SICP Examples

H. Conrad Cunningham

10 April 2022

Contents

Professor of Computer and Information Science
University of Mississippi
214 Weir Hall
P.O. Box 1848
University, MS 38677
(662) 915-7396 (dept. office)

Browser Advisory: The HTML version of this textbook requires a browser that supports the display of MathML. A good choice as of April 2022 is a recent version of Firefox from Mozilla.

SICP Examples

Background

The examples given in the collection are based on examples given in Chapter 1 of the classic SICP textbook [1]:

> Harold Abelson and Gerald J. Sussman with Julie Sussman. *Structure and Interpretation of Computer Programs*, Second Edition, MIT Press, 1996.

- book site at MIT Press
- HTML book
- SICP ebook site
- local copy of source code

Haskell

The Haskell solutions are discussed in Chapters 4, 6, and 9 of the textbook *Exploring Languages with Interpreters and Functional Programming*.

Scala

First-order functions in Scala

- Square root (Newton's Method) with all public functions: Scala source
- Square root (Newton's Method) with function definitions: Scala source
- Factorial (Note 1): Scala source
- Fibonacci (Note 1): Scala source
- Exponentiation using `Double` (Note 1): Scala source
- Exponentiation using `BigInt` (Note 1): Scala source
- Greatest common divisor: Scala source

Notes:

1. Similar Factorial, Fibonacci, and Exponentiation examples are discussed in the Recursion Styles, Correctness, and Efficiency (Scala) chapter.

Higher-order functions in Scala

- Summation (takes function arguments): Scala source
- Derivative (returns function result): Scala source

Elixir

First-order functions in Elixir

- Square root (Newton's Method) with all public functions: Elixir source
- Square root (Newton's Method) with function definitions: Elixir source
- Factorial: Elixir source
- Fibonacci: Elixir source
- Exponentiation: Elixir source
- Greatest common divisor: Elixir source

Higher-order functions in Elixir

- Summation (takes function arguments): Elixir source
- Derivative (returns function result): Elixir source

Lua

First-order functions in Lua

- Square root (Newton's Method): Lua source
- Factorial: Lua source
- Fibonacci: Lua source
- Exponentiation: Lua source
- Greatest common divisor: Lua source

Higher-order functions in Lua

- Summation: Lua source
- Derivative: Lua source

Elm

First-order functions in Elm

- Square root (Newton's Method): Elm source
- Factorial: None

- Fibonacci: Elm source
- Exponentiation: Elm source
- Greatest common divisor: None

Higher-order functions in Elm

- Summation: None
- Derivative: None

Racket Scheme

First-order functions in Racket Scheme

- Square root (Newton's Method): None
- Factorial: None
- Fibonacci: None
- Exponentiation: Racket Scheme source
- Greatest common divisor: None

Higher-order functions in Racket Scheme

- Summation: None
- Derivative: None

Python 3

First-order functions in Python

- Square root (Newton's Method): Python source
- Factorial: None
- Fibonacci: None
- Greatest common divisor: None

Higher-order functions in Python

- Summation: None
- Derivative: None

Acknowledgements

I often use some or all of these "seven" examples from Chapter 1 of the classic SICP textbook [1] when I am learning and/or teaching a "functional" programming language. I have done that with (at least) Lua, Elixir, Scala, Elm, Haskell, and Python since 2013. This chapter collects the source code for most of these efforts.'

I retired from the full-time faculty in May 2019. As one of my post-retirement projects, I am continuing work on possible textbooks based on the course materials I had developed during my three decades as a faculty member. In January 2022, I began refining the existing content, integrating separately developed materials together, reformatting the documents, constructing a unified bibliography (e.g., using citeproc), and improving my build workflow and use of Pandoc. I adapted this index page from a portion of my Spring 2019 CSci 555 course notes.

I maintain this chapter as text in Pandoc's dialect of Markdown using embedded LaTeX markup for the mathematical formulas and then translate the document to HTML, PDF, and other forms as needed.

References

[1] Harold Abelson and Gerald Jockay Sussman. 1996. *Structure and interpretation of computer programs (SICP)* (Second ed.). MIT Press, Cambridge, Massachusetts, USA. Retrieved from https://mitpress.mit.edu/sicp/