An Introduction to LISP

Using the compilers conforming to Common Lisp of:

Steel Bank Common Lisp ~ SBCL

&

CLISP

Waikato Linux Users Group

Ian Stewart & Jake Waas

23 November 2020

Contents

History

- LISP derives from **LIS**t **P**rocessor
- Originally created as a practical mathematical notation by John McCarthy in 1958.
- Steve Russell wrote the Lisp interpreter.
- 1962 Lisp compiler.
- Second oldest commonly used language after Fortran.
- All program code is written as s-expressions, or parenthesized lists.
- Data may be a list. E.g.: '("A" "B" "C") or (list 1 2 3 4).
- Function may be a list: (+ 2 3) Add 2 and 3 to return a result of 5.
- Many variations of Lisp. Common Lisp in 1984 a consolidaton.
- Compilers: CLISP and Steel Bank Common Lisp (SBCL)
- Ubuntu $ apt search lisp finds CLISP and SBCL
- CLISP V2.49 – July 2010
- SBCL V2.0.10 – October 2020

Algebraic Notations

- Infix notation: 5 * 2 + 3
- Postfix notation (Reverse Polish Notation): 5 2 * 3 +
  - No need not to have Operator Precedence.
- Prefix notation: + 3 * 5 2
- **Lisp** notation (+ (* 5 2) 3)
  - sexp (short for S-Expression, where S stands for Symbolic).
  - It is sometimes known as Fully Parenthesized Notation.
- functional notation +(*(5 2) 3)
- matchfix notation: (* (+ 5 2 +) 3 *)

Reference: http://xahlee.info/UnixResource_dir/writ/notations.html

Notation. Comments and Blank lines

- **Lisp** comments notation examples:

```
;;;; Main heading for program is
;;;; four semi-colons.

;;; Section 1. Three semi-colons.

   ;; Sub-secton 1a. The next part of the program.

(print (lisp-implementation-type)) ; Comment. prints SBCL.

#|
This comment, using hash and vertical bar
is spread over multiple lines.
|#
```

Notation. Parenthesis and Whitespace

- **Lisp** notation examples (function arg arg):

```
(+ 3 4)  ; Result is 7
(* 2 (+ 3 4))  ; Result is 14
```

- Left and Right parenthesis pairs must match.
- Function first followed by arguments.
- A space character delimits functions and arguments
- May have multiple whitespace characters. Spaces, tabs newlines.
- Four examples of the same multiplication function that return 6:

  - `( print ( * 2 3 ) )`
  - `(print( * 2 3 ))`
  - `(print(* 2 3))`

  -
    ```
    (print
        (
        * 2 3
        )
    )
    ```

Documentation Links

- http://www.sbcl.org/manual/index.html
- http://www.gigamonkeys.com/book/
- http://www.lispworks.com/documentation/common-lisp.html
- https://www.gnu.org/software/emacs/manual/html_node/elisp/
- http://www.ai.mit.edu/projects/iiip/doc/CommonLISP/HyperSpec/FrontMatter/Chapter-Index.html
- http://www.lispworks.com/documentation/HyperSpec/Front/Contents.htm
- https://www.tutorialspoint.com/lisp/
- https://riptutorial.com/Download/common-lisp.pdf

Documentation Internal

- * (documentation 'princ 'function)

  "Output an aesthetic but not necessarily READable printed representation

    of OBJECT on the specified STREAM."
- * (describe 'princ)

Installation

CLISP. Website: https://clisp.sourceforge.io/

- $ sudo apt install clisp clisp-doc

SBCL. Website: http://www.sbcl.org/

- $ sudo apt install sbcl sbcl-doc
- $ sudo apt install rlwrap
- Latest sbcl may be downloaded from web-site as AMD64 binary.
  http://www.sbcl.org/platform-table.html

rlwrap for SBCL – Thanks to Lawrence D'Oliveiro

CLISP Launch

