Symbols

In addition to numbers, strings, and booleans, Scheme has a primitive data-type (atom) called symbol.

A symbol is a lot like a string. It is written: 

'identifier

Here are some examples:

'apple
'pear
'automobile

(symbol? arg) checks if an atom is a symbol.

To compare two symbols for equality, use (eq? arg1 arg2). HTDP says to use (symbol=? arg1 arg2) but DrScheme doesn’t seem to support this.

Symbols...

> (symbol? "hello")
#f

> (symbol? 'apple)
#t

> (eq? 'a 'a)
#t

> (eq? 'a 'b)
#f

> (display 'apple)
apple

> (string->symbol "apple")
apple

> (symbol->string 'apple)
"apple"

(define (healthy? f)
  (case f
    [(sushi sashimi) 'hell-yeah]
    [(coke) 'I-wish]
    [(licorice) 'no-but-yummy]
    [else 'nope]
  ))

> (healthy? 'sashimi)
hell-yeah
> (healthy? 'coke)
i-wish
> (healthy? 'licorice)
no-but-yummy
> (healthy? 'pepsi)
nope
Structures

Some versions of Scheme have structures. Select Advanced Student in DrScheme.

These are similar to C’s struct, and Java’s class (but without inheritance and methods).

Use define-struct to define a structure:

\[
\text{(define-struct \textit{struct-name} (f1 f2 \ldots))}
\]

define-struct will automatically define a constructor:

\[
\text{(make-struct-name (f1 f2 \ldots))}
\]

and field-selectors:

\[
\begin{align*}
\text{struct-name-f1} \\
\text{struct-name-f2}
\end{align*}
\]

Equivalence

Every language definition has to struggle with equivalence; i.e. what does it mean for two language elements to be the same?

In Java, consider the following example:

```java
void M(String s1, String s2, int i1, int i2) {
    if (i1 == i2) \ldots;
    if (s1 == s2) \ldots;
    if (s1.equals(s2)) \ldots;
}
```

Why can I use == to compare ints, but is it usually wrong to use it to compare strings?

Equivalence...

Scheme has three equivalence predicates \texttt{eq?}, \texttt{eqv?} and \texttt{equal?}.

\texttt{eq?} is the pickiest of the three, then comes \texttt{eqv?}, and last \texttt{equal?}.

In other words,

- If \texttt{(equal? a b)} returns \#t, then so will \texttt{(eq? a b)} and \texttt{(eqv? a b)}.
- If \texttt{(eqv? a b)} returns \#t, then so will \texttt{(eq? a b)}.
- \texttt{(equal? a b)} generally returns \#t if \texttt{a} and \texttt{b} are structurally the same, i.e. print the same.
(eqv? a b) returns #t if:
- a and b are both #t or both #f.
- a and b are both symbols with the same name.
- a and b are both the same number.
- a and b are strings that denote the same locations in the store.

> (define S "hello")
> (eqv? S S)
true
> (eqv? "hello" "hello")
false
> (eqv? 'hello 'hello)
true

(equal? a b) returns #t if a and b are strings that print the same.
This is known as structural equivalence.

> (equal? "hello" "hello")
true
> (equal? alice bob)
false
> (define alice1 (make-person "alice" 'female '1979))
> (define alice2 (make-person "alice" 'female '1979))
> (equal? alice1 alice2)
true