Consider the similarities between syntactic conventions in text and related conventions in graphical representations.
Computer programmers didn’t invent bugs. Think about how bugs might be introduced in copying or modifying graphical representations.
Ubiquitous Syntax
Computing factorial!

(define (factorial n)
    (if (= n 0)
        1
        (* n (factorial (- n 1)))))
Don’t Sweat Syntax

Proceed by substitution

(factorial 3)
(if (= 3 1) 1 (* 3 (factorial (- 3 1))))
(if #f 1 (* 3 (factorial (- 3 1))))
(* 3 (factorial (- 3 1)))
(* 3 (factorial 2))
...

Don’t Sweat Syntax

Other syntactic conventions

\[
\text{Factorial}[n_] := \\
\quad \text{If}[n == 0, \\
\quad \quad 1, \\
\quad \quad 1, \\
\quad \quad n \ast \text{Factorial}[n - 1]]
\]
Don’t Sweat Syntax

Strip it to the bone

```java
factorial( n )
    if ( n == 0 )
        1
    n * factorial( n - 1 )
```
Don’t Sweat Syntax

Pity the poor compiler

```c
int factorial( int n ) {
    int result;
    if ( n == 0 )
        result = 1;
    else
        result = n * factorial( n - 1 );
    return result;
}
```
And then iterate ...

```c
int factorial( int n ) {
    int result = 1;
    for ( int i = 2; i <= n; i++ ) {
        result = result * i;
    }
    return result;
}
```
Substance Over Style

And again ...

```c
int factorial( int n ) {
    int result = 1;
    while ( n > 1 ) {
        result = result * n;
        n = n - 1;
    }
    return result;
}
```
Larry Wall meet SICP

(define (factorial n)
  (aux-factorial 1 1 n))

(define (aux-factorial result i n)
  (if (> i n)
      result
      (aux-factorial (* result i)
                     (+ i 1)
                     n))))