```
ian@ian:~$ clisp
  i i i i i i i       ooooo    o        ooooooo   ooooo   ooooo
  I I I I I I I      8     8   8           8     8     o  8    8
  I  \ `+' /  I      8         8           8     8        8    8
   \  `-+-'  /       8         8           8      ooooo   8oooo
    `-__|__-'        8         8           8           8  8
        |            8     o   8           8     o     8  8
  ------+------       ooooo    8oooooo  ooo8ooo   ooooo   8

Welcome to GNU CLISP 2.49.92 (2018-02-18) <http://clisp.org/>

Copyright (c) Bruno Haible, Michael Stoll 1992-1993
Copyright (c) Bruno Haible, Marcus Daniels 1994-1997
Copyright (c) Bruno Haible, Pierpaolo Bernardi, Sam Steingold 1998
Copyright (c) Bruno Haible, Sam Steingold
Copyright (c) Sam Steingold, Bruno Haible

Type :h and hit Enter for context help.

[1]>  (quit)
Bye.
ian@ian:~$
```

```
[1]> :h
You are in the top-level Read-Eval-Print loop.
Help (abbreviated :h) = this list
Use the usual editing capabilities.
(quit) or (exit) leaves CLISP.
```

(quit) or (exit) or Ctrl D to get out and back to bash prompt

CLISP includes GNU_Readline

```
Type :h and hit Enter for context help.

[1]> (princ "hello world")
"(princ \"hello world\")" ;
hello world
"hello world"
[2]> (princ "hello world")
hello world
"hello world"
[3]> (princ "hello world")
hello world
"hello world"
[4]> (princ "up arrow gets history OK")
up arrow gets history OK
"up arrow gets history OK"
[5]> (+ 2 3)
5
[6]> (+ 2 3)
5
```

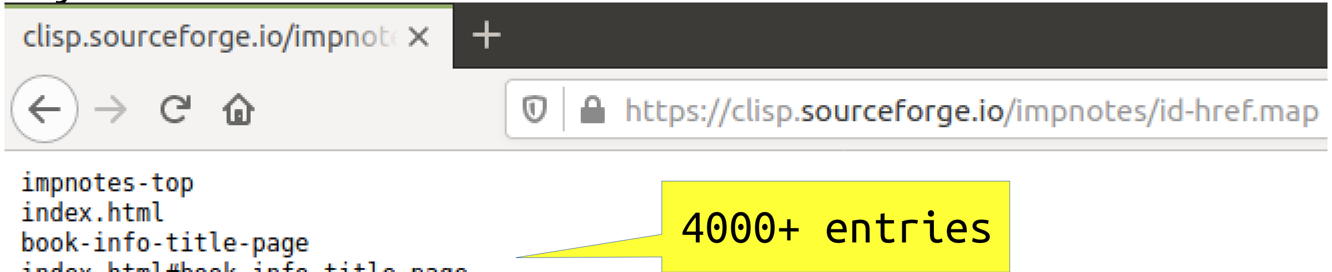
Up-Arrow recalls last line. Plus Ctrl A start, Ctrl E end, etc.

Reference: https://en.wikipedia.org/wiki/GNU_Readline

CLISP (print (describe 'print))

```
PRINT is the symbol PRINT, lies in #<PACKAGE COMMON-LISP>, is accessible in 11
packages CLOS, COMMON-LISP, COMMON-LISP-USER, EXPORTING, EXT, FFI, POSIX,
READLINE, REGEXP, SCREEN, SYSTEM, names a
;; connecting to "http://www.ai.mit.edu/projects/iiip/doc/CommonLISP/HyperSpec/Data/
Symbol-Table.text"...connected...HTTP/1.1 200 OK...45,322 bytes
;; SYSTEM::GET-CLHS-MAP(#<IO INPUT-BUFFERED SOCKET-STREAM CHARACTER
www.ai.mit.edu:80>)...978/978 symbols
 function.
ANSI-CL Documentation is at

