THE MANGA GUIDE TO LISP

SUZU

WAKANA

RIKKA

LICO

ORIGINAL: BUGYO
EDITOR: WEBM
TRANSLATOR: GLITTERGIRL


UMA
RESEARCH CLUB
JOIN
NOW!
SUZU! COME HELP ME HANG THESE UP!


I'M LEAVING..
PAY ATTENTION, OKAY?


YEAH!
IT'S BECAUSE YOU HAVE THE HIGHEST UMA ENCOUNTER RATES!
WHAT? ME?


MIRIN
MIRIN...


UMA
MEOW!


HUH?
MEOOOW!
WAAA!
MEOW!
HYA~
ゴ
I'M FINE
THAT WAS A LOUD NOISE THOUGH, RIGHT?
UHHH..
ARE YOU OKAY?


BUT...
IT'S JUST A SCRATCH!
OKAY...
I'M FINE!
CAN I PET IT?
YEAH..
MIRIN? THE COMMUNAL CAT?
BY THE WAY, IS THAT YOUR CAT?
..IS IT BECAUSE YOU HIT YOUR HEAD BACK THERE?
NO!
SUZU! THE WORLD IS IN DANGER! WE HAVE TO DO SOMETHING!
BUT EXCUSE ME! I'M LICO.
AH, NICE TO MEET YOU LICO. I'M SUZU

TRANSLATORS NOTE: I HAVE NO IDEA.

THE MANGA GUIDE TO LISP

INTRODUCTION
WHILE LISP HAS MANY MEANINGS, HERE I INTEND TO COVER COMMON LISP. NONETHELESS, THE FOUNDATIONS OF THE DIFFERENT KINDS OF LISPS ARE SIMILIAR, SO PLEASE DON'T SAY THINGS LIKE "I WANT TO DO SCHEME, SO I'LL LEAVE" > <

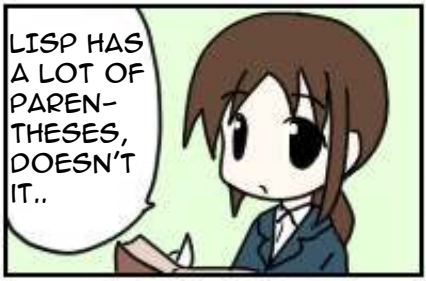
THE REST OF THIS PAGE WILL BRIEFLY COVER HOW TO INSTALL A LISP IMPLEMENTATION. IF YOU'VE ALREADY PREPARED YOUR ENVIRONMENT, PLEASE SKIP THIS SECTION.

IMPLEMENTATION INSTALLATION
THERE ARE MANY IMPLEMENTATIONS OF COMMON LISP. THE ONLY ONE I RECOMMEND TO BEGINNERS IS CLISP, BECAUSE IT RUNS ON MANY PLATFORMS, AND HAS AN EASY-TO-USE INTERACTIVE ENVIRONMENT. ON WINDOWS, YOU CAN DOWNLOAD THE BINARY FROM THE WIN32 LINK ON THE OFFICIAL SITE [1]. ON MAC OS X, *BSD, LINUX, ETC. YOU SHOULD BE ABLE TO INSTALL THEM FROM YOUR PACKAGE MANAGER.

TO THE PEOPLE WHO SAY "IN ANY CASE, I WANT TO WRITE HIGH-SPEED RUNNING PROGRAMS!", I RECOMMEND SBCL. IF YOU TRY YOUR BEST, YOU CAN WRITE PROGRAMS WITH SPEED THAT WON'T EVEN LOSE TO C. FOR MORE DETAILS, PLEASE SEE THE OFFICIAL SITE [2].

[1] HTTP://CLISP.CONS.ORG
[2] HTTP://WWW.SBCL.ORG

PARENTHESES ARE FRIENDS



DOES EVERYBODY LIKE PARENTHESES? LISP PROGRAMS HAVE A GREAT NUMBER OF PARENTHESES APPEARING IN THEM. AS OPPOSED TO THE MANY PEOPLE WHO ACCEPT PYTHON AND HASKELL, WHICH HAVE FEW PARENTHESES AND MEANINGFUL INDENTATION, THERE ARE UNFORTUNATELY MANY WHO REJECT LISP WITH ITS CONSPICIOUS AMOUNT OF PARENTHESES. THE APPEARANCE OF MANY PARENTHESES MIGHT BE DIFFICULT TO APPROACH, BUT IN FRIENDS, CONTENT NOT APPEARANCE IS WHAT'S IMPORTANT.

```
;; A PROGRAM THAT SEARCHES
;; FOR THE FACTORIAL OF N.
;; AREN'T THERE MANY BUSY
;; FRIENDS?
(DEFUN FACT(N)
 (COND ((= N 0) 1)
       (T (* (FACT (1- N))
             N))))
```

DECREASING FRIENDS

SINCE MANY LISP IMPLEMENTATIONS PROVIDE AN INTERACTIVE ENVIRONMENT, YOU CAN TEST LISP RIGHT AWAY. JUST WRITE THE PROGRAM YOU WANT TO RUN AND PRESS ENTER. IT'S EASY, DON'T YOU THINK? IF YOU INPUT (+ 1 1), 2 WILL COME OUT. IF YOU INPUT (+ 1 2 3), 6 WILL COME OUT. (+ (+ 1 2)3) IS ALSO THE SAME, BUT IT'S EASIER TO READ WITH LESS PARENTHESES. MORE FRIENDS ISN'T ALWAYS BETTER, RIGHT?

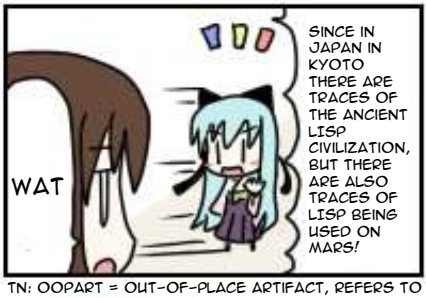
```
;; WHEN CALCULATING
;; 1+2+3, THERE ARE TWO
;; SITUATIONS TO
;; CONSIDER: ADDING 1 AND 2
;; FIRST, AND ADDING 2 AND 3
;; FIRST.

;; CALCULATING (1+2)+3
(+ (+ 1 2) 3)
;; CALCULATING 1+(2+3)
(+ 1 (+ 2 3))

;; WHEN THE SUM IS THE
;; SAME REGARDLESS
;; OF WHERE YOU
;; START FROM, YOU
;; CAN WRITE IT LIKE THIS
(+ 1 2 3)
```

INLAND SEA

YOU CAN FREELY TRY EASY CALCULATIONS LIKE THIS. 5*4/2 BECOMES (/(*5 4) 2). IN LISP YOU CAN ALSO USE FRACTIONS. YOU CAN CALCULATE 8/6 AS (/ 8 6), AND THE RESULT IS 4/3 (IT WILL REDUCE A FRACTION TO ITS LOWEST TERMS!). BY THE WAY, IF YOU TRY AND DO THE CALCULATION THAT SUZU DID (88..89 * 99..99), YOU CAN GET A GOOD FEELING.

```
;; * SINCE EVERYTHING FROM ; TO
;; THE END OF THE LINE IS
;; IGNORED, IT ADDS THE EFFECT
;; OF THE PROGRAM ON THE SIDE.
;; (IT'S OKAY TO NOT WRITE
;; ANYTHING AFTER THE ';' )
;; 1+1/3:
(+ 1 (/ 3 3)) ; => 4/3

;; USING DECIMALS
;; (NOT FRACTIONS):
(FLOAT (+1 (/ 1 3))) ; => 1.34
;; THE SQUARE ROOT OF TWO:
(SQRT 2) ; => 1.4142135

;; YOU CAN ALSO USE
;; COMPLEX NUMBERS:
(SQRT -1) ; => #C(0 1)
```

