C.W
Tuesday

Date 7/03/23
51

CHAPTER-4
CLISP

CLISP – Common LISP

- Founded in 1950s by John McArthy (father of AI)

The term AI was coined in dartmouth in a seminar which John McArthy was presiding.

LISP : LISP processing

Atom comprises of strings. They are basic building blocks of LISP.
String means collection of contiguous characters, numbers, etc.
~~You may have~~
String: Collection of characters within double quotes.
Maybe integer, special characters also.

LISP (collection of Atoms & Lists) | We may have list within list.

Every list, atom & string is called sExpression. An sExpression in itself is a complete program.

- LISP uses prefix Operation.

T → True | NIL → NIL / ()
↓
List as well as Atom

String → Constant value
If string is in lowercase, we get output in uppercase.

- LISP is a symbol processing language
- LISP lack data types

C.W
Tuesday . Friday

10/03/22

* **Arithmetic Operation**

(+ 34 29) → (LIST)

0/P 57 — Operand Atom, Integer type

(+ 28 -9 1.7 17) → 29.7 (Negative no., floating point, Integer)

(+ (+ 4 4.5) 34) → 42.5

Arithmetic Addition: Arithmetic Addition take any number of arguments.

(- 34 56)
-22

> '+ 2.3 ← Makes a list does not add.

(quit) (exit) → to close

> +0 = O/P → 0
> * = O/P → 1
> (+ (+) 9 2) → 11
> (+ (*) 9 2) → 12
> (+ 9 2) → 11
> (- 9 54) → 0
> (- 99 5 3 6 7) → -21
> (/ 23 4) → 23/4
> (/ 23.0 4) → 5.75
> (- 4.0 13) → -9.0
> (/ 23 4 5 7 9 13) → 23/16380 = 0.0014041
> (* 2 2.3 3) → 13.7999?

List Manipulation function:
* → setq (xyz, 10) → assigns 10 to variable (xyz)

Wednesday

15/02/23

Basic List Manipulation functions

* Car - (Content Address Register)
  Returns the first element of the list.

Ex:
> (car '(abc (a b) c e))
O/P → ABC

' → changes a group of literals into list.

> (car '(a b c))
O/P → A

* Cdr → Content Decrement Register)
> (cdr '((a b) (d) 2 4)) → O/P (D) 2 4)

↳ Returns a list after removing the topmost element.

* Cons → Constructed Memory Object
  → It takes 2 arguments.

> Cons 'a '((a b) 2 3))
  element list

O/P → (A (A B) 2 3)

* List → Any number of arguments
  → Arguments can be atom or list.
  → Forms a list

(list 'a '(a c) '(1 2) 4)

O/P → (A (A C) (1 2) 4)

* append - Each argument should be a list.
- Forms all arguments in a single list

>append '(1 2) '(3) '(A B))
O/P → (1 2 3 A B)

* last - Argument should be a list.
- Only one argument
- Returns last element of the list in a list.

>(last '(1 (2) (2 3) 4 6 '(A))
O/P → ((A))

* member → Takes two argument
→ scans the first element in second argument list.
→ Returns a list of elements from the point a match is found in a list.

>(member 'b '(a b c))
O/P (B C)

* reverse → Takes one argument.
→ Returns a reversed list.

>(reverse '(a b (1 2) 6 (D))
O/P ((D) 6 (1 2) B A)

> (cons '(* 2 3) '(1)) → ((* 2 3) 1)

> (cons (* 2 3) '(1)) → (6 1)

> (setq x '(a b c)) → (A B C)

> ~~(setq x)~~ 'x → X

> x → A B C

> (car (cdr 'abcd)) → B

> (cdr (car '((a b) (c d)) → (B)

> (cons 'one '(two three)) → ONE TWO THREE

> (cons (car '(a b c)) (cdr '(ab c)) (A (B C))

> (list '(a b) 'c 'd) → ((A B) C D)

> (append '(a (b c)) '(d e)) → (A (B C) D E)

> (append '(a) '(b c) '(d)) → (A BC D)

> (~~append~~ last '(a b (c d) (e))) → ((E))

> (reverse '(a b (c d) e)) → (E (C D) B A)

C.W

Tuesday

CLISP 2.49 → greaterP / lessP (don't work)

21/5

> (defun s1 (a b c) (+ a b c))

→ It creates a function for user
→ Syntax: (defun name (parameters) (funcn))
function

> (s1 10 20 30)
60

> (defun avg (a b c) (/ (+ a b c) 3))

we can also calculate square root using (sqrt x)

Predicate functions

> (atom 12)
T

> (greaterP 2 4)
NIL

> (lessP 2 4)
T

> (evenP 3)
NIL

> (oddP 4)
NIL

> (numberP abc)
NIL

> (numberP 10)
T

> (zeroP 0.001)
NIL

> (equal a (car '(a b c)))
T

> (listP 12)
NIL

> (null '(a b c))
NIL

> (null ())
T

> (null NIL)
T

CLISP → PI

>(max 10 20 30)
30

>(min 10 20 30)
10

car → first ( )
↑
Predicate

cdr → rest ( )
↑
Predicate.

> (< 2 4 3)
NIL
↳ 2<4 (T) & 4<3 (NIL)

>(< 2 4)
T

(> 2 4)
NIL

→ If version is given CLISP 2.49 & greaterP & lessP is used then print Error.

Q)1 Write a user defined function to calculate area of ~~the~~ triangle (right angle)

Q)2 Write a user defined function to calculate simple ~~int~~ interest of a value

Q)3 Write a user defined function to calculate area of a circle.

1) (defun triArea (height length) (/ (* length height) 2))

2) (defun sInter (prinp rt t) (/ (* prinp rt t) 100))

3) (defun crtArea (radius) ((* r r PI))

C.W
Wednesday

* CONDITIONALS

↳ cond, if

> (defun max2 (a b))

(cond ((> a b) a) → If true it will return a Otherwise b
(T b)))

MAX 2

> (MAX 2 10 5)

10

(if (> a b) a b))

For greater of three numbers.

(if (> a b) (if (> a c) a c))

LOGICAL FUNCTIONS

↳ NOT, OR & AND

[NOT/nOT → are same]

* (NOT (parameter [listp '(a b c)]))
only one

> (nOT (listp '(a b c)))

NOT
T → NIL
NIL → T

* (OR (any number of argument))

Scanning of arguments left to right. First NON NIL value will be returned

> (OR (list-P) (+ 3 5) (* 2 3 6) (atom 2))
NIL 8

↳ 8 ← output

* (AND (Any number of parameters)) → Scanning left to right → Last non NIL value returned.

→ (AND (listp '(a b c)) (+3 5) (atom 4) (* 5 6))

⇒ 30

⇒ (AND (listp 'a) (+3 5) (atom 4) (*5 6))

= NIL

* Taking User Input

→ read

> (+ (read) (read))

4

5

→ 9 output

> (+ 5 (read))

4

9 → output

* Printing output

↳ print

print "Hello"

"Hello"

"Hello"] o/p

↳ princ

(princ "Hello")

Hello

"Hello"] o/p

↳ without double quotation

* printing new line

↳ (terpri)

(defun Print_Terpri_Princ ( )

(princ "Vaibhav") (princ "Kant") (princ "Kant") (princ "Singh")

(terpri) (princ "Aghar") (princ "Khan") (print "XYZ"))

~~Vaibhav Kant Singh~~