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, creator of Emacs and 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

Numbers are self-evaluating; symbols are bound to values

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

**Scheme Interpreters**

Lambda Expressions

Lambda expressions evaluate to anonymous procedures

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

Two equivalent expressions:

\[
\text{define} \quad \text{plant} \quad (x + 4)
\]

\[
\text{define plant} \quad (x + 4)\]

An operator can be a call expression too:

\[
\text{lambda} \quad (x \quad y) \quad (x \quad y \quad \text{square} \quad 2 \quad 3) \quad 12
\]

Evaluation:

1. Evaluate the lambda expression
2. Evaluate the arguments
3. Substitute the arguments
4. Evaluate the expression

Lists

Lambda expressions evaluate to anonymous procedures

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

```scheme
> (1 2)
> (define x (cons 1 (cons 2 nil))
> x
(1 2)
> (car x)
1
> (cdr x)
(2)
> (cons 1 (cons 2 (cons 3 (cons 4 nil))))
(1 2 3 4)
```

### Symbolic Programming

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

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

Quotation is used to refer to symbols directly in Lisp.

```scheme
> (list 'a 'b)
(a b)
> (list 'a b)
(a 2)
```

Quotation can also be applied to combinations to form lists.

```scheme
> '(a b c)
(a b c)
> (car '(a b c))
a
> (cdr '(a b c))
(b c)
```

Short for (quote a), (quote b):

- Special form to indicate that the expression itself is the value.

---

**Pairs Review**

In the late 1950s, computer scientists used confusing names:

- **cons**: Two-argument procedure that creates a pair.
- **car**: Procedure that returns the first element of a pair.
- **cdr**: Procedure that returns the second element of a pair.
- **nil**: The empty list.

- A (non-empty) list in Scheme is a pair in which the second element is **nil** or a Scheme list.

- Important! Scheme lists are written in parentheses separated by spaces.

```scheme
> (1 2)
> (define x (cons 1 2))
> x
(1 . 2)
> (car x)
1
> (cdr x)
2
> (cons 1 (cons 2 (cons 3 (cons 4 nil))))
(1 2 3 4)
```

### Pairs and Lists

In the late 1950s, computer scientists used confusing names:

- **cons**: Two-argument procedure that creates a pair.
- **car**: Procedure that returns the first element of a pair.
- **cdr**: Procedure that returns the second element of a pair.
- **nil**: The empty list.

- A (non-empty) list in Scheme is a pair in which the second element is **nil** or a Scheme list.

- Important! Scheme lists are written in parentheses separated by spaces.

- A well-formed list has some value for the second element of the last pair that is not a list.

```scheme
> (1 2)
> (define x (cons 1 (cons 2 nil)))
> x
(1 2)
> (car x)
1
> (cdr x)
2
> (cons 1 (cons 2 (cons 3 (cons 4 nil))))
(1 2 3 4)
```

### Programming Languages

A computer typically executes programs written in many different programming languages:

- **Machine languages**: statements are interpreted by the hardware itself.
  - A fixed set of instructions (machine operations) implemented by the circuitry of the central processing unit (CPU).
  - Operations refer to specific hardware memory addresses; no abstraction mechanisms.

- **High-level languages**: statements & expressions are interpreted by another program or compiler (interpreted) or another language.
  - Provide means of abstraction such as naming, function definition, and objects.
  - Abstract away system details to be independent of hardware and operating system.

<table>
<thead>
<tr>
<th>Programming Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
</tr>
<tr>
<td>R</td>
</tr>
<tr>
<td>Julia</td>
</tr>
</tbody>
</table>

### Sierpinski’s Triangle

**Type of application**: Erlang was designed for concurrent programs. It has built-in elements for expressing concurrent communication.

**Problem domain**: The MediaWiki mark-up language was designed for generating static web pages. It has built-in elements for text formatting and cross-page linking.

### Metalinguistic Abstraction

A powerful form of abstraction is to define a new language that is tailored to a particular type of application or problem domain:

- **Syntax**: The legal statements and expressions in the language.
- **Semantics**: The execution/evaluation rule for those statements and expressions.

To create a new programming language, you either need a:

- **Specification**: A document that describes the precise syntax and semantics of the language.
- **Canonical implementation**: An interpreter or compiler for the language.