ULTRA ANCIENT CIVILIZATION

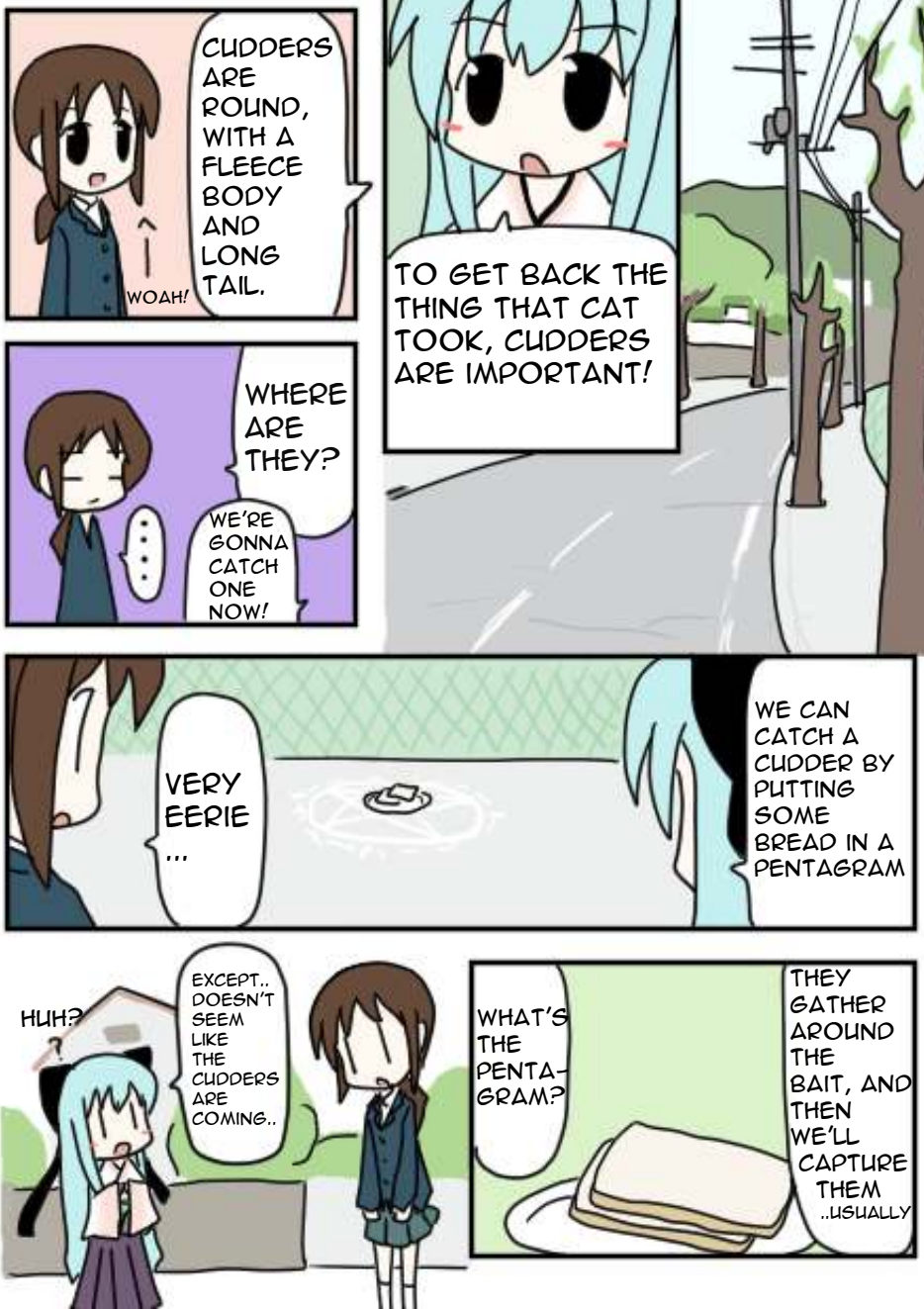


FOR HISTORICAL REASONS, THERE ARE MANY INTRODUCTIONS OF LISP AS "A LANGUAGE FOR AI*", BUT ACTUALLY YOU CAN USE IT FOR MANY DIFFERENT PURPOSES. THERE ARE ALSO PEOPLE WHO SAY "LISP IS ONLY USED BY ONE TYPE OF PERSON", BUT IN FACT AN EXPANSIVE CLASS, FROM MIDDLESCHOOLERS TO OLD PEOPLE USE IT. AH, IF YOU KNOW ANY ELEMENTARY SCHOOLERS WHO ARE USING LISP, PLEASE GET IN TOUCH.

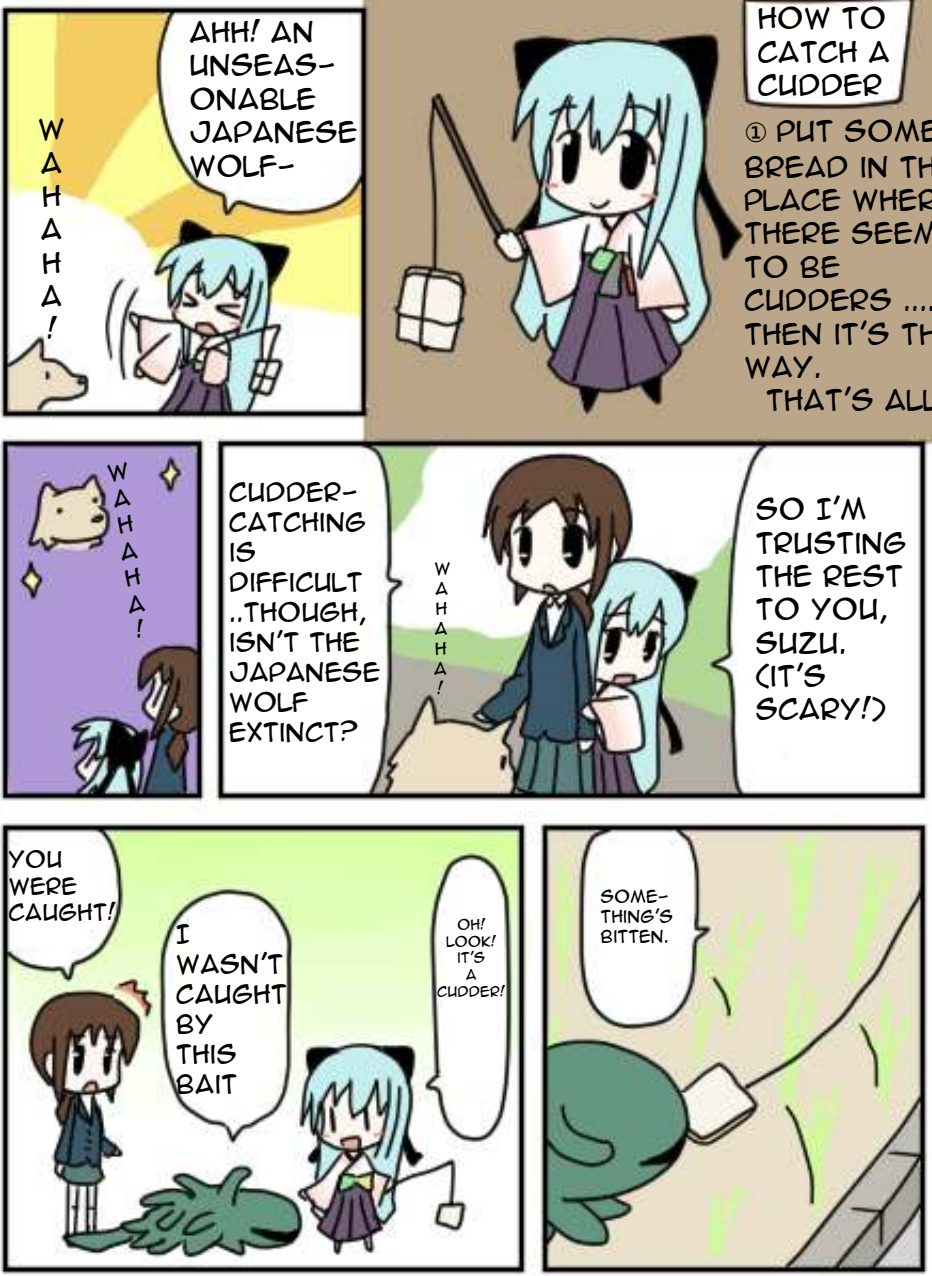
*NOTE: AI STANDS FOR 'ARTIFICIAL INTELLIGENCE'