"http://www.ai.mit.edu/projects/iiip/doc/CommonLISP/HyperSpec/Body/
fun_writecm_p_rintcm_princ.html"
;; connecting to "http://clisp.org/impnotes/id-href.map"...connected...HTTP/1.1 301
Moved Permanently --> "https://clisp.sourceforge.io/impnotes/id-href.map"
*** - OPEN-HTTP("https://clisp.sourceforge.io/impnotes/id-href.map"): HTTPS
      protocol is not supported yet
ian@ian:~/lisp$
```

No https only http ~ huh?

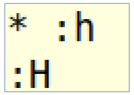

clisp.sourceforge.io/impnot
https://clisp.sourceforge.io/impnotes/id-href.map

```
impnotes-top
index.html
book-info-title-page
index.html#book-info-title-page
```

4000+ entries

SBCL Launch and Readline

```
ian@ian:~$ sbcl
This is SBCL 2.0.1.debian, an implementation of ANSI Common Lisp.
More information about SBCL is available at <http://www.sbcl.org/>.

SBCL is free software,... for more information.

* ^[[A^A^E

* (quit)

ian@ian:~$ rlwrap sbcl
This is SBCL 2.0.1.debian, an implementation of ANSI Common Lisp.
More information about SBCL is available at <http://www.sbcl.org/>.

SBCL is free software,... for more information.
* (+ 2 3)
5
* (+ 2 3)
```



```
* :h
:H
```

No "help" like CLISP

No readline. Trying to do up-arrow and Ctrl A/E start / end of line

Have Readline. E.g. Up-arrow displays previous command line, etc.

SBCL (print (describe 'print))

```
COMMON-LISP:PRINT
  [symbol]

PRINT names a compiled function:
  Lambda-list: (SB-IMPL::OBJECT &OPTIONAL STREAM)
  Declared type: (FUNCTION (T &OPTIONAL (OR STREAM BOOLEAN))
                  (VALUES T &OPTIONAL))
  Derived type: (FUNCTION (T &OPTIONAL T) (VALUES T &OPTIONAL))
  Documentation:
    Output a newline, the mostly READable printed representation of OBJECT, and
      space to the specified STREAM.
  Known attributes: unwind, any
  Source file: SYS:SRC;CODE;PRINT.LISP
```

SBCL (describe) OK – No Error

SBCL Command line Interpreter

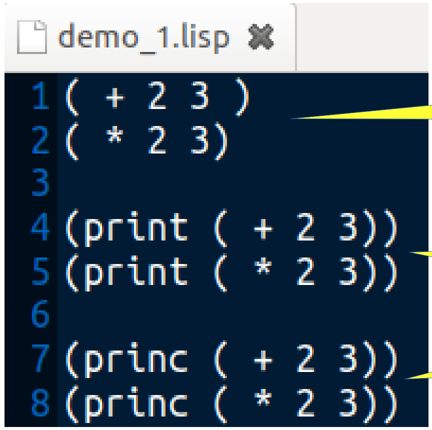
```
* ( + 2 3 )
5
* (* 2 3)
6


* (print (* 2 3))

6
6
* (print (+ 2 3))

5
5

* (princ (+ 2 3))
5
5
* (princ (* 2 3))
6
6
```

1. Interactive. Result displayed

2. Print inserts newline.

3. Princ no newline

SBCL Save file as demo_1.lisp



```lisp
( + 2 3 )
( * 2 3)

(print ( + 2 3))
(print ( * 2 3))

(princ ( + 2 3))
(princ ( * 2 3))
```

1

2

3

```
ian@ian:~/lisp/sbcl$ sbcl --load demo_1.lisp
This is SBCL 2.0.1.debian, an implementation of ANSI Common Lisp.
More information about SBCL is available at <http://www.sbcl.org/>.

SBCL is free software, provided as is, with absolutely no warranty.
It is mostly in the public domain; some portions are provided under
BSD-style licenses.  See the CREDITS and COPYING files in the
distribution for more information.

5
6 56
*
```

2 Print preceeds result with newline

3 Princ no newline

File: demo_print.lisp

Output to Console.

