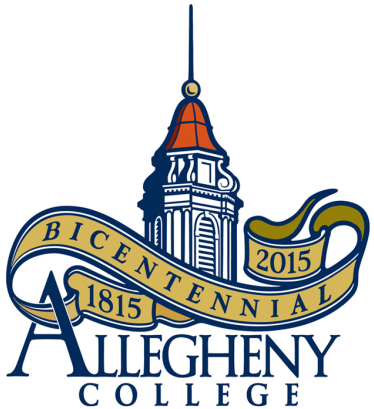
CS201 - PL'S

Functional Programming - LISP1

Aravind Mohan

Allegheny College

November 1, 2021



History

LISP

LISt Processing: invented in 1958 by John McCarthy (the person who coined the term “Artificial Intelligence”; the person who invented garbage collection for managing memory).

LISP is the second-oldest programming language still in use today.

Lisp Basics

- **Lisp syntax:** parenthesized prefix notation
- **Lisp interpreter:** read-eval-print loop
- Nested evaluation
- Preventing evalution (quote and other special forms)
- Forcing evaluation (eval)

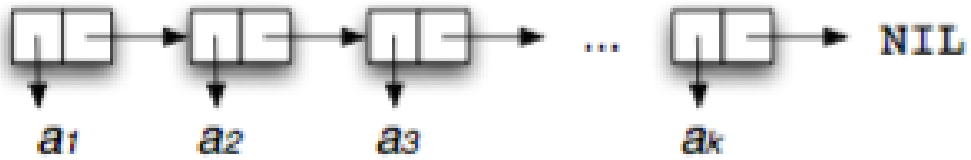
Lisp Basics

- **Fundamental structures**: atoms and lists.
- **Atom**: indivisible unit (e.g., a number, a character).
- **List**: a structure with a *head* and a *tail*. Lists are written as parenthesized expressions, e.g. (+ 1 2 3 4 5)

Atoms Examples

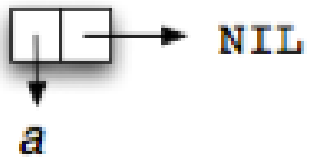
numbers	`235.4`	`2e10`	`#x16`	`2/3`
variables	`val39`	`2nd-place`	`*foo*`	
constants	`pi`	`t`	`nil`	`:keyword`
strings, chars	`"Hello!"`	`#\a`		
arrays	`#(1 "foo" A)`	`#1A(1 "foo" A)`	`#2A((A B C) (1 2 3))`	
structures	`#s(person first-name dana last-name nau)`			

List Examples

$(a_1\ a_2\ a_3\ \ldots\ a_k) \Longrightarrow$ 

$a_1, a_2, \ldots, a_k$ may be atoms or other lists

The empty list is called `()` or `NIL`; it's both a list and an atom

$(a) \Longrightarrow$ 

List Notation

```
$ clisp
 [1]> (+ 1 2 3 4 5)
15
[2]> (* 3 5 4)
60
[3]> (sqrt 2)
1.4142135
[4]> (- (+ 2 3) (* 4 8))
-27
```

The first element of a list is usually considered to be a function; the remaining elements are the arguments.

Quoted Lists

If we want a list to be just an unevaluated list of data, we can "quote it"

```
[5]> '(a b c d)
(A B C D)
[6]> '(+ 1 2 3 4 5)
(+ 1 2 3 4 5)
[7]> (quote (1 2 3 4 5))
(1 2 3 4 5)
```

“first” and “rest” (CAR and CD)

```
[11]> (first '(a b c))
A
[12]> (rest '(a b c))
(B C)
[13]> (car '(a b c))
A
[14]> (cdr '(a b c))
(B C)
```

“CAR” was the assembly language abbreviation for “Contents of the Address Register”; “CDR” was “Contents of the Decrement Register” (from the IBM 704 computer)

“first” and “rest” (CAR and CD)

```
[11]> (first '(a b c))
A
[12]> (rest '(a b c))
(B C)
[13]> (car '(a b c))
A
[14]> (cdr '(a b c))
(B C)
```

“CAR” was the assembly language abbreviation for “Contents of the Address Register”; “CDR” was “Contents of the Decrement Register” (from the IBM 704 computer)

The Empty List – () or NIL

```
[15]> (first '(10))
10
[16]> (rest '(10))
NIL
[17]> ()
NIL
```

Constructing Lists

```
[18]> (cons '+ '(1 2 3))
(+ 1 2 3)
[19]> (cons '10 NIL)
(10)
[20]> (cons '10 (cons '20 (cons '30
NIL)))
(10 20 30)
```

Creating Functions

```
[21]> (defun f (x y) (+ x y))
F
[22]> (f 20 30)
50
[23]> (defun g (list value) (cons value
list))
G
[24]> (g '(a b c) '100)
(100 A B C)
```

Conditions

```
(cond
    (condition value)
    (condition value)
    ...
    (condition value)
    (T value)
)
```

Conditionals

```
[37]> (defun mn (a b) (cond
    ( (< a b) a)
      (t b))
)
MN
[38]> (mn 30 40)
30
[39]> (mn 40 30)
30
```

Built-In Functions

list:

put things into a list

```
[18]> (list 'a)
(A)
[19]> (list '(a))
((A))
[20]> (list (list (list (list '(10 20 30)))))
(((((10 20 30)))))
[21]> (list 'a 'b 'c)
(A B C)
[22]> (list '(a b) '(10 20 30) '40 'x)
((A B) (10 20 30) 40 X)
```

Built-In Functions

length: return length of a list

```
[23]> (length '(a b c d e f g))
7
[24]> (length (list '(a b c d e f g)))
1
```

append: join two lists together

```
[25]> (append '(a b c) '(10 20 30))
(A B C 10 20 30)
[26]> (append 'a 'b) ; NOT LISTS--ERROR
*** - APPEND: A is not a list
```

User-Defined Functions

rot:

rotate a list one place to the left

```
[33]> (defun rot (lst) (append (rest lst) (list (first lst))))
ROT
[34]> (rot '(1 2 3 4))
(2 3 4 1)
[35]> (rot '(a b c d e f g h))
(B C D E F G H A)
```

Recursion: Good Old Factorial!

```
[36]> (defun fac (n)
        (cond
          ((<= n 0) 1)          ; fac(n) = 1 if n <= 0
          (t (* n (fac (- n 1)))) ; n * (n-1)! otherwise
        ))
FAC
[37]> (fac 3)
6
[39]> (fac 10)
3628800
[40]> (fac 30)
265252859812191058636308480000000
```

Class Exercise

Problem

Write a Lisp function named `half` that takes an argument list `lst` and returns a new list consisting of the second half of lst concatenated with the first half, e.g.,

```
(half '(1 2 3 4 5 6 7 8))
```

should return the value (56781234).

Reading Assignment

PLP Chapter 10

Questions

Do you have any questions from this class discussion?