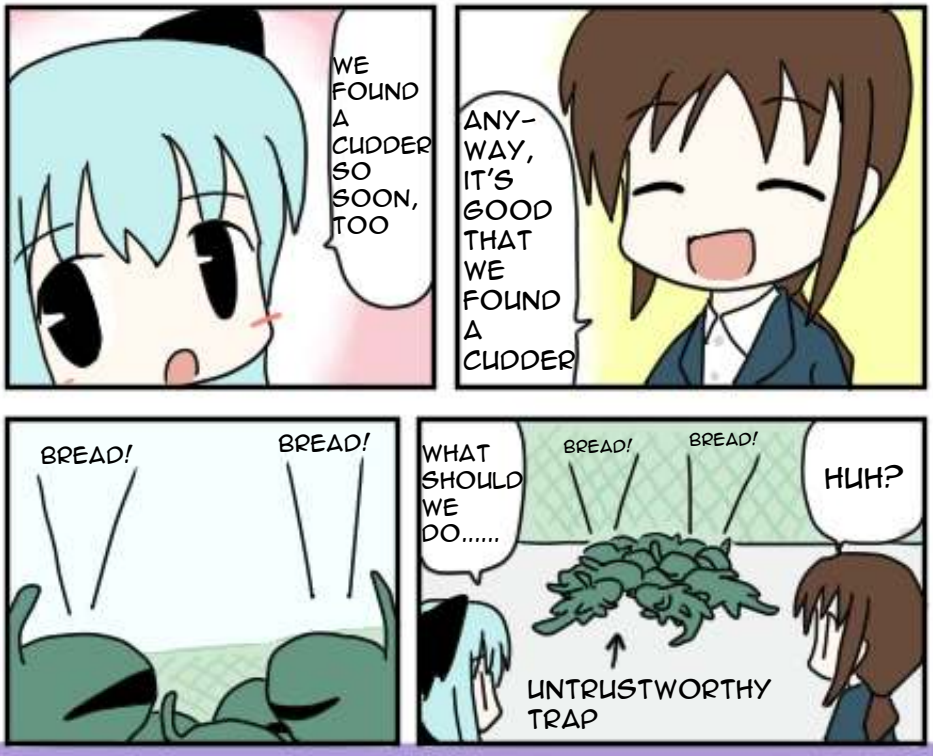
```
;; CALCULATIONS WITH FRACTIONS
;; ARE EXACT:
(+1/3 1/3 1/3)  ; => 1

;; CALCULATIONS
;; WITH DECIMALS
;; HAVE MEASUR-
;; EMENT ERROR:
(+ 0.3 0.3 0.3) ; => 0.9000004

;; THE RESULT OF DIVIDING WHOLE
;; NUMBERS IS A FRACTION:
(* (/ 1 123) 123) ; => 1

;; THE RESULT OF DIVIDING
;; DECIMALS IS A DECIMAL
(* (/ 1.0 123.0) 123)
; => 0.99999994
```

TN: OOPART = OUT-OF-PLACE ARTIFACT, REFERS TO A HISTORICAL OBJECT FOUND IN AN UNUSUAL CONTEXT "THAT COULD CHALLENGE CONVENTIONAL HISTORICAL CHRONOLOGY".

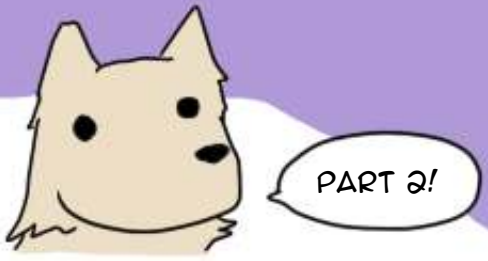

CUDDERS ARE ROUND, WITH A FLEECE BODY AND LONG TAIL.
WOAH!
TO GET BACK THE THING THAT CAT TOOK, CUDDERS ARE IMPORTANT!
WHERE ARE THEY?
WE'RE GONNA CATCH ONE NOW!
VERY EERIE ...
WE CAN CATCH A CUDDER BY PUTTING SOME BREAD IN A PENTAGRAM
HUH?
EXCEPT.. DOESN'T SEEM LIKE THE CUDDERS ARE COMING..
WHAT'S THE PENTA-GRAM?
THEY GATHER AROUND THE BAIT, AND THEN WE'LL CAPTURE THEM ..USUALLY


car…
カー
カー
DON'T GIVE UP!
I MADE A SPELLING MISTAKE?!
I'M STARTING TO NOT REALLY TRUST LICO..
SINCE WE CAN'T REALLY TRUST THE TRAP, LET'S JUST GO FIND ONE!


WAHAHA!
AHH! AN UNSEASONABLE JAPANESE WOLF-
HOW TO CATCH A CUDDER
① PUT SOME BREAD IN THE PLACE WHERE THERE SEEMS TO BE CUDDERS .... THEN IT'S THIS WAY.
THAT'S ALL!
WAHAHA!
CUDDER-CATCHING IS DIFFICULT ..THOUGH, ISN'T THE JAPANESE WOLF EXTINCT?
WAHAHA!
SO I'M TRUSTING THE REST TO YOU, SUZU. (IT'S SCARY!)
YOU WERE CAUGHT!
I WASN'T CAUGHT BY THIS BAIT
OH! LOOK! IT'S A CUDDER!
SOME-THING'S BITTEN.



THE MANGA GUIDE TO LISP



PROGRAMS ARE EXPRESSIONS
IN LISP, PROGRAMS (OR MORE CORRECTLY, PARTS OF PROGRAMS) ARE CALLED "EXPRESSIONS". UNTIL YOU ARE EXPERIENCED, IT MIGHT BE GOOD TO REPLACE "EXPRESSION" WITH "PROGRAM" WHILE READING. THE IMPORTANT PROPERTY OF EXPRESSIONS IS THAT THE PARTS COMPOSING AN EXPRESSION ARE ALSO EXPRESSIONS (SUBEXPRESSIONS). SINCE (+ 1 (+ 2 3) IS AN EXPRESSION, ONE OF ITS PARTS, (+ 2 3), IS ALSO AN EXPRESSION. MOREOVER, EVEN 1 AND 2 ARE EXPRESSIONS.

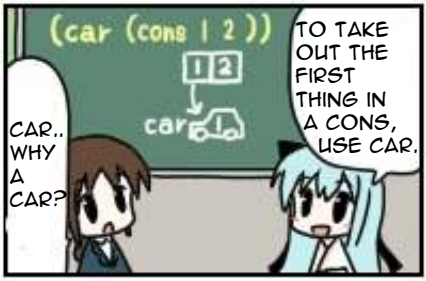
EXECUTION IS EVALUATION
IN LISP, RUNNING A PROGRAM IS REFERRED TO AS "EVALUATING AN EXPRESSION". IN OTHER WORDS, RUNNING THE PROGRAM (+ 1 2) WOULD BE CALLED "EVALUATING THE EXPRESSION (+ 1 2)".

THE RESULT OF EXECUTION IS THE VALUE OF AN EXPRESSION
IF YOU EVALUATE THE EXPRESSION (+ 1 2), YOU WILL GET THE RESULT 3. THIS RESULT OF EXECUTING THE PROGRAM IS CALLED "THE VALUE OF THE EXPRESSION (+ 1 2)".

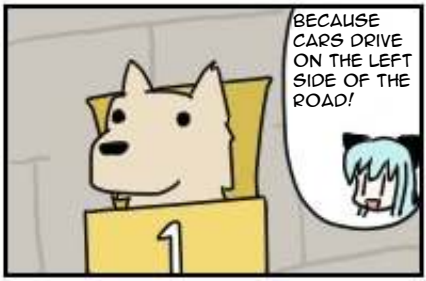
SUBEXPRESSIONS ALSO HAVE VALUES
EXPRESSIONS HAVE VALUES, BUT SUB-EXPRESSIONS ALSO HAVE VALUES. CONSIDERING THE EXPRESSION (+ 1 (+ 2 3)), THE SUBEXPRESSION (+ 2 3) HAS THE VALUE 5, AND THE EXPRESSION 1 HAS THE VALUE 1. IF YOU INPUT 1 INTO YOUR INTERACTIVE ENVIRONMENT, YOU CAN CONFIRM THAT THE VALUE OF 1 IS 1.

AMERICAN

A CONS IS A BOX WITH TWO ITEMS IN IT. THE EXPRESSION (CONS 1 2) MAKES A CONS WITH 1 AND 2 IN IT. THIS DISPLAYS AS (1 . 2). THOUGH THERE ARE PARENTHESES HERE, PLEASE NOTE THAT THIS IS NOT AN EXPRESSION (PROGRAM), BUT A VALUE (RESULT OF EXECUTION). THE LEFT SIDE OF AN EXPRESSION IS CALLED THE CAR, AND YOU CAN TAKE ITS ITEM OUT USING CAR.



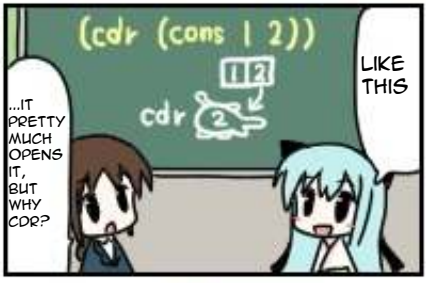
```
;; MAKE A CONS
(CONS 1 2) ; => (1 . 2)

;; TAKE OUT THE CAR OF A CONS
(CAR (CONS 1 2)) ; => 1
```




```
;; (1 . 2) IS NOT A VALID
;; EXPRESSION
(1 . 2) ; => ERROR!
```

MANY CAR ACCIDENTS

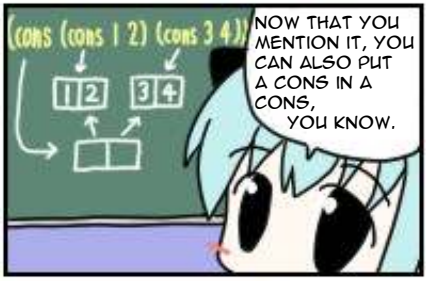
THE RIGHT SIDE OF A CONS IS CALLED THE CDR, AND YOU CAN TAKE OUT ITS ITEM USING CDR. SINCE THE CDR OF THE CONS IS MADE WITH (CONS 1 2) IS 2, THE VALUE OF (CDR (CONS 1 2)) IS 2. INCIDENTALLY, THERE IS AN URBAN LEGEND THAT "CAR" IS AN ABBREVIATION OF "CONTENTS OF THE ADDRESS PART OF REGISTER", AND "CDR" IS "CONTENTS OF THE DECREMENT PART OF THE REGISTER".