```
#!/usr/bin/sbcl --script
;;;; Output to Console
;;;; demo print.lisp
;; Print. Inserts a newline prior to outputting quoted string
(princ "|")
(print "Hello world using print")
(princ "|")

;; Princ. Output, without quotes, just the string
(terpri) ; Output a newline
(princ "|")
(princ "Hello world using princ")
(princ "|")

;; write Outputs quoted string
(terpri)
(princ "|")
(write "Hello world using write")
(princ "|")

```

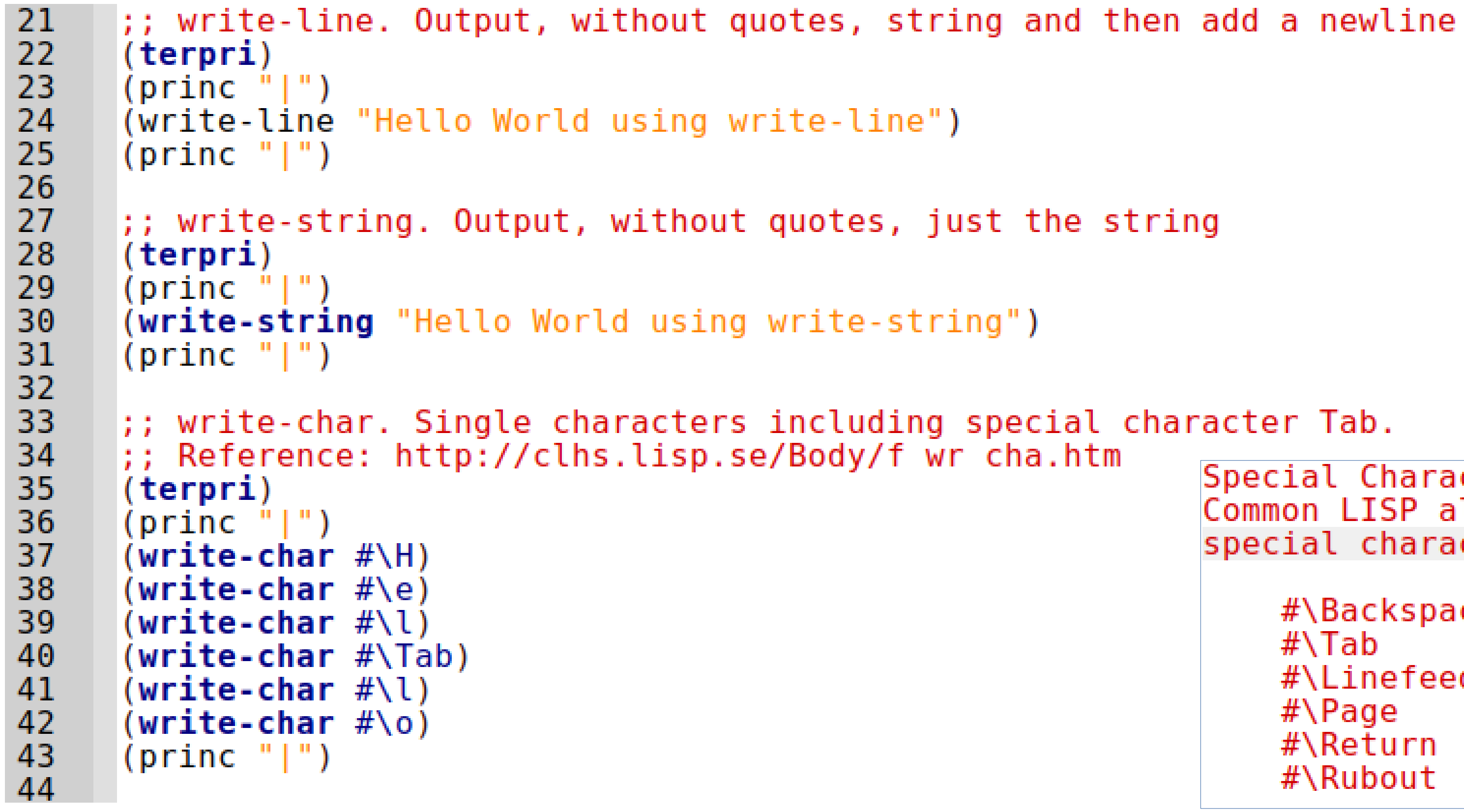
Must be double quotes. " not '

princ does not provide new-line. Use (terpri) to insert newlines

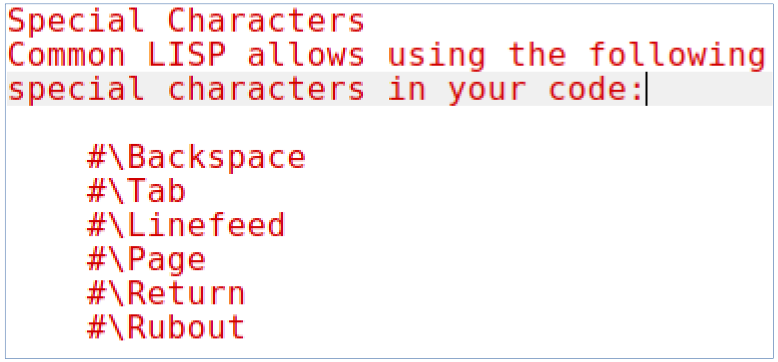
```
ian@ian:~/lisp/sbcl$ sbcl --script demo_print.lisp
|
"Hello world using print" |
|Hello world using princ|
|"Hello world using write"|
```

File: demo_print.lisp

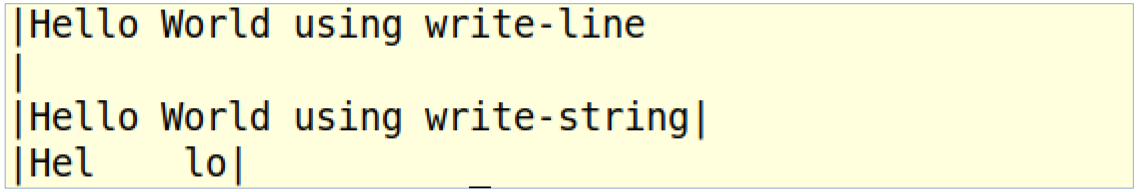
Output to Console continued...



