

## Brief Study of LISP

NIKITA SHARMA

SOC &E, IPS Academy,

Rajendra Nagar, Indore, M.P., India

Sharmanikita738@gmail.com

**Abstract:** Artificial Intelligence (AI) is a broad field, and means different things to different people. It is concerned with getting computers to do tasks that require human intelligence. Lisp is used in AI programming, because it supports the implementation of software that computes with symbols very well. Symbols, symbolic expressions and computing with those are at the core of Lisp. Lisp is the second-oldest high-level programming language. This paper is totally based on the list programming (LISP). In this paper we will discuss about history of lisp, types of lisp, features of lisp, applications of lisp, Connection to artificial intelligence via lisp, Basic functions of lisp, user define function in lisp and scope of the lisp in AI field. And finally this paper will conclude about the importance of lisp in AI field. In other words we can say that this paper will introduce you with List Programming (LISP).

**Keywords:** AI Programming, functional language ,high-level programming language, object-oriented programming, symbolic expressions, [SHRDLU](#), [natural language understanding computer program](#), Prolog, flexibility, edit-test-debug cycle.

### History:

Lisp is the second-oldest high-level programming language after Fort edit-test-debug cycle ran and has changed a great deal since its early days, and a number of dialects have existed over its history. Today, the most widely known general-purpose Lisp dialects are Common Lisp and Scheme. Lisp was invented by John McCarthy in 1958 while he was at the Massachusetts Institute of Technology (MIT). [1]

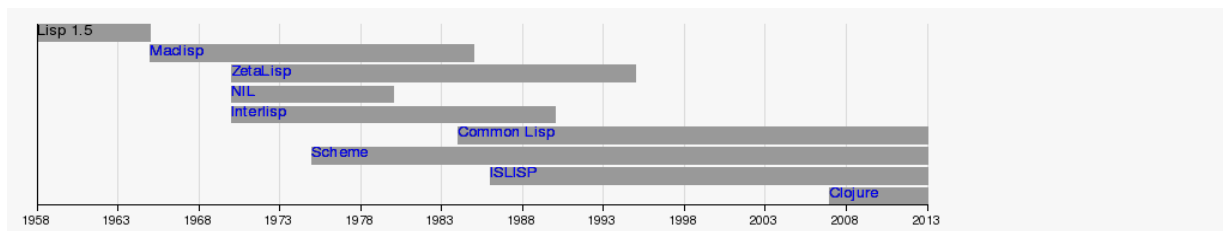


Figure 1: Types of LISP

### Features of Common LISP

- It is machine-independent.
- It is a dynamic language: editing changes take effect immediately, without the need for recompilation
- Lisp is the second-oldest high-level programming language after Fortran
- It is primarily a functional language: all work can be done via function.

- It provides advanced object-oriented programming.
- It provides convenient macro system.
- LISP expressions are called symbolic expressions or s-expressions. The s-expressions are composed of three valid objects, atoms, lists and strings.
- It provides an object-oriented condition system.
- It provides complete I/O library.
- It provides extensive control structures.
- LISP programs run either on an interpreter or as compiled code. [2]