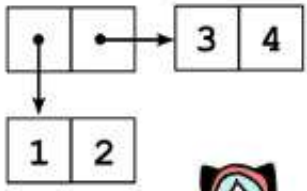

```
(CDR (CONS 1 2)) ;=>2

(CONS (CDR (CONS 1 2)) 3)
; => (2 . 3)

(CDR
  (CONS (CDR (CONS 1 2)) 3))
; => 3
```


```
(CAR (CONS 1 2)) ; => 1

(CAR
  (CONS (CAR (CONS 1 2)) 3))
; => 1
```

MATRYOSHKA



IF YOU EVALUATE EXPRESSIONS LIKE (CONS (CONS 1 2) (CONS 3 4)), YOU CAN MAKE CONSES WITH CONSES IN THEM. OF COURSE, YOU CAN ALSO MAKE CONSES WITH CONSES WITH CONSES. IF ITS HARD TO UNDERSTAND, IT MIGHT BE GOO TO LOOK AT THIS PICTURE:



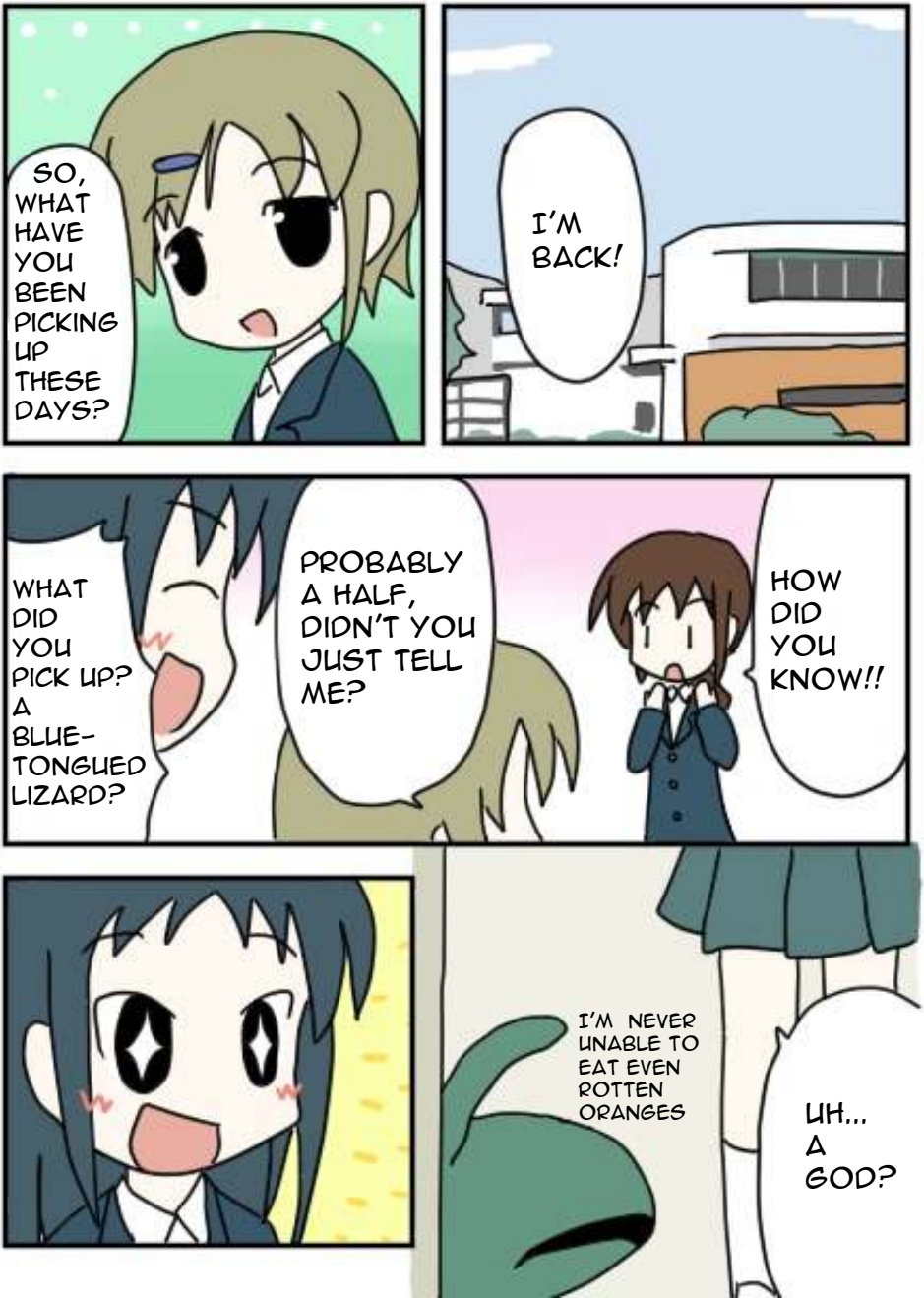
```
(CAR (CONS (CONS 1 2)
           (CONS 3 4)))
; => (1 . 2)

(CDR (CONS (CONS 1 2)
           (CONS 3 4)))
; => (3 .4)

(CDR (CDR (CONS (CONS 1 2)
                (CONS 3 4))))
; => 4
```

CAT SENPAI

WHITE, LIKE A WHITE TIGET. WHITE, LIKE A BLANK SPACE. IN THE LAST SECTION, WE USED AN EXPRESSION WRITTEN ON MULTIPLE LINES. SINCE LISP TREATS MULTIPLE CONSECUTIVE SPACES, TABS, OR NEWLINES AS ONE SPACE, WHEN THERE IS A LOT OF STUFF BETWEEN PARENTHESES YOU CAN ADD SOME NEWLINES TO MAKE IT EASIER TO READ.

AEROBRAKING!

```
;; CALCULATING (1+2)*(3-4)*(5+6)
;; ON ONE LINE
(* (+ 1 2) (- 3 4) (+ 5 6))

;; ON MULTIPLE LINES
(* (+ 1 2)
   (- 3 4)
   (+ 5 6))
```


SO, WHAT HAVE YOU BEEN PICKING UP THESE DAYS?
I'M BACK!
WHAT DID YOU PICK UP? A BLUE-TONGUED LIZARD?
PROBABLY A HALF, DIDN'T YOU JUST TELL ME?
HOW DID YOU KNOW!!
I'M NEVER UNABLE TO EAT EVEN ROTTEN ORANGES
UH... A GOD?


THAT'S NOT AN EVIL GOD, IT'S A CUDDER!


AMA-ZING! A GREEN TENT-ACLE GOD!
WHAT PLACE ARE YOU THE EVIL GOD OF?
A WORD FOR A PER-SON.

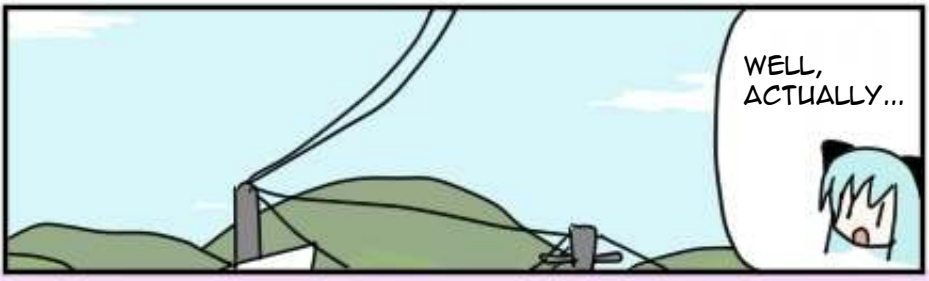

WHO ARE YOU?


WELL, SINCE IT'S HALF, RIGHT?


HOW DO YOU KNOW LICO?
OH, NICE TO MEET YOU, GOD.


A GOD CAME!
AMAZING! A LITTLE GIRL AND GOD! I WONDER WHAT BLESSING THIS IS?
VERY CALM


DOBON!!
ALL RIGHT! ONE CARD LEFT!
ONE STEP AWAY!
I WONDER IF IT'S BECAUSE SHE'S THAT KIND OF KID?
..HOW WERE BOTH CARDS 13..
SWEET! DOBON RETURNED!
OKAY, OKAY, DON'T GODS EAT TOFU?
.....



THE MANGA GUIDE TO LISP

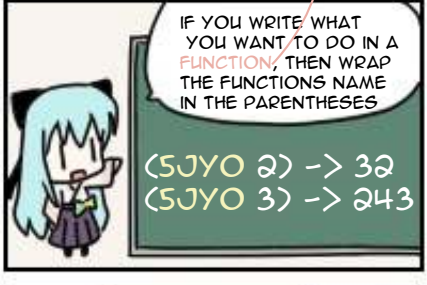
PART 3!

