Evaluation is application of eval function

Evaluation is the only thing the Racket computer does.

In Chapter 2 of our textbook, we went over all the Racket data types that evaluate to themselves. This means that when the eval function is applied to the character sequence representing the datum, the result (the output of the eval function) is represented by the same character sequence as the input.

The eval function

In Racket, unquoted symbols are used as names for constants and functions.

The evaluation of unquoted symbols is different from the evaluation of primitive data types because unquoted symbols evaluate to the value associated with the symbol name in the GE.

Evaluation of constant names

Suppose you have the following define statements in a Racket program:

\[
\begin{align*}
\text{(define NUM 3)} \\
\text{(define XYZ 42)} \\
\text{(define BOOL #t)} \\
\text{(define STR "hello")}
\end{align*}
\]

These define statements add the lines shown to the GE.

Evaluation of unquoted symbols

The values are shown as phrases to emphasize the point that we don’t know exactly how the values are stored by the Racket computer (although we will soon drop this style of notation).

<table>
<thead>
<tr>
<th>Symbol Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUM</td>
<td>number three</td>
</tr>
<tr>
<td>XYZ</td>
<td>number forty-two</td>
</tr>
<tr>
<td>BOOL</td>
<td>boolean true</td>
</tr>
<tr>
<td>STR</td>
<td>string &quot;hello&quot;</td>
</tr>
</tbody>
</table>

The textbook uses the symbol \( \Rightarrow \) to represent application of the eval function.

Evaluation of symbol names

When the symbol NUM is evaluated, the following notation is used:

\[
\text{NUM} \Rightarrow \text{[the number three} \Rightarrow \text{the number three}] \Rightarrow 3
\]

The textbook uses the symbol \( \Rightarrow \) to represent denotation, so we have NUM denotes the number three and, since numbers evaluate to themselves, the number three is returned as 3. Everything inside \('\)'s is not visible to us.
Chapter 3 Summary
The `eval` function evaluates unquoted symbols by looking up their value in the closest enclosing environment, for now, the Global Environment (GE).
1. Each valid expression denotes a Racket datum, and
2. Each Racket datum evaluates to a (possibly different) Racket datum

To see why the second statement holds, consider that the symbol `xyz` evaluates not to the symbol `xyz`, but to the number 42. Symbols can evaluate to only 1 data type.

Chapter 4 Summary
Chapter 4 in our textbook goes over the DrScheme (DrRacket) IDE. We will be using DrRacket throughout the semester.

Chapter 4 Summary
The main point of this chapter is given in the figure below:
1. the input character sequence $C_{in}$ denotes some Racket datum $D_{in}$
2. the datum $D_{in}$ is evaluated by Racket, yielding $D_{out}$, and
3. $D_{out}$ is displayed as the character sequence $C_{out}$

Chapter 5 – Built-in Functions
Chapter 5 in our textbook goes over some of the built-in functions available in DrRacket.

It also introduces Contracts (i.e., header comments) for functions: `+`, `-`, `quotient`, `remainder`, `integer?`, `printf`, and `void`.

Chapter 5 – Built-in Functions
The header comments in the book are slightly different from what you will be asked to write. The difference is that the textbook's header comments contain no "Usage" line, something I consider to be essential in the header.