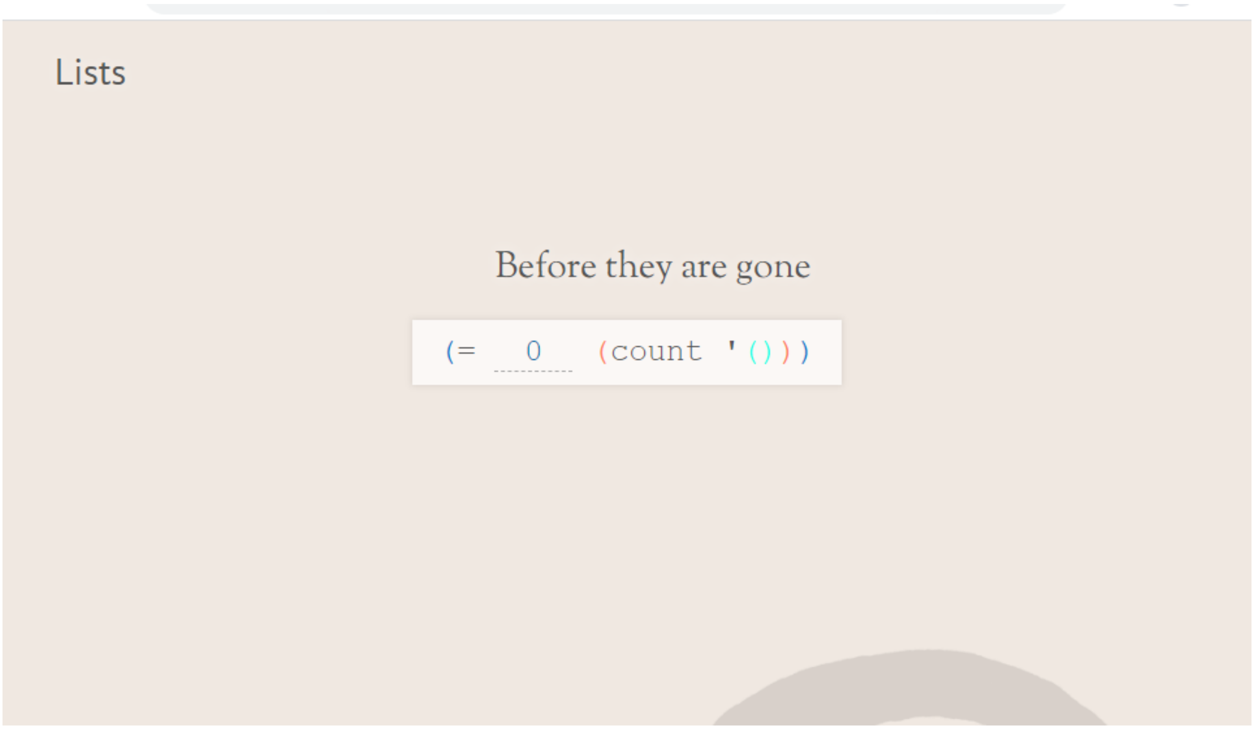
```
(=  '(2 3 4 5)  (rest '(1 2 3 4 5)))
```

You have not yet attained enlightenment.