ERRAND-RUNNER

WHEN YOU WANT TO DO THE SAME CALCULATIONS OVER AND OVER, YOU SHOULD MAKE A FUNCTION. FUNCTIONS ARE MADE LIKE THIS:

```
(DEFUN FUNCTION-NAME
  (FUNCTION-ARGUMENTS)
    (WHAT-YOU-WANT-TO-DO))
```

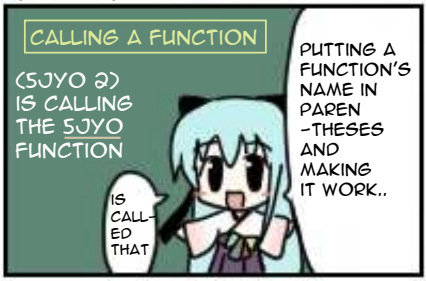
TN: 'SJYO' MEANS 5TH POWER

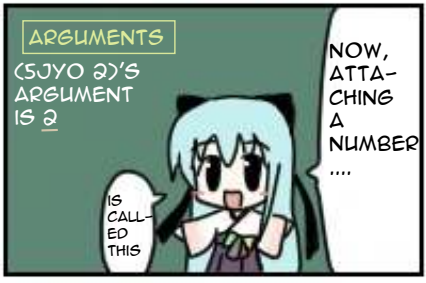


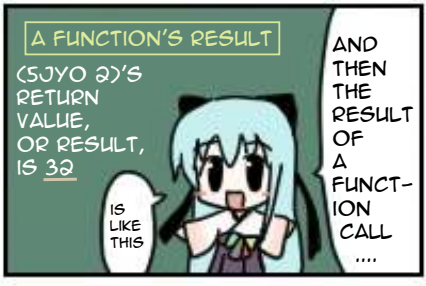
WHEN YOU PUT THE FUNCTIONS NAME IN PARENTHESES, "WHAT-YOU-WANT-TO-DO" WILL BE EVALUATED. WE'LL GIVE A MORE DETAILED EXPLANATION LATER, BUT LET'S BRIEFLY LOOK AT DIFFERENT FUNCTIONS

```
;; AREA OF SQUARE FUNCTION
(DEFUN SQARE (X)
    (* X X)
;; AREA OF TRIANGLE FUNCTION
(DEFUN TRIANGLE (W H)
    (/ (* W H) 2))
;; AREA OF CIRCLE FUNCTION
(DEFUN CIRCLE (R)
    (* R R PI))

;; USING THE ABOVE FUNCTIONS:
(SQUARE 4) ; => 16
(TRIANGLE 5 6) ; => 15
(CIRCLE 2) ; => 12.566
```

INVOKING







LET'S LOOK AGAIN AT HOW TO MAKE A FUNCTION.

```
(DEFUN FUNCTION
  (PARAMETER-1 PARAMETER-2...)
    FUNCTION-BODY)
```

NEXT, FUNCTION INVOCATION:

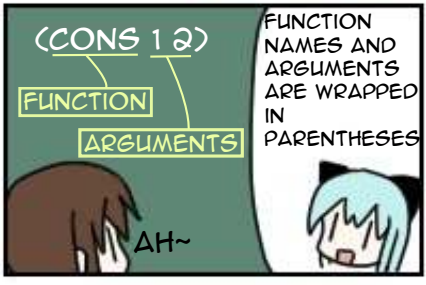
```
(FUNCTION
    ARGUMENT-1 ARGUMENT-2...)
```

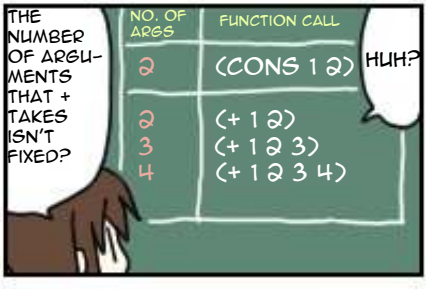
WHEN YOU EVALUATE THIS EXPRESSION, FIRST THE ARGUMENTS ARE EVALUATED. THEN, THE ACTUAL FUNCTION IS CALLED, AND THE FUNCTIONS BODY IS EVALUATED. WHEN THIS HAPPENS, THE VALUES OF THE PARAMETERS BECOME THE VALUES OF THE ARGUMENTS.

```
;; MAKE A FUNCTION CALLED ADD
;; ADD HAS PARAMETERS X AND Y
(DEFUN ADD (X Y)
    (+ X Y)) ; => ADD

;; PASS THE PARAMETERS 1 AND 2
;; TO ADD IN AN ENVIRONMENT
;; WITH X'S VALUE BEING 1, AND Y'S
;; BEING 2, (+ X Y) IS EVALUATED,
;; AND THE FUNCTION RETURNS 3
(ADD 1 2) ; => 3
```

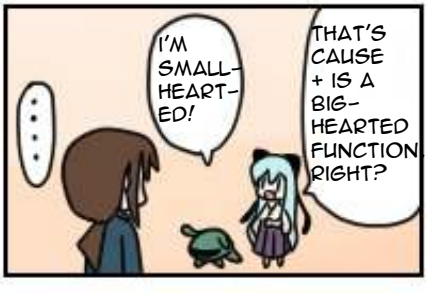
OCTOPUS ALSO LIVE HERE

THE THINGS APPEARING SO FAR: +, -, *, /, FLOAT, SQRT, CAR, AND CDR ARE ALL FUNCTIONS. MANY FUNCTIONS TAKE A FIXED NUMBER OF ARGUMENTS. FOR EXAMPLE, CONS TAKES 2, AND CAR AND CDR EACH TAKE ONE. HOWEVER, + AND * TAKE 0 OR MORE, AND - AND / TAKE AT LEAST ONE ARGUMENT. THE DETAILS OF MAKING FUNCTIONS LIKE THIS WILL COME LATER.



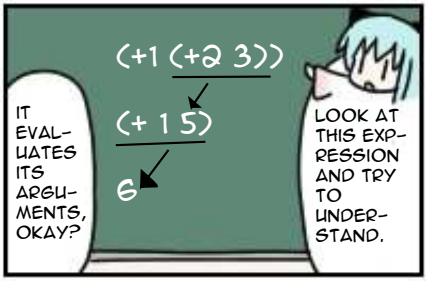



```
;; A SLIGHTLY DIFFERENT EXAMPLE:
(+)     ; => 0
(*)     ; => 1
(- 2) ; => -2
```




```
;; FUNCTION TAKING 1 OR 2 ARGS:
(DEFUN INC (X &OPTIONAL (Y 1))
    (+ X Y)) ; => INC

;; PASSING 1 OR 2 ARGUMENTS:
(INC 5) ; => 6
(INC 6 7) ; => 13
```

SOMETHING SOMETHING



IN FUNCTION CALL EXPRESSIONS, THE ARGUMENTS (IN ORDER FROM THE LEFT) ARE EVALUATED, BUT IN DEFUN EXPRESSIONS THE ARGUMENTS ARE NOT EVALUATED. THIS IS BECAUSE DEFUN IS NOT A FUNCTION. OTHER THAN FUNCTIONS, THE THINGS THAT ARE ENCLOSED IN PARENTHESIS INCLUDE SPECIAL OPERATORS AND MACROS. THESE LATTER TWO THINGS DON'T EVALUATE THEIR ARGUMENTS. DEFUN IS A MACRO.

```
;; THE FUNCTION PRINT,
;; WHICH DISPLAYS ITS
;; ARGUMENT AND THEN
;; RETURNS IT IN
;; ADDITION TO ITS
;; VALUE, IT PRINTS 3
(PRINT (+ 1 2) ; => 3

;; SEE THAT 3 AND 7 ARE DISP-
;; LAYED BY PRINT, AND THE
;; ARGUMENTS OF THE EXPRESS-
;; ION ARE EVALUATED
(+ (PRINT (+ 1 2))
   (PRINT (+ 3 4))) ; => 10

;; THE SPECIAL OPERATOR QUOTE
;; RETURNS ITS ARGUMENTS
;; "UNCHANGED"
(QUOTE (+ 1 2) ; => (+ 1 2)
```

USING FILES
ONCE YOU'VE MADE A FUNCTION ONCE, YOU CAN USE IT MANY TIMES, BUT AFTER YOU RESTART YOUR ENVIRONMENT, YOU'LL HAVE TO REMAKE THE FUNCTION.
IF IT'S DIFFICULT TO INPUT THE SAME FUNCTION SO MANY TIMES, YOU SHOULD USE A TEXT EDITOR TO WRITE THE FUNCTION INTO A FILE, THEN LOAD IT FROM YOUR ENVIRONMENT.

```
;;; /USR/HOME/ZICK/TEST.LISP
(DEFUN ADD (X Y)
    (+ X Y))

(DEFUN INC (X &OPTIONAL) (Y 1))
    (+ X Y))
```

ONCE YOU'VE CREATED A FILE AS MENTIONED ABOVE, WHEN YOU EVALUATE THE EXPRESSION (LOAD "/USR/HOME/ZICK/TEST.LISP) YOU'LL BE ABLE TO USE THE FUNCTIONS ADD AND INC. ON WINDOWS OS, REMEMBER THAT IT'S IMPORTANT TO INPUT "\\" FOR "/" INSTEAD.

