

Laboratory: Numeric Values

Summary: We explore the variety of types of numbers that Scheme (well, DrScheme) provides. We also explore some of the interesting procedures that operate on numeric values.

Numeric predicates used in this lab: `complex?`, `exact?`, `inexact?`, `number?`, `rational?`, and `real?`.

Numeric procedures used in this lab: `denominator`, `numerator`, and `sqrt`.

Contents:

- Exercises
 - Exercise 0: Preparation
 - Exercise 1: Types of Numbers
 - Exercise 2: The Type of a Square Root
 - Exercise 3: Checking Answers
 - Exercise 4: Checking Answers, Revisited
 - Exercise 5: A Large Number
 - Exercise 6: Inexact Fractions
 - Exercise 7: Other Integer Procedures
 - Exercise 8: Exploring Rationals
 - Exercise 9: Rounding, Revisited
 - Exercise 10: Other Numeric Predicates
- Extra Work
- Notes
 - Notes on Exercise 7
 - Notes on Exercise 8

Exercises

Exercise 0: Preparation

Start DrScheme.

Exercise 1: Types of Numbers

Have DrScheme confirm that $3/4$ is a rational number but not an integer and that the square root of -1 is a complex number but not a real number.

Exercise 2: The Type of a Square Root

Confirm that the value DrScheme computes for `(sqrt 2)` is an inexact real that is also rational.

Exercise 3: Checking Answers

As you know, one of the goals we have in this class is of verifying the answers given to us by algorithms (primarily our own algorithms, but, at times, those given to us by the computer).

We might confirm that the value returned by `(sqrt 2)` is correct by computing the square of that value and then subtracting 2. If the square root is correct, the result should be 0.

Check the value returned by the previously described expression (that is, two less than the square of the square root of 2). Is it 0? Why do you think you got the answer you got?

Exercise 4: Checking Answers, Revisited

a. Do you expect to have the same problem as in the previous exercise if you compute the square root of 4 rather than the square root of 2? Why or why not?

b. Confirm your answer experimentally.

Exercise 5: A Large Number

Write a Scheme numeral for 1.507 times ten to the fifteenth power, as an *exact* number. (Make sure to use `exact?` to check whether the number is exact.) Have Scheme evaluate the numeral.

Exercise 6: Inexact Fractions

Write a Scheme numeral for one-third, as an *inexact* number. Have Scheme evaluate the numeral.

Exercise 7: Other Integer Procedures

Scheme provides a number of numerical procedures that can produce integer results. We've already explored `expt`, `abs`, `+`, `-`, and `*`.

Here are some others. For each, try to figure (by experimentation, by discussing results with other students, and, eventually, by reviewing the reading and associated documentation) out how many parameters each procedure can take and what the procedure does. Make sure to try a variety of values for each procedure, including positive and negative, integer and real.

Warning: You may not be able to figure all of them out.

a. `quotient`

- b. remainder
- c. modulo
- d. max
- e. min
- f. numerator
- g. denominator
- h. gcd
- i. lcm
- j. floor
- k. ceiling
- l. truncate
- m. round

Exercise 8: Exploring Rationals

Since you've found that DrScheme seems to represent every real number as a rational, it might be worth finding a way to see what that rational number is.

- a. Determine the numerator and denominator of the rational representation of the square root of 2.
- b. Determine the numerator and denominator of the rational representation of 1.5.
- c. Determine the numerator and denominator of the rational representation of 1.2.
- d. Determine the numerator and denominator of $6/5$.
- e. Determine the numerator and denominator of `#i6/5`.

If you're puzzled by some of the later answers, you may want to read the notes on this problem.

Exercise 9: Rounding, Revisited

For small numbers, the `exact->inexact` procedure produces lots and lots of digits after the decimal point. Figure out how to get just two digits after the decimal point. You may need to use multiplication, division, and some of the last procedures from the previous exercise.

You need not implement your algorithm; simply come up with one you think will work.

Exercise 10: Other Numeric Predicates

We've already seen a variety of predicates (procedures that return true or false) that can be applied to numbers. These predicates include `exact?`, `integer?`, and `real?`.

By reading the Scheme documentation, identify other predicates that can be applied to numbers.

Extra Work

If you finish early, you might

- Implement a working version of the algorithm from Exercise 9.
- Identify a fraction a/b such that `(inexact->exact (exact->inexact a/b))` is not the same as a/b .

Notes

Notes on Exercise 7

When you get stuck on this problem, it's probably worth skimming through DrScheme's *Help Desk*. The numeric operations are documented in section 6.2.5 of the Revised(5) Report on the Algorithmic Language Scheme. (I have not yet put all of these procedures in the Glimmer Scheme Reference.

Notes on Exercise 8

DrScheme seems to represent the fractional part of many numbers as the ratio of some number and 4503599627370496, which happens to be 2^{52} . (Most computers like powers of 2.) If you are energetic, you might scour the Web to find out why they use an exponent of 52.

Copyright © 2007 Samuel A. Rebelsky. This work is licensed under a Creative Commons Attribution-NonCommercial 2.5 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/2.5/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.