Scheme is a Dialect of Lisp

What are people saying about Lisp?

• “If you don’t know Lisp, you don’t know what it means for a programming language to be powerful and elegant.”
  Richard Stallman, created Emacs & the first free variant of UNIX

• “The only computer language that is beautiful.”
  Neal Stephenson, DeNero’s favorite sci-fi author

• “The greatest single programming language ever designed.”
  Alan Kay, co-inventor of Smalltalk and OOP (from the user interface video)

Scheme Expressions

Scheme programs consist of expressions, which can be:

• Primitive expressions: 2 3.3 true + quotient

• Combinations: (quotient 10 2) (not true)

Numbers are self-evaluating; symbols are bound to values

Call expressions include an operator and 0 or more operands in parentheses

(Demo)

> (quotient 10 2)
5
> (quotient (+ 8 7) 5)
3
> (define pi 3.14)
> (* pi 2)
6.28
> (define (abs x)
   (if (< x 0)
       (- x)
       x))
> (abs -3)
3

Special Forms

A combination that is not a call expression is a special form:

• if expression: (if <predicate> <consequent> <alternative>)

• and, or: (and <e1> ... <en>, or <e1> ... <en>)

• Binding symbols: (define <symbol> <expression>)

• New procedures: (define (<symbol> <formal parameters>) <body>)

Evaluation:
(1) Evaluate the predicate expression
(2) Evaluate either the consequent or alternative

(Demo)
Lambda Expressions

Lambda expressions evaluate to anonymous procedures

\[(\lambda (\text{formal-parameters}) \text{body})\]

Two equivalent expressions:

(define (plus4 x) (+ x 4))
(define plus4 (lambda (x) (+ x 4)))

An operator can be a call expression too:

\[(\lambda (x y) (+ x y \text{square}(z)))\]\(\text{Evaluate to the}\)\(x+y+z\) procedure

Lists

Scheme Lists

In the late 1950s, computer scientists used confusing names

- cons: Two-argument procedure that creates a linked list
- car: Procedure that returns the first element of a list
- cdr: Procedure that returns the rest of a list
- nil: The empty list

Important! Scheme lists are written in parentheses with elements separated by spaces

\[(1 2)\]
(define x (cons 1 (cons 2 nil))
\)[(Demo)]

Symbolic Programming

Symbols normally refer to values; how do we refer to symbols?

> (define a 1)
> (define b 2)
> (list a b)

Quotation is used to refer to symbols directly in Lisp.

> (list 'a 'b)
> (list 'a b)

Quotation can also be applied to combinations to form lists.

> '(a b c)
> (list 'a b c)
> (car '(a b c))
> (cdr '(a b c))

Sierpinski's Triangle

(Demo)