Lecture 02: Data Types and Evaluation
10:00 AM, Sep 7, 2018

Contents

1 Procedural versus Functional Programming 1
  1.1 History ................................................................. 1
  1.2 CS17’s Approach .................................................. 1

2 Lisp and Racket 1

3 Data Objects and Types 1
  3.1 Numbers .............................................................. 2
  3.2 Booleans .............................................................. 2
  3.3 Symbol ............................................................... 2
  3.4 Lists ................................................................. 3

4 Denotation vs. Expressions 3

5 Evaluation 3
  5.1 Rules of Evaluation .............................................. 4

1 Procedural versus Functional Programming

1.1 History

David Hilbert: Asked, “Is there a ‘mechanical’ process that can determine whether any mathematical statement is true or false?”

Alan Turing: Formulated a model of a machine capable of general computation. Used it to prove that there is no such process!

Alonzo Church: I proved that already! (came up with lambda calculus which gave rise to Lisp, one of the earliest programming languages invented)

(Anything we consider computing can be modeled by the Turing machine.)

1.2 CS17’s Approach

Our approach to programming in CS17 closely resembles the Church model of computing. In CS18, you will work more on the Turing approach to programming.
2 Lisp and Racket

Many dialects evolved from Lisp. In this class, we use the Racket language and system, a simple subset of Scheme.

There is a tradition of using Scheme/Racket as an educational language.

This set of features and concepts that we will see in Racket is very small, but very powerful.

3 Data Objects and Types

Data objects and types are essentially things that the computer can think about. Data should be considered to be mathematical objects not objects in the real world.

There are multiple representations for data and different ways to remember them in a computer.

Data types in our first language:

- Numbers
- Booleans
- Symbols
- Lists
- Functions(procedures)

3.1 Numbers

Ways you can write them:

- 17 (The integer seventeen)
- -17 (The integer negative seventeen)
- 3.14159 (The real number that approximates pi)
- 2/3 (The fraction two-thirds)
- 6.022e+23 (Scientific notation $6.022 \cdot 10^{23}$)

Racket uses formats, but not necessarily ones you are used to seeing

3.2 Booleans

There are only two boolean values:

- True
- False
3.3 Symbol

Sequence of letters and digits and punctuation marks. Examples:

- cs17
- MyName
- if.you#dare
- +
- *
- -
- /

3.4 Lists

A list is a sequence of data objects (Any number, including zero). Denotation of lists:

- Left parenthesis
- First item
- Space(s)
- ....
- Last item
- Right parenthesis

Examples:

- (1 3 5 7)
- (hello world)
- (+ 10 7)
- (+ (* 2 5) 7)
4 Denotation vs. Expressions

*Text:* A sequence of characters that you or the computer writes.

*Denotation of a text:* The data object that the text represents.

Examples:

- 17 denotes the integer seventeen
- Hello denotes a symbol
- + denotes a symbol
- (a b c) denotes a list
- (+ 10 7) denotes a list

5 Evaluation

Evaluation is a complicated process but it follows certain rules, the *Rules of Evaluation.*

For example, the list denoted by (+ 10 7) evaluates to 17.

5.1 Rules of Evaluation

- The value of a number is itself.
- The value of a Boolean is itself.
- The value of a symbol is the data object it is *bound to.*
- The value of a list is obtained as follows: evaluate each item in the list. The first item should evaluate to a procedure. Apply the procedure to the rest of the items. The result of the procedure application is the value of the list.

Examples (in arithmetic)

- The symbol + is bound to the addition procedure
- The symbol * is bound to the multiplication procedure
- The symbol / is bound to the division procedure
- The symbol - is bound to the subtraction procedure

*Note:* The subtraction and division procedures in Racket evaluate the integers that follow in order. For example, the expression (- 5 2) evaluates to 3 and the expression (/ 4 2) evaluates to 2.
Please let us know if you find any mistakes, inconsistencies, or confusing language in this or any other CS 17 document by filling out the anonymous feedback form: http://cs.brown.edu/courses/csci0170/feedback