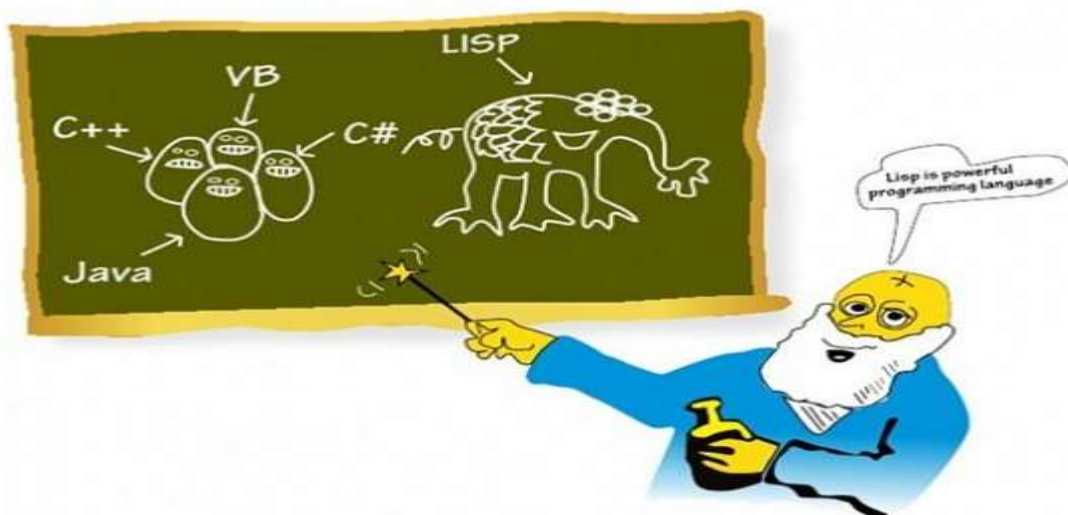


Figure 2: Flexibility of LISP

### Applications Built in LISP

Large successful applications built in Lisp.

- Emacs
- G2
- AutoCad
- Igor Engraver
- Yahoo Store [2]

### *Connection to artificial intelligence*

Since its inception, Lisp was closely connected with the [artificial intelligence](#) research community, Lisp was used as the implementation of the programming language [Micro Planner](#) which was used in the famous AI system [SHRDLU](#). In the 1970s, as AI research spawned commercial offshoots, the performance of existing Lisp systems became a growing issue. [3]

SHRDLU was an early [natural language understanding computer program](#), developed by [Terry Winograd](#) at [MIT](#) in 1968–1970. In it, the user carries on a conversation with the computer, moving objects, naming collections and querying the state of a simplified "blocks world". [4]

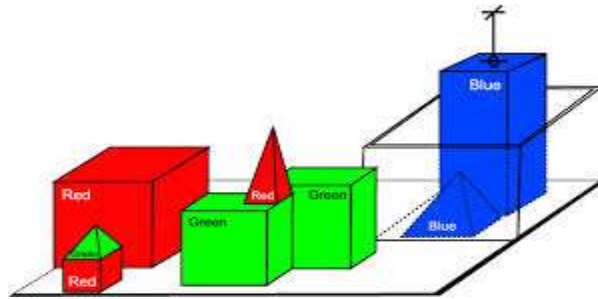


Figure 3: SHRDLU, an early [natural language understanding computer program](#)

### *Basic functions of lisp:*

- (LIST *S1 S2* ...)  
Form a list of the (evaluated) arguments *S1, S2* ...
- (MEMBER *S L*)  
Test whether *S* is a top-level (i.e. not embedded in a sublist) element of *L*.
- (APPEND *L1 L2* ...)  
Make a new list from the elements of *L1, L2, ...,* in order.
- (REVERSE *L*)  
Return a list containing the same elements as *L* but in reverse order.
- (LENGTH *L*)  
Returns the length of *L*, that is, the number of top-level elements in *L*.



```
[lisp - [Terminal]
File Edit View Project Tools Window Help

[1]> (reverse '(1 2 3 4 5 6))

(6 5 4 3 2 1)
[2]> (append '(1 2 3 4 (5 6 7) 8 9) '(1 2 (3) 4))

(1 2 3 4 (5 6 7) 8 9 1 2 (3) 4)
[3]> (list '(1 2 3 4 (5 6 7) 8 9) '(1 2 (3) 4))

((1 2 3 4 (5 6 7) 8 9) (1 2 (3) 4))
[4]> (member 'f '(a b c f g h j))

(T (S H J))
[5]> (length ("ab" "bc" "kfg"))

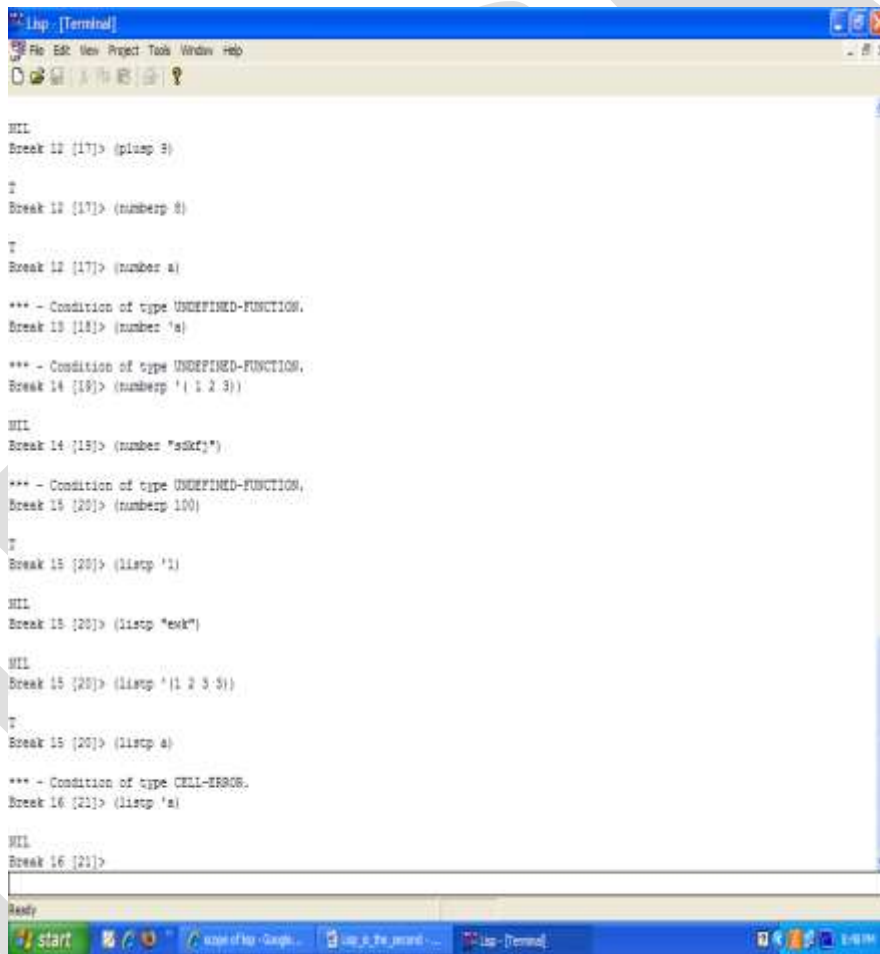
3
[6]>
```

