Notation

Instead of traditional arithmetic notation, we’ll use **Racket** notation

a.k.a. *Lisp, Scheme, Beginning Student*...

<table>
<thead>
<tr>
<th>Traditional</th>
<th>Racket</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(x) = \cos(x) + 2$</td>
<td><code>(define (f x) (+ (cos x) 2))</code></td>
</tr>
</tbody>
</table>
Racket Expression Notation

- Put all operators at the front
- Start every operation with an open parenthesis
- Put a close parenthesis after the last argument
- Never add extra parentheses

<table>
<thead>
<tr>
<th>Traditional</th>
<th>Racket</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 + 2$</td>
<td>$(+ 1 2)$</td>
</tr>
<tr>
<td>$4 + 2 \times 3$</td>
<td>$(+ 4 (* 2 3))$</td>
</tr>
<tr>
<td>$\cos(0) + 1$</td>
<td>$(+ (\cos 0) 1)$</td>
</tr>
</tbody>
</table>
Racket Definition Notation

- Use **define** instead of =
- Put **define** at the front, and group with parentheses
- Move open parenthesis from after function name to before

<table>
<thead>
<tr>
<th>Traditional</th>
<th>Racket</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(x) = \cos(x) + 2 )</td>
<td><code>(define (f x) (+ (cos x) 2))</code></td>
</tr>
</tbody>
</table>

- Move open parenthesis in function calls

<table>
<thead>
<tr>
<th>Traditional</th>
<th>Racket</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(0) )</td>
<td><code>(f 0)</code></td>
</tr>
<tr>
<td>( f(2+3) )</td>
<td><code>(f (+ 2 3))</code></td>
</tr>
</tbody>
</table>
Evaluation is the Same as Before

\[
\text{(define } (f \ x) (\text{+ } (\cos \ x) \ 2))
\]

\[
(f \ 0)
\]
Evaluation is the Same as Before

\[(\text{define } (f \ x) \ (+ \ (\cos \ x) \ 2))\]

\[ (f \ 0) \]
\[ \rightarrow (+ (\cos \ 0) \ 2) \]
Evaluation is the Same as Before

\[(\text{define} \ (f \ x) \ (+ \ (\cos \ x) \ 2))\]

\[(f \ 0)\]
\[\rightarrow (+ (\cos \ 0) \ 2)\]
\[\rightarrow (+ 1 \ 2)\]
Evaluation is the Same as Before

\[
\text{(define (f x) (+ (cos x) 2))}
\]

\[
(f \ 0)
\]
\[
\rightarrow (+ (cos \ 0) 2)
\]
\[
\rightarrow (+ 1 2)
\]
\[
\rightarrow 3
\]