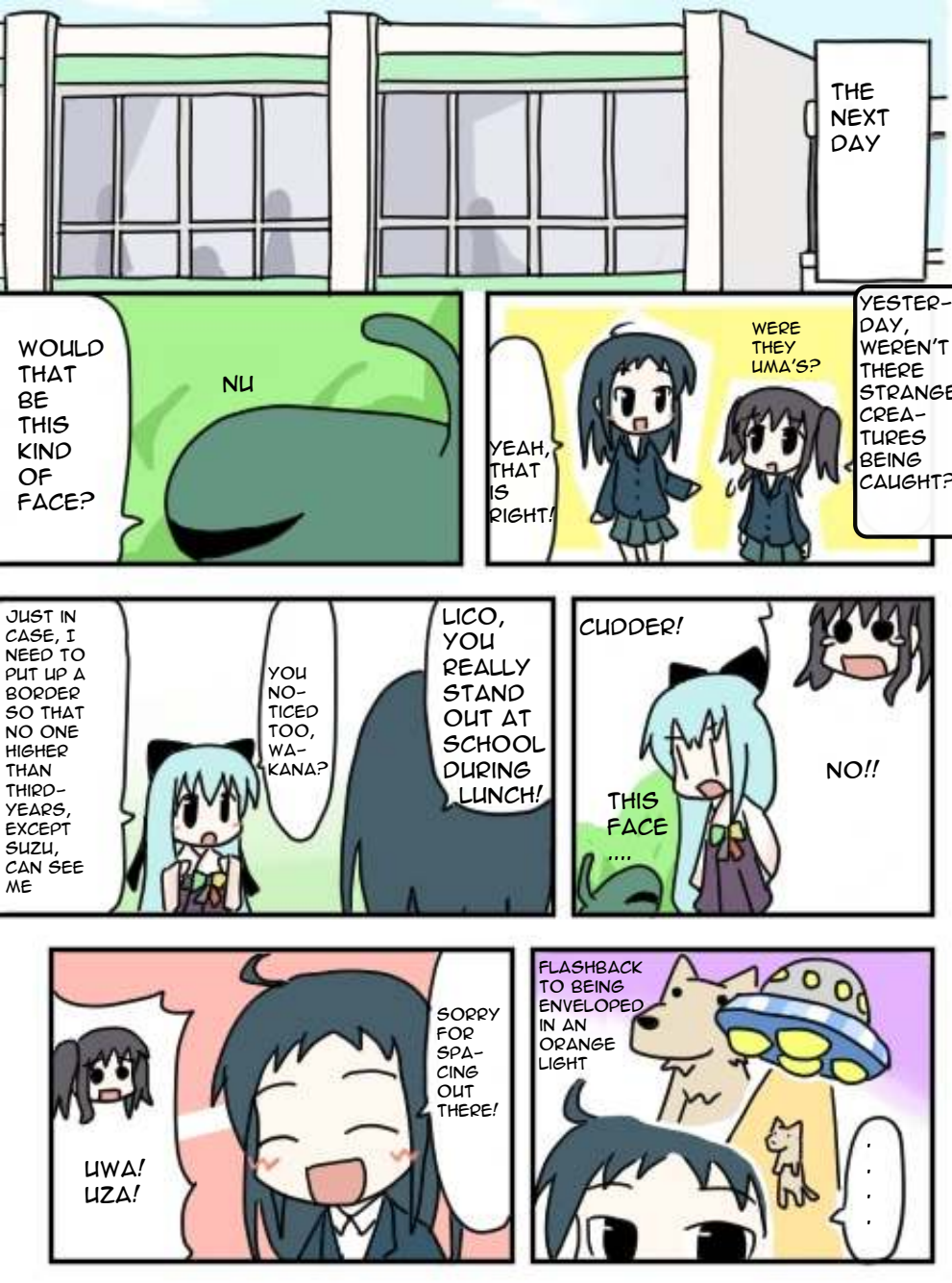

SAVE THE EARTH
THE WORLD IS IN TROUBLE AND YOUR POWER MAY BE IMPORTANT!
LICO WAS A GOD


SUZU MEETS LICO WHILE CHA-SING A CAT


REFERENCE BOOK
AND THEN... LISP
THERE WAS LISP.

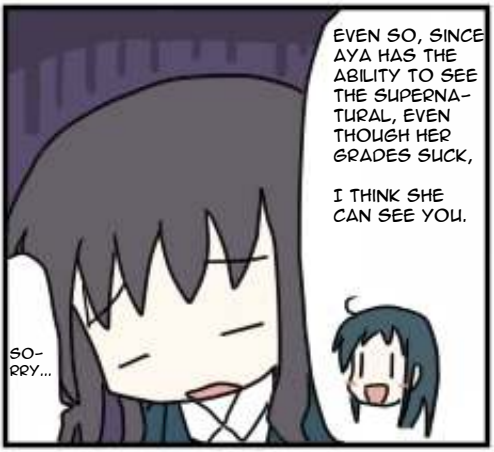

cdr
IN ORDER TO RECOVER THE ITEM THAT WILL SAVE THIS TOWN, LICO CAUGHT SOME CUDDERS


THAT WAS A SUMMARY OF WHAT'S HAPPENED UNTIL NOW IN THE MANGA GUIDE TO LISP.
WELL, SEE YOU TOMORROW~


THE NEXT DAY
WOULD THAT BE THIS KIND OF FACE?
NU
YESTER-DAY, WEREN'T THERE STRANGE CREA-TURES BEING CAUGHT?
WERE THEY UMA'S?
YEAH, THAT IS RIGHT!
JUST IN CASE, I NEED TO PUT UP A BORDER SO THAT NO ONE HIGHER THAN THIRD-YEARS, EXCEPT SUZU, CAN SEE ME
YOU NO-TICED TOO, WA-KANA?
LICO, YOU REALLY STAND OUT AT SCHOOL DURING LUNCH!
CUDDER!
NO!!
THIS FACE ....
SORRY FOR SPA-CING OUT THERE!
UWA! UZA!
FLASHBACK TO BEING ENVELOPED IN AN ORANGE LIGHT
. . . .


WHAT'S A TAN?
AYA-TAN?


WHICH IS CUTER?
SORRY FOR BEING SO CUTE

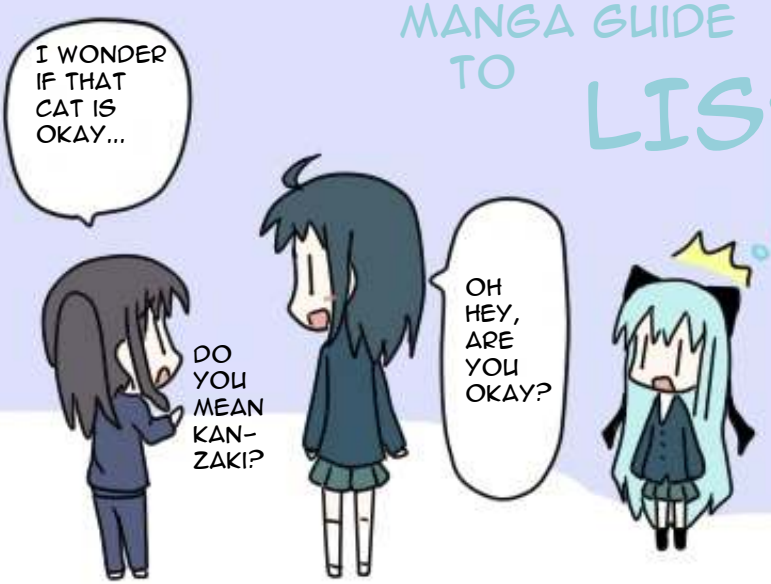

EVEN SO, SINCE AYA HAS THE ABILITY TO SEE THE SUPERNATURAL, EVEN THOUGH HER GRADES SUCK,
I THINK SHE CAN SEE YOU.
SORRY...


NOOO! ITS A MISUNDERSTANDING!
JUST STOP IT!
ELEMENTARY SCHOOLERS CAN SEE YOU, BUT IF A BAD SITUATION MAKES YOU DRINK MEDICINE..


CAN'T YOU USE YOUR HEAD AND DO SOMETHING?
YUP..
THAT'S UNEXPECTED.