Lists

Count your blessings

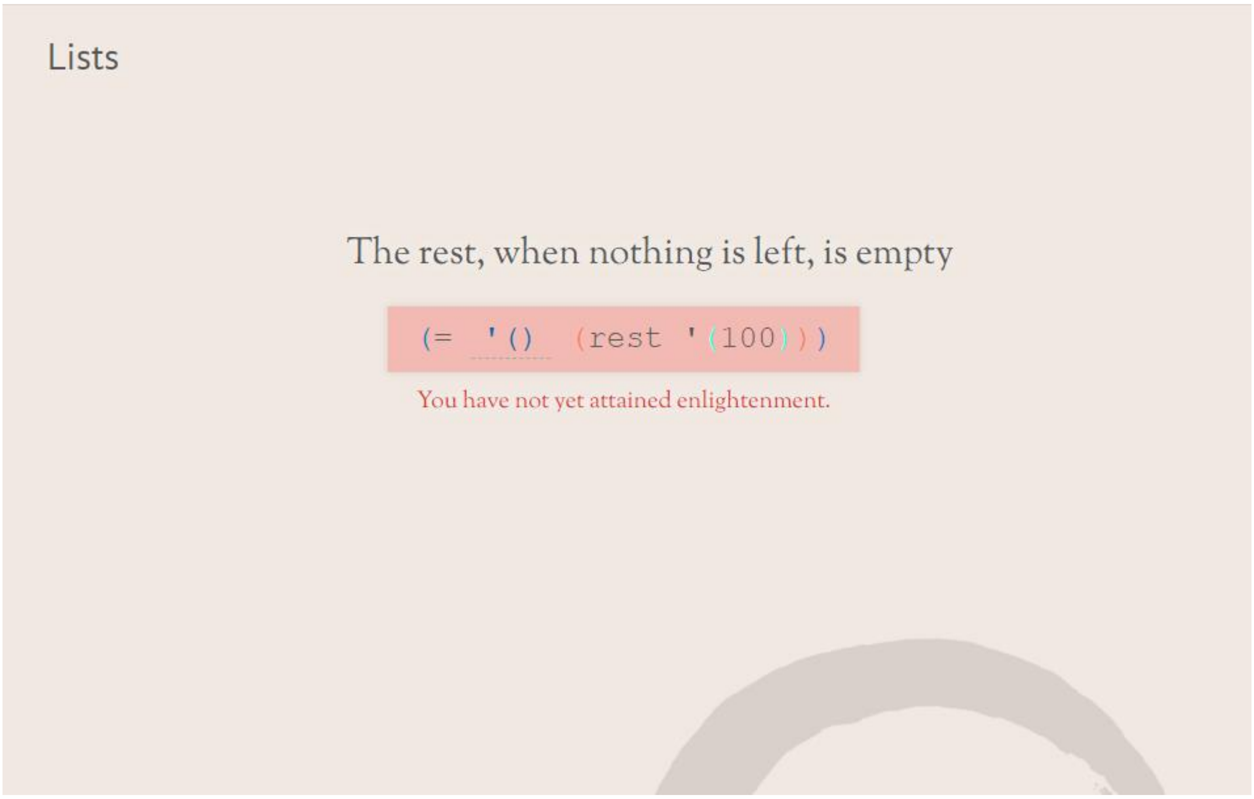
```
(=  3  (count '(dracula dooku chocula)))
```



