

# Introduction to Lush

Fu Jie Huang

<http://lush.sourceforge.net>

# Language Design

- Simple syntax
- Interpreted (typeless)
- Can be compiled (typed)
- C/C++ inline
- Powerful built-in data structure
- Online help

# Functions

- The essential two things:
  - Define a function
  - Apply the function
- Compare with C

```
? (def square (x)      double square (double x) {
    (* x x))           return x*x;
                       }

```

```
? (square 4)
= 16
```

```
square(4);
```

# Operators

- Prefix, no infix

– ? (+ 3 4)                   ;; 3+4

– ? (\* 3 4)                   ;; 3\*4

– ? (> (- 2 4) 1)           ;; (2 - 4) > 1

– ? (- 4)                   ;; -4

- No precedence

- Really just built-in functions

# Variables and Assignment

- Global variables: `setq`
  - ? (`setq x 5`)                    `:: x = 5`
  - ? (`* x x`)
- Local variables: `let`, `let*`
  - ? (`de harmonic (n)`
    - `(let* ((z 0) (i 0))`
      - `(while (< i n)`
        - `(incr i)`
        - `(incr z (/ i))))))`

# Control Flow

- Selection `if, when, cond, selectq`

- no `else!`

- ? (`if (> x 0)` ;; a sign function  
+1  
-1)

- Repetition `for, while, repeat`

- ? (`for (i 2 5)`  
(`print i (sqrt i)`))

2	1.4142
3	1.7321
4	2
5	2.2361
= 2.2361	

# Basic Types

- Literals have types!
  - numbers      ? (setq x `-3.4e6`)
  - strings      ? (setq x `“foo”`)
  - boolean      ? (setq x `t`)  
                  ? (setq x `()`)            ;: (if x 1 2)
  - lists         ? (setq x `'(1 2 3)`)
  - symbols      ? (setq x `'foo`)

# Arrays

- 0 to 8 dimensions

– ? (setq x (**matrix** 10 8 4))      ;; double x[10][8][4];

- Get value

– ? (x 0 0 0)      ;; x[0][0][0]

- Set value

– ? (x 0 0 0 **45.6**)      ;; x[0][0][0] = 45.6;



# Scalar

- Is 0 dimensional array
- **Is NOT number!**
  - Number to scalar:
    - ? (setq x ((matrix) 10))
  - Scalar to number:
    - ? (x)
- Very useful

# Scalar

- As results

- ? (idx-dot [1 2 3] [1 2 3]) ;; W'X  
= [@ 14]

- As operands

;; (y-sgn(W'X))X

- ? (idx-dotm0 [1 2 3] ((matrix) 10))  
= [10 20 30]

- ? (setq p [1 2 3])

- ? (idx-dotm0acc [1 2 3] ((matrix) 10) p) ;;what's p?

# Narrow and Select

0	1	2	3	4	5
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55

? (**select** x 1 1) 1D

? (**narrow** (**narrow** x 0 2 4) 1 2 4) 2D

? (**narrow** (**select** x 0 0) 0 2 3) 1D

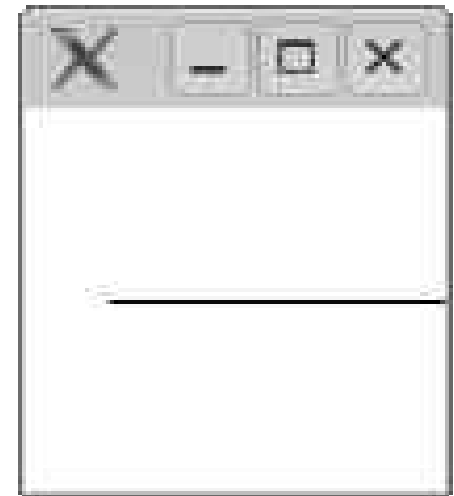
? (**select** (**select** x 1 3) 0 2) scalar!

# Array Iterator

- Works like `foreach`
  - ? `(setq x [1 2 3])`
  - ? `(idx-bloop ((v x))  
                  (v (* (v) 10)))`  
= [ 10 20 30]
- Iterators can be nested
  - inner most: manipulate scalar

# GUI, Plotting

- Open a window
  - ? (`new-window 10 10 100 100`)
  - `<x> <y> <w> <h>`
- Draw a line
  - ? (`draw-line 20 50 100 50`)
  - `<x1> <y1> <x2> <y2>`
  - figure out the origin, x-axis, y-axis!



# Searchable Online Help

The screenshot shows a window titled 'Lush Manual' with a search bar containing 'Scalars'. The left pane displays a tree view of the manual's contents, with 'Scalars, Vectors, Matrices, Tensors, and Strings' selected. The right pane shows the search results for this topic.

**Scalars, Vectors, Matrices, Tensors, and Strings**

Lush's plane of existence is its matrix engine. Lush can operate on scalars, vectors, matrices, or high-dimensional tensors from 0 to 5 dimensions, creating a tensor of **float** or **double** type.

```
float m = matrix( [ 0 4 ] , 2, 1); // create 2x1 matrix
m // [0 4]
m [ 4 2 4 ] // set value of element (0,4,2) to 4.5
m // [0 4 2 4.5]
m [ 4 2 4 ] // get value of element (0,4,2)
```

Tensors of various basic types can be created with the functions listed below. Each function has two versions, the regular version initializes all the elements to zero, while the version with **no** at the end does not. And we don't forget, all of these functions take 0 to 5 integer arguments that are indices in each dimension.

- † **double-matrix**, **double-matrix-no**, **real-matrix**, **real-matrix-no**: doubles
- † **float-matrix**, **float-matrix-no**, **float-matrix**, **float-matrix-no**: floats
- † **int-matrix**, **int-matrix-no**: 32-bit ints
- † **short-matrix**, **short-matrix-no**: 16-bit shorts
- † **byte-matrix**, **byte-matrix-no**: 8-bit bytes
- † **ubyte-matrix**, **ubyte-matrix-no**: 8-bit unsigned bytes
- † **qptr-matrix**, **qptr-matrix-no**: generic pointers (void\*)

A matrix with 0 dimensions is called a scalar. It is different from a regular number in that it behaves like a pointer to a number. (The value of a scalar passed as argument to a function can be modified by the function.)

Matrices of doubles can be entered literally using square brackets.

```
{0 4 2 4.5} // 1x4 double matrix
```

Matrices of other types can be generated by adjusting an appropriate parameter attribute upon **matrix**. Here is how to create a matrix of ints:

```
{0 4 2 4.5}
```