Figure 4: Implementation of Basic functions of LISP

*Predicates (tests):*

- (LISTP *S*)  
True if *S* is a list.
- (NUMBERP *S*)  
True if *S* is a number.
- (NOT *S*)  
True if *S* is false, and false if *S* is true. Same as (NULL *S*).
- (EQUAL *S1 S2*)  
True if *S1* and *S2* are equal. Like EQ, but may be used for anything.
- (ZEROP *N*)  
True if number *N* is zero.

- (PLUSP  $N$ )  
True if number  $N$  is positive.
- (MINUSP  $N$ )  
True if number  $N$  is negative.
- (EVENP  $N$ )  
True if integer  $N$  is even.
- (ODDP  $N$ )  
True if integer  $N$  is odd.



```
lisp [Terminal]
File Edit View Project Tools Window Help

NIL
Break 12 [17]> (plusp 3)
T
Break 13 [17]> (minusp 8)
T
Break 12 [17]> (numberp a)
*** - Condition of type UNDEFINED-FUNCTION.
Break 13 [18]> (number 'a)
*** - Condition of type UNDEFINED-FUNCTION.
Break 14 [18]> (numberp '(1 2 3))

NIL
Break 14 [18]> (number "sdkf")
*** - Condition of type UNDEFINED-FUNCTION.
Break 15 [20]> (numberp 100)
T
Break 15 [20]> (listp '1)

NIL
Break 15 [20]> (listp "eskl")

NIL
Break 15 [20]> (listp '(1 2 3 5))
T
Break 15 [20]> (listp a)
*** - Condition of type CELL-ERROR.
Break 16 [21]> (listp 'a)

NIL
Break 16 [21]>

Ready
```

Figure 5: Predicates of LISP

*Arithmetic functions:*

- (+  $N1 N2 \dots$ )  
Returns the sum of the numbers.
- (-  $N1 N2 \dots$ )

Returns the result of subtracting all subsequent numbers from  $N1$ .

- $(* N1 N2 \dots)$

Returns the product of all the numbers.

- $(/ N1 N2 \dots)$

Returns the result of dividing  $N1$  by all subsequent numbers.

- $(1+ N)$

Returns  $N$  plus one. (Note that there is no space between the "1" and the "+".)

- $N)$

Returns  $N$  minus one. (Note that there is no space between the "1" and the "-".)

- $(/ N)$

Return the reciprocal of  $N$ .

