Things you should know about cons, pairs and lists

The *pair* or *cons cell* is the most fundamental of Scheme's structured object types.

A **list** is a sequence of **pairs**; each pair's cdr is the next pair in the sequence.

The cdr of the last pair in a **proper list** is the empty list. Otherwise the sequence of pairs forms an **improper list**. I.e., an empty list is a proper list, and and any pair whose cdr is a proper list is a proper list.

An improper list is printed in **dotted-pair notation** with a period (dot) preceding the final element of the list. A pair whose cdr is not a list is often called a **dotted pair**

`cons` vs. `list`: The procedure `cons` actually builds *pairs*, and there is no reason that the cdr of a pair must be a list, as illustrated on the next page.

The procedure `list` is similar to `cons`, except that it takes an arbitrary number of arguments and always builds a proper list.

E.g., `(list 'a 'b 'c)` → `(a b c)`

More about lists

A list in dotted-pair notation:

`(a b c)` → `(a . (b . (c . ())))`

```
1 ]=> (define foo '(a . (b . (c . ()))))
;Value: foo

1 ]=> (list? foo)
;Value: #t

1 ]=> (pair? foo)
;Value: #t
```

Proper lists:

`(), (a (b (c) d) e)`

`(cons 'a '(b))` → `(a b)`

Dotted pairs (improper lists):

`(cons 'a 'b)` → `(a . b)`

`(car '(a . b))` → `a`

`(cdr '(a . b))` → `b`

`(cons 'a '(b . c))` → `(a b . c)`