Beginning Scheme

You may also want to keep the corresponding reading at hand.

- **Exercises**
  - Exercise 0: Preparation
  - Exercise 1: Square Roots
  - Exercise 2: Simple Subtraction
  - Exercise 3: Simple Multiplication
  - Exercise 4: Extended Addition
  - Exercise 5: Absolute Value
  - Exercise 6: Exponentiation
  - Exercise 7: Normal Mathematical Notation
  - Exercise 8: Simple Definitions
  - Exercise 9: The Definitions Window
  - Exercise 10: Saving Files
  - Exercise 11: Reloading Files
  - Exercise 12: Wrapup

- **Notes**
  - Notes on Exercise 7

### Exercises

**Exercise 0: Preparation**

Start DrScheme and make sure that you’re in full scheme mode. You may want to read the DrScheme lab to make sure you understand DrScheme.

**Exercise 1: Square Roots**

a. Convince DrScheme to compute the square root of 137641.

b. Verify that the value that DrScheme returns is indeed the square root of 137641. (Try squaring the value DrScheme gives you by hand.)

**Exercise 2: Simple Subtraction**

Ask DrScheme to subtract 68343 from 81722.
**Exercise 3: Simple Multiplication**

Tell DrScheme to multiply 162 by 1383.

**Exercise 4: Extended Addition**

a. Ask DrScheme to add 3 and 4.

b. Ask DrScheme to add 3 and 4 and then add 5 to the result. You’ll need two calls to +.

c. Ask DrScheme to add 3, 4, and 5 using only one call to +.

d. What happens if you call the procedure + with no arguments? With only one?

**Exercise 5: Absolute Value**

Have DrScheme compute the absolute value of -197. You can use the abs procedure.

**Exercise 6: Exponentiation**

a. Ask DrScheme to compute the cube of 19 (that is, the result of raising 19 to the power 3). You can use expt to compute exponents.

b. Ask DrScheme to computer the nineteenth power of 3.

c. What do these results indicate about the relationship between procedures and arguments in Scheme?

**Exercise 7. Normal Mathematical Notation**

Type each of the following expressions at the Scheme prompt and see what reaction you get.

1. (2 + 3)
2. 7 * 9
3. sqrt(49)

You may wish to read the notes on this problem for an explanation of the results that you get.

**Exercise 8: Simple Definitions**

a. Write a definition that will cause Scheme to recognize dozen as a name for the number 12.

b. Write a definition that will cause Scheme to recognize raise-to-power as a synonym for expt.

c. Use both names in expressions to verify that Scheme has understood them.
Exercise 9: The Definitions Window

Copy the definitions you wrote for the preceding two exercises into the definitions window and execute them.

Exercise 10: Saving Files

Save the definitions that you copied into the definitions window in the previous exercise in a file named beginning-scheme.ss. (Conventionally, the names of files containing Scheme programs end in .ss.)

Exercise 11: Reloading Files

a. Quit and restart DrScheme.

b. Determine whether dozen is still defined. (It shouldn’t be.)

c. See if you can figure out how to get DrScheme to reload your saved definitions.

Exercise 12: Wrapup

Quit DrScheme and log out of the workstation.

Notes

Notes on Exercise 7

\((2 + 3)\)

When DrScheme sees the left parenthesis at the beginning of the expression \((2 + 3)\), it expects the expression to be a procedure call, and it expects the procedure to be identified right after the left parenthesis. But 2 does not identify a procedure; it stands for a number. (A “procedure application” is the same thing as a procedure call.)

\(7 * 9\)

In the absence of parentheses, DrScheme sees \(7 * 9\) as three separate and unrelated expressions -- the numeral 7; *, a name for the primitive multiplication procedure; and 9, another numeral. It interprets each of these as a command to evaluate an expression: “Compute the value of the numeral 7! Find out what the name * stands for! Compute the value of the numeral 9!” So it performs the first of these commands and displays 7; then it carries out the second command, reporting that * is the name of the primitive procedure *; and finally it carries out the third command and displays the result, 9. This behavior is confusing, but it’s strictly logical if you look at it from the computer’s point of view (remembering, of course, that the computer has absolutely no common sense).
As in the preceding case, DrScheme sees `sqrt(49)` as two separate commands: `sqrt` means “Find out what `sqrt` is!” and `(49)` means “Call the procedure 49, with no arguments!” DrScheme responds to the first command by reporting that `sqrt` is the primitive procedure for computing square roots and to the second by pointing out that the number 49 is not a procedure.