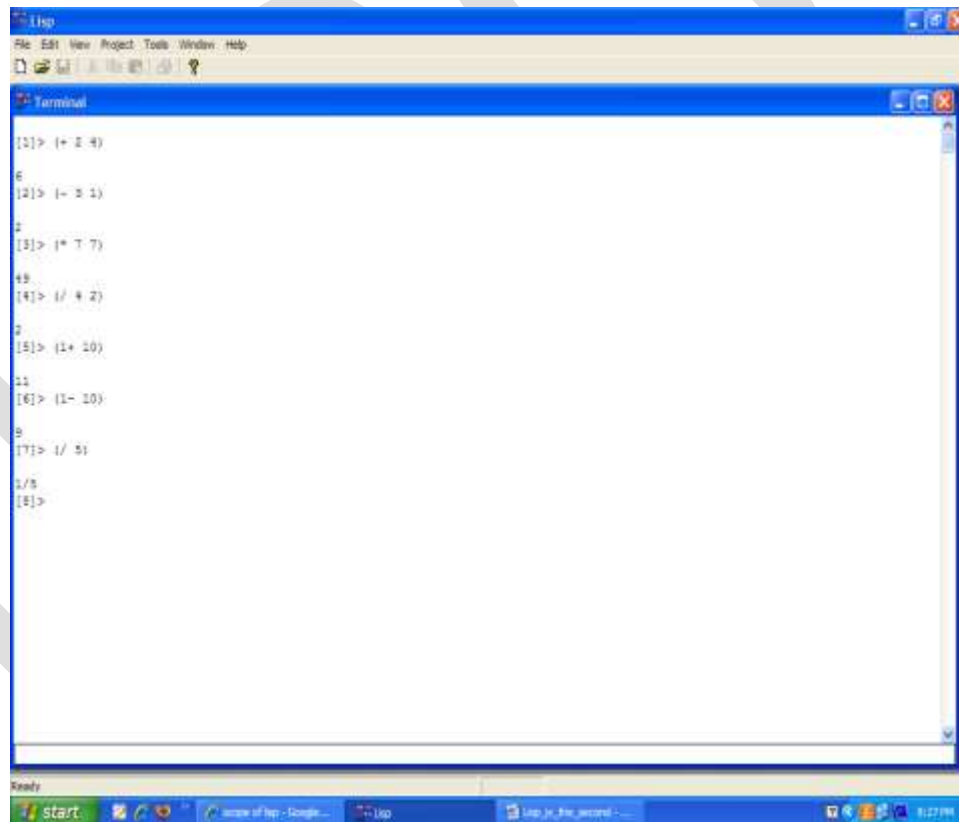


Figure 6: Arithmetic functions of LISP

**Input/output functions:**

- $(LOAD F)$

Load the source file whose name (without extensions) is  $F$ .

- (DRIBBLE *F*)

Causes the current session to be recorded file whose name (without extensions) is *F*. To stop recording, call (DRIBBLE) with no parameters. *Not available on all systems.*

- (PRIN1 *S*)

Print, on the current line, the result of evaluating the S-expression *S*.

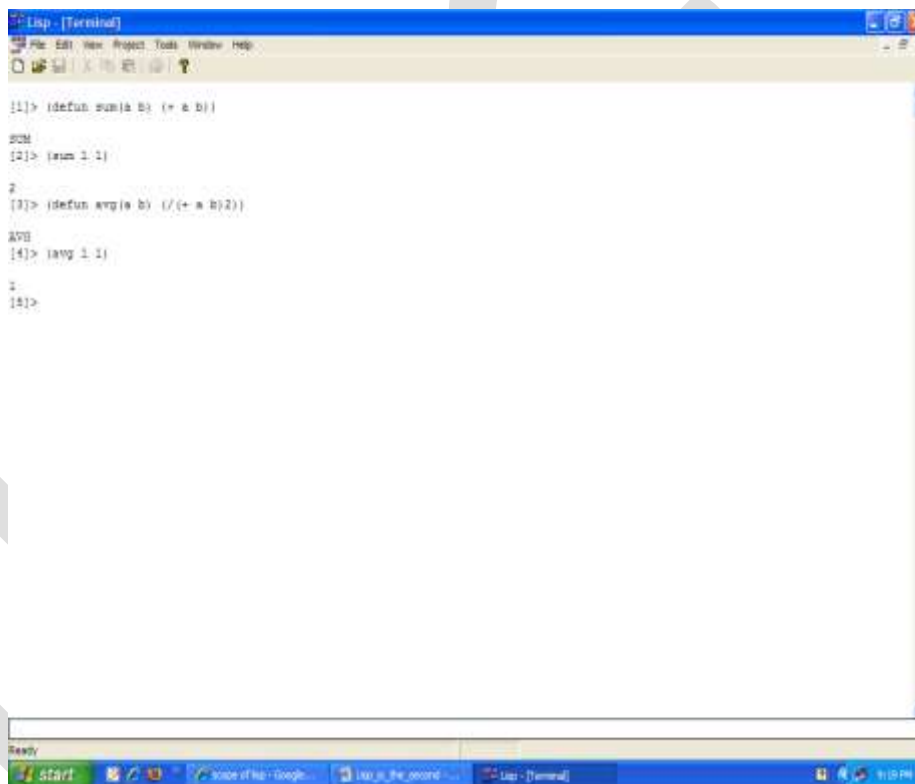
- (TERPRI)