Lists

Before they are gone

```
(= 0 (count '()))
```

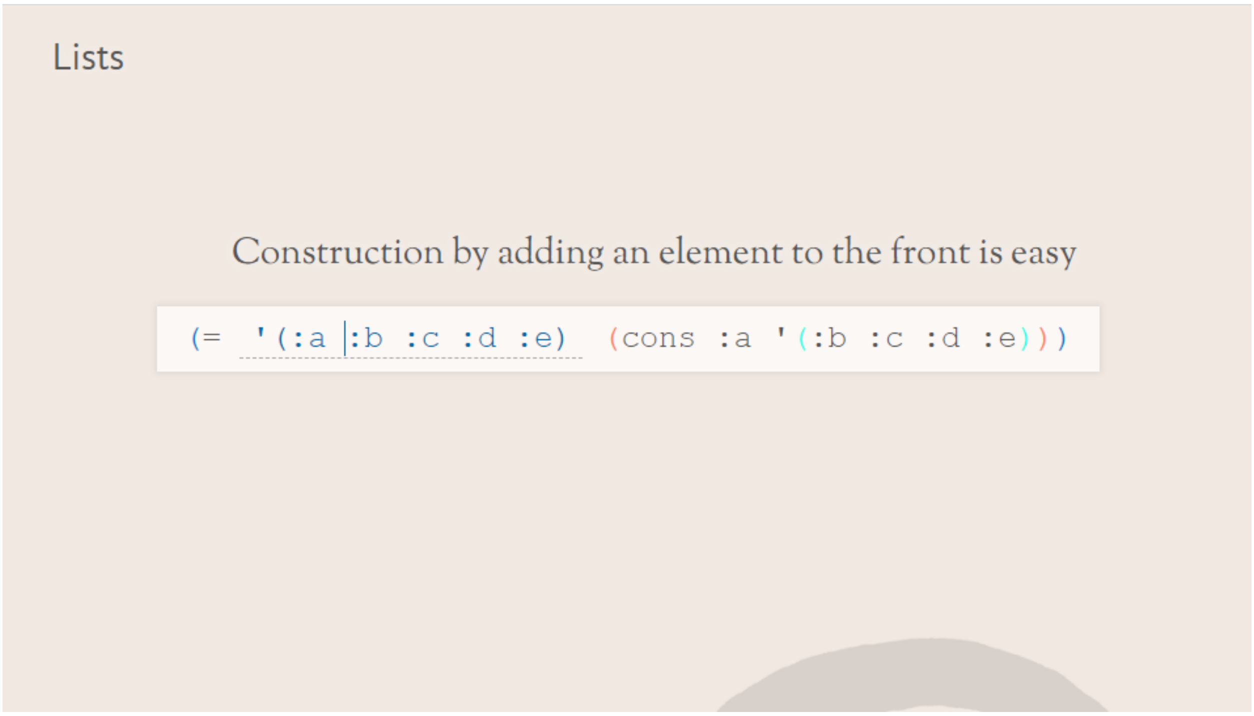


Lists

The rest, when nothing is left, is empty

```
(= '() (rest '(100)))
```

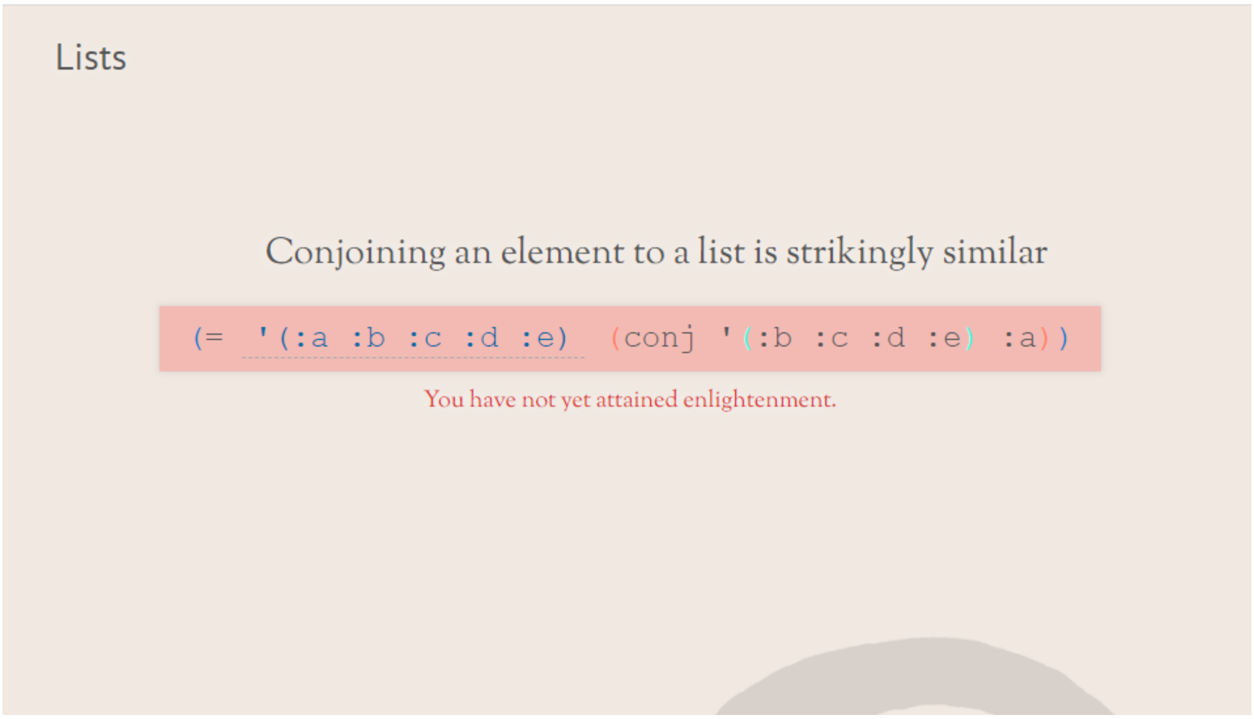
You have not yet attained enlightenment.



Lists

Construction by adding an element to the front is easy

```
(=  '(:a :b :c :d :e)  (cons :a '(:b :c :d :e)))
```



Lists

Conjoining an element to a list is strikingly similar

```
(=  '(:a :b :c :d :e)  (conj '(:b :c :d :e)  :a))
```

You have not yet attained enlightenment.