THE MANGA GUIDE TO LISP

PART 4



T \

EXPRESSIONS AND VALUES
ONE OF LISP'S INTERESTING TRAITS IS THAT YOU CAN MAKE "PROGRAMS THAT WRITE PROGRAMS". THIS FEATURE IS CALLED A "MACRO", AND TO DEMONSTRATE ITS POWER, THIS SECTION WILL PRESENT AN EASY WAY TO USE EVAL. EVAL IS AN EXPRESSION-EVALUATING FUNCTION. FOR MORE DETAILS, PLEASE READ THE SEQUEL MANGA.

READ AND EVAL
BEFORE THEY ARE LOADED INTO YOUR INTERPRETER, EXPRESSIONS ARE JUST STRINGS OF CHARACTERS. WHEN THEY ARE LOADED INTO YOUR INTERPRETER, THEY ARE TURNED INTO UNIQUE INTERNAL REPRESENTATION. THEN, THIS IS EVALUATED. SOMETIMES PEOPLE MISUNDERSTAND AND THINK THAT IT EVALUATES STRINGS OF CHARACTERS, SO PLEASE BE CAREFUL. SHORTENING YOUR FUNCTIONS' NAMES WILL NOT CHANGE THEIR EXECUTION SPEED! BECAUSE OF THIS, YOU CAN THINK OF LISP AS EVALUATING LISTS. IN FACT, IN LISP THERE IS THE FUNCTION READ, WHICH ACCEPTS INPUT (STRINGS OF CHARACTERS) AND TURNS IT INTO A LIST, AND THE FUNCTION EVAL, WHICH ACCEPTS AND EVALUATES A LIST.

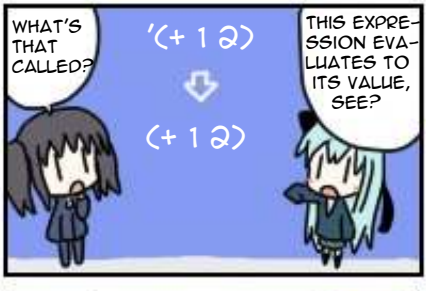
EVALUATING INPUT
(PRINT (EVAL (READ)))
WHEN THE ABOVE EXPRESSION IS EVALUATED, THE INTERPRETER WAITS FOR INPUT FROM THE KEYBOARD. IF YOU INPUT AN APPROPRIATE EXPRESSION, IT WILL DISPLAY ITS VALUE.

FOUNTAIN OF WISDOM

IF YOU PUT A QUOTATION MARK IN FRONT OF AN EXPRESSION, THE EXPRESSION BECOMES THAT VALUE. IN FACT, THE EXPRESSION 'EXPR IS AN ABBREVIATION FOR (QUOTE EXPR), AND QUOTE IS A SPECIAL OPERATOR WHICH RETURNS ITS ARGUMENT "WITHOUT EVALUATING IT". THE VALUE OF 'HELLOWORLD,HELLOWORLD IS CALLED A SYMBOL TYPE (VARIETY) VALUE, WHILE NUMBERS AND CONS COME OUT THE SAME. USING A QUOTATION MARK IS CALLED "QUOTING"


```
;; GET A SYMBOL
'LICO    ; => LICO
'SUZU    ; => SUZU

;; PUT SYMBOLS INTO A CONS
(CONS 'LICO 'SUZU)
; => (LICO . SUZU)

;; A NUMBER HAS THE SAME
;; VALUE WHEN QUOTED
'1  ; => 1
1   ; => 1
```

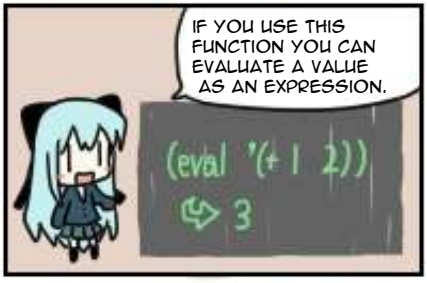
ENTER EGGHEAD



YOU CAN ALSO QUOTE EXPRESSIONS THAT ARE WRAPPED IN PARENTHESES. THESE VALUES ARE LIST TYPE. LISTS ARE EITHER "THE EMPTY LIST, NIL" OR "A CONS WITH A LIST IN THE CDR". SINCE RECURSIVE DEFINITIONS CAN BE PUZZLING, LOOKING AT AN EXAMPLE MIGHT BE EASIER TO UNDERSTAND. NIL IS A SPECIAL LITERAL THAT REPRESENTS THE EMPTY LIST, AND ITS VALUE IS ALSO NIL

```
;; A LIST WITH THE
;; EMPTY LIST IN ITS
;; CDR. THE ". NIL" IN
;; (2 . NIL) IS OMITTED
(CONS 2 NIL)   ; =>(2)

;; A LIST WITH A LIST IN ITS CDR
;; HERE ". (" IS OMITTED
(CONS 1 (CONS 2 NIL)   ; => (1 2)
(CONS '+ (CONS 1 (CONS 2 NIL)))
; => (+ 1 2)

;; THE VALUE OBTAINED IS THE
;; SAME AS THE VALUE OF THE
;; ABOVE EXPRESSION
'(+ 1 2) ; => (+ 1 2)
```

AT FIELD

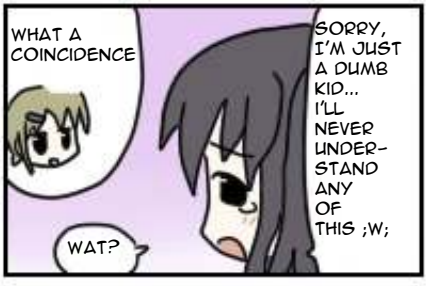


THE FUNCTION EVAL ACCEPTS A VALUE REPRESENTING AN EXPRESSION, THEN RETURNS THE RESULT OF EVALUATING IT. IN OTHER WORDS, EVAL'S ARGUMENT IS AN EXPRESSION WHICH EVALUATES TO A VALUE REPRESENTING AN EXPRESSION. IF EVAL'S RETURN VALUE IS A VALUES REPRESENTING AN EXPRESSION, YOU CAN USE IT AS ANOTHER ARGUMENT TO EVAL. THIS IS VERY FUN, RIGHT? INCIDENTALLY, THE PEOPLE AROUND ME NORMALLY SAY "EVAL"

TN: REFERENCE TO A JAPANESE VERB "EBARU" AS A SLANG SHORTENING OF "EVALUATION"

```
(EVAL '(+ 1 2)) ; => 3

(EVAL
  '(CONS '+
     (CONS 1 (CONS 2 NIL))))
; => (+ 1 2)

(EVAL
  (EVAL
    '(CONS '+
       (CONS 1 (CONS 2 NIL)))))
; => 3
```

COINCIDENCE SENPAI



THERE SEEM TO BE A LOT OF PEOPLE CONFUSED BY DISCUSSIONS ON QUOTE AND EVAL. HOWEVER, DON'T WORRY IF YOU DON'T UNDERSTAND IMMEDIATELY. IF YOU START GETTING TIRED, YOU SHOULD DRAW A PICTURE OF THE CONS. IF YOU GET GOOD AT DRAWING PICTURE OF EXPRESSIONS' CONSES, YOU WILL SURELY START UNDERSTANDING. FOR HOW TO DRAW A PICTURE, PLEASE SEE SECTION 1.2

```
(CONS 'B NIL) ; => (B)
(CONS 'A (CONS 'B NIL))
; => (A B)

(CDR (CONS 'A (CONS 'B NIL)))
; => (B)

(CDR '(A B))  ; => (B)

(CAR '(QUOTE A)) ; => QUOTE
(CAR ''A) ; => QUOTE
```


IF I WIN, YOU'LL GIVE ME THAT THING AND THE CUDDER
WE'LL SETTLE THIS MEAN-INGLESS DISPUTE


A FEW DAYS AGO


FINE.
AND IF I WIN, YOU'LL LEAVE THIS PLACE!


THE FLAVOR DOESN'T MATTER..
OH, IS THAT A WHITE CUDDER? I THOUGHT THAT WAS IM-POSSIBLE


GO FULL THROTTLE, MIRIN!
NYAL-RIGHT!


GO!
YELLOW CUDDER:
* LEMON FLAVOR
* LIGHT X2
WHITE CUDDER:
* RARE
* HIGH POWER
* LANTERN
WHOEVER IS IN THE LEAD BY SUNRISE WINS!