Print a newline. [5]

### How to create user defines function:

(DEFUN function name parameter list function body)

Where function name is an identifier, parameter list is a list (possibly empty) of identifiers, and Function body is an S-expression. [6]



```
[Lisp - [Terminal]]
File Edit View Project Tools Window Help
[1]> (defun sum(a b) (+ a b))
SUM
[2]> (sum 1 1)
2
[3]> (defun avg(a b) (/ (+ a b) 2))
AVG
[4]> (avg 1 1)
1
[5]>
```

Figure 7: Creation of user defines function by defun.

### Scope of Lisp in AI programming:

The main programming languages used in AI are Lisp and Prolog. Both have features which make them suitable for AI programming, such as support for list processing, pattern matching and exploratory programming. [7]

Lisp can be viewed as the grandfather of functional programming

Case Studies of Lisp tell us that Lisp's flexibility allows it to adapt as programming styles change, but more importantly, Lisp can adapt to your particular programming problem. In other languages you fit your problem to the language; with Lisp you extend the language to fit your problem. A lot of AI programs deals with symbols and their manipulation, and Lisp are great for that because of its fast prototyping and the macro utility that helps you extend the language in a great way. Because of its flexibility, Lisp has been

successful as a high-level language for rapid prototyping in such areas as AI, graphics, and user interfaces. Lisp makes it easy to define new languages especially targeted to the problem at hand. Programmers love lisp because of the increased productivity it provides. The edit-test-debug cycle is so fast in Lisp. It made complex programs easy and fast to write. In Europe and Japan Prolog is the preferred choice while in America LISP is usually the way to go. [8]

Any programmer wants a simple, flexible language for programming, if we have a logic reasoning problem then go for Prolog. If on the other hand you have a space search problem that fits perfectly to Lisp list and symbols, go for Lisp.

### **Conclusion:**

Lisp has been hailed as the world's most powerful programming language. But only a few programmers use it because of its cryptic syntax. In this paper we have discussed several points related to lisp. On the basis of those points we can conclude that in AI field we can use LISP as a programming language. The task which is not fully logic based and instead of they are based on list and symbol can be programmed in LISP. This paper has discussed basic functions of LISP, with its application, features and scope. Hence we can conclude that LISP can be assumed as an important language in AI field.

### **REFERENCES:**

- [1] <http://www.tutorialspoint.com/lisp/>.
- [2] [http://www.tutorialspoint.com/lisp/lisp\\_overview.htm](http://www.tutorialspoint.com/lisp/lisp_overview.htm).
- [3] [http://en.wikipedia.org/wiki/Lisp\\_\(programming\\_language\)#Connection\\_to\\_artificial\\_intelligence](http://en.wikipedia.org/wiki/Lisp_(programming_language)#Connection_to_artificial_intelligence).
- [4] <http://en.wikipedia.org/wiki/SHRDLU>.
- [5] <http://www.cis.upenn.edu/~matuszek/LispText/lisp-morefun.html>.
- [6] <http://www.cis.upenn.edu/~matuszek/LispText/lisp-write.html>.
- [7] <http://www.go4expert.com/articles/artificial-intelligence-lisp-t274/>.
- [8] <http://cseweb.ucsd.edu/classes/sp00/cse151/q1/crocha.html>.