DO-OVER!
I REFUSE.
THAT WAS NOT A FAIR RACE!
YOUR CUDDER CAN'T BE YOUR PARTNER UNLESS YOU HAVE THE THING
IT SHOULD BE JUDGED BY THE RULES, FURTHER-MORE-
MEOW'LL NEVER BE CAUGHT BY THIS SLOW-POKE~
MMMM.....
YOUR OWN BEHAVIOR-!
WHAT A PITI-FUL LOSER, MEOW~
IF YOU CATCH ME,
I YOU CAN HAVE A REMATCH, MEOW~


AND SO, PRESENT DAY
MIRIN~
SMALL-TYPE UMA CAPTURE CASE (U RESEARCH EQUIPMENT)
IS WHAT SHE SAID.
THIS SHOULD BE HELPFUL FOR CATCHING A CAT, RIGHT?
WELL..
AS EXPECTED OF YOU, SUZU!
WHY DO YOU HAVE THOSE CLOTHES, LICO?
AND WHAT ABOUT YOUR HEAD, SUZU?
ひゃん
?
NOW THAT I'VE CAUGHT YOU, WHAT NEXT?
WELL,

THE MANGA GUIDE TO LISP

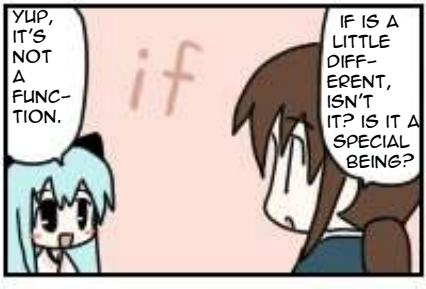
05

HELLO WORLD
THE VALUE OF THE EXPRESSION 'HELLOWORLD IS HELLOWORLD, SO YOU CAN USE IT TO DISPAY THE STRING "HELLOWORLD". THIS IS A LITTLE MUCH, THOUGH. IF YOU DO (PRINT 'HELLOWORLD), IT'S CLEARER THAT YOU ARE PRINTING SOMETHING, BUT WE WANT TO PUT A SPACE BETWEEN HELLO AND WORLD. TO DO THAT WE WILL USE AN EXPRESSION LIKE THIS:
(PRINT "HELLO WORLD")

SOMETIMES WRAPPED IN DOUBLE QUOTES (") BECOMES A "STRING". LIKE NUMBERS, THEY RETURN THEMSELVES WHEN EVALUATED, SO YOU DON'T NEED TO QUOTE THEM. DISPLAYING A STRING WITH PRINT WILL ALSO DISPLAY THE QUOTES. THIS IS UGLY, SO LET'S JUST PRINT THE CONTENTS OF A STRING:
(FORMAT T "HELLO WORLD~%")
THIS WILL DISPLAY HELLO WORLD.

"~%" ISN'T DISPLAYED, AND INSTEAD A NEWLINE IS. THIS IS A FUNCTION OF FORMAT, AND PRINT "~%") WILL DISPLAY "~%" UNCHANGED. TO PRINT A "~%" WITH FORMAT, YOU CAN DO (FORMAT T "~~%")

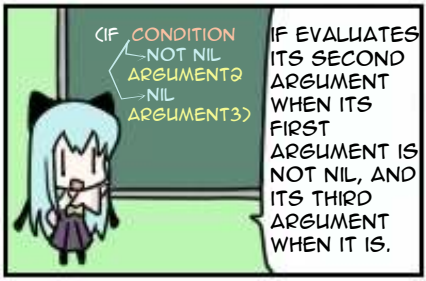
THE LONG AWAITED DIFFICULT 4KOMA

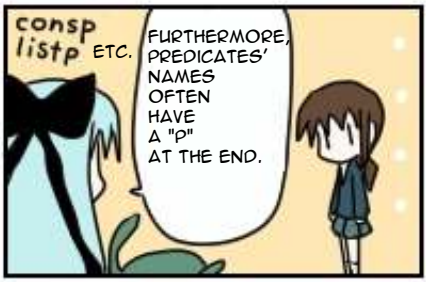


IF IS A SPECIAL OPERATOR, WHICH ALWAYS EVALUATES ITS FIRST ARGUMENT (ARGUMENT 1), BUT EVALUATES ONLY ONE OF ARGUMENTS 2 AND 3, AND THAT VALUE BECOMES THE VALUE OF THE WHOLE IF EXPRESSION. INCIDENTALLY, AN EXPRESSION WHICH USES A SPECIAL OPERATOR IS CALLED A SPECIAL FORM. = IS A FUNCTION THAT CHECKS WHETHER TWO (OR MORE PRECISELY, AT LEAST ONE) NUMBERS ARE EQUAL. FOR MORE DETAILS, PLEASE READ THE NEXT MANGA.

```
;; A FUNCTION THAT PRINTS "X=0"
;; IF ITS ARGUMENT IS
;; NOT 0:
(DEFUN TEST1 (X)
  (IF (= X 0)
    (PRINT 'X=0)
    (PRINT 'X/=0)))

;; THE VALUE CHANGES DEPENDING
;; ON THE ARGUMENT
(TEST 0) ; => X=0
(TEST 1)  ; => X/=0
(TEST -1) ; => X/=0
```

HARUKA





WHEN IF'S FIRST ARGUMENT IS SOMETHING OTHER THAN NIL, IT EVALUATES ITS SECOND ARGUMENT, AND WHEN THAT IS NIL TOO, IT EVALUATES ITS THIRD ARGUMENT. FUNCTIONS THAT RETURN T OR NIL ARE CALLED "PREDICATES" AND ARE OFTEN USED AS THE FIRST ARGUMENT TO IF. FOR EXAMPLE, = IS A PREDICATE. THERE ARE OTHER PREDICATES, LIKE CONSP, WHICH RETURNS T WHEN ITS ARGUMENT IS A CONS. MANY PREDICATES HAVE A P ATTACHED TO THE END OF THEIR NAMES. P IS AN ABBREVIATION OF PREDICATE.

```
;; USING A
;; PREDICATE:
(= 0 0) ; => T
(= 0 1 ) ; => NIL
(< 0 0) ; => NIL
(< 0 1 ) ; => T

;; A FUNCTION THAT
;; DISTINGUISHES ZERO, POSITIVE,
;; AND NEGATIVE
(DEFUN TEST2 (X)
  (IF (= X 0)
    (PRINT 'X=0)
    (IF (< X 0)
      (PRINT 'X<0)
      (PRINT 'X>0))))
```

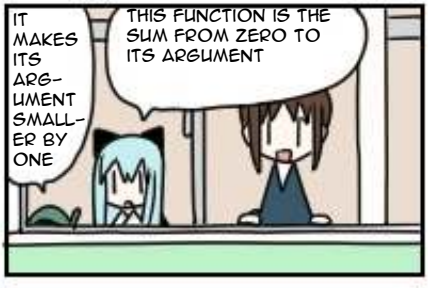
AND NOW, COND

COND TAKES AN ARBITARY NUMBER OF (CONDITIONAL EXPRESSION) FORM ARGUMENTS. COND FIRST EVALUATES THE FIRST CONDITIONAL, THEN IF IT IS NOT NUL, EVALUATES THE CORRE-SPONDING EXPRESSION, AND IF THAT IS NIL, EVALUATES THE NEXT CONDITIONAL. AN IF WHICH ALSO HAS AN IF IN ONE OF ITS ARGUMENTS CAN BE WRITTEN AS A SIMPLER COND. PLEASE COMPARE THE PREVIOUS MANGA'S TEST2 AND TEST3 BELOW.

```
;; A FUNCTION THAT
;; IS THE SAME AS
;; TEST2
(DEFUN TEST3 (X)
  (COND
    ((= X 0)
      (PRINT 'X=0))
    ((< X 0)
      (PRINT 'X<0))
    (T (PRINT 'X>0))))
```

```
;; DETERMINES WHETHER ITS
;; ARGUMENT IS ZERO,
;; POSITIVE, OR NEGATIVE
(TEST3 0) ; => X=0
(TEST3 1) ; => X>0
(TEST3 -1) ; => X<0
```

RECURSION



YOU CAN IMPLEMENT ITERATION WITH RECURSIVE FUNCTION CALLS. THE FUNCTION F ON THE LEFT, IF X IS BIGGER THAN 0, CALLS F WITH MINUS 1. IT MAKES THE ARGUMENT SMALLER ONE BY ONE, SO THAT IT FINISHES WHEN IT REACHES 0. HEY, IF YOU PASS A NEGATIVE NUMBER TO F, IT WILL BECOME AN INFINITE LOOP. IN THIS CASE, EXIT THE INTERPRETER WITH CTRL-C.

```
;; A FUNCTION THAT FINDS
;; THE FACTORIAL
;; (1- N) IS THE SAME
;; AS (- N 1)
(DEFUN FACT (N)
  (COND ((= N 0) 1)
    (T (* FACT (1- N)
       N))))

;; SINCE N IS 0, RETURN 1
(FACT 0) ; => 1
;; MULTIPLY 1 AND A RECURSIVE
;; (FACT 0)
(FACT 1) ; => 1
;; MULTIPLY 2 AND A RECURSIVE
;; (FACT 1)
(FACT 2) ; => 2
;; MULTIPLY 3 AND A RECURSIVE
;; (FACT 2)
(FACT 3) ; => 6
```