```lisp
;; write-line. Output, without quotes, string and then add a newline
(terpri)
(princ "|")
(write-line "Hello World using write-line")
(princ "|")

;; write-string. Output, without quotes, just the string
(terpri)
(princ "|")
(write-string "Hello World using write-string")
(princ "|")

;; write-char. Single characters including special character Tab.
;; Reference: http://clhs.lisp.se/Body/f wr cha.htm
(terpri)
(princ "|")
(write-char #\H)
(write-char #\e)
(write-char #\l)
(write-char #\Tab)
(write-char #\l)
(write-char #\o)
(princ "|")
```



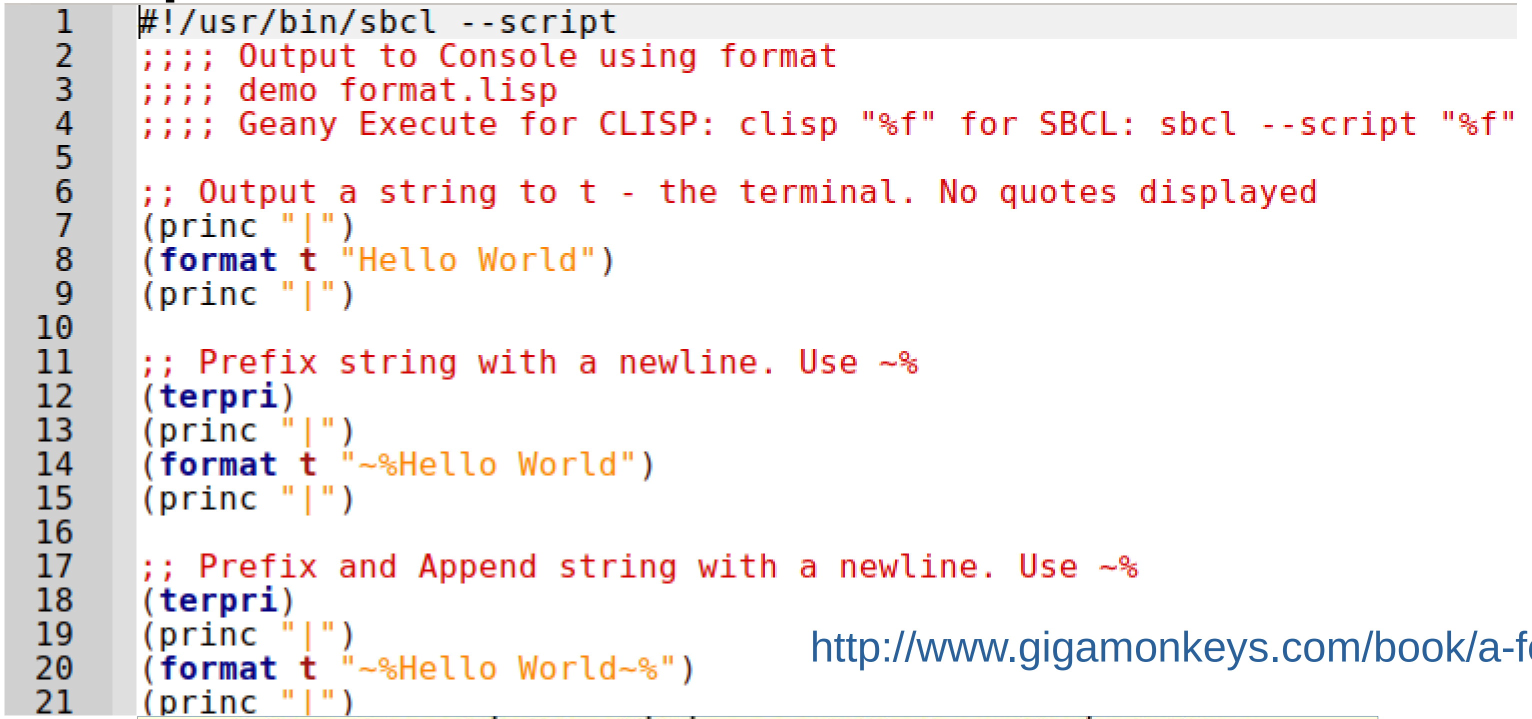
```
Special Characters
Common LISP allows using the following
special characters in your code:

    #\Backspace
    #\Tab
    #\Linefeed
    #\Page
    #\Return
    #\Rubout
```



```
|Hello World using write-line
|
|Hello World using write-string|
|Hel     lo|
```

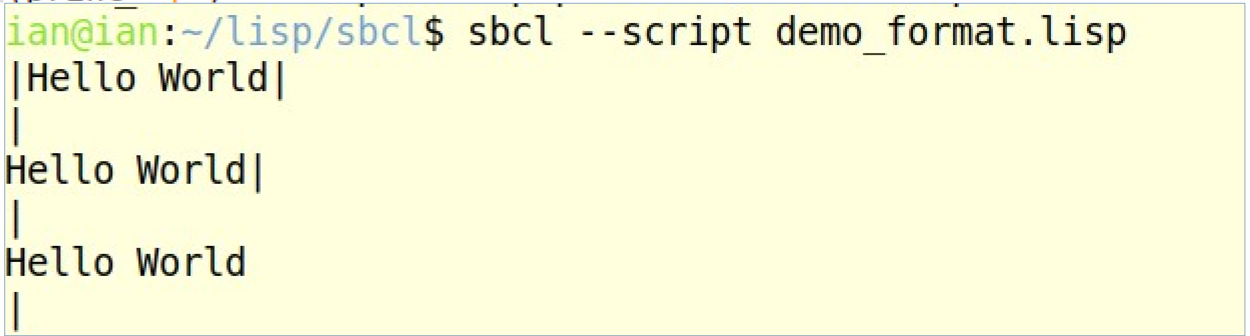
File: demo_print.lisp

Output to Console. Format...



```
#!/usr/bin/sbcl --script
;;;; Output to Console using format
;;;; demo format.lisp
;;;; Geany Execute for CLISP: clisp "%f" for SBCL: sbcl --script "%f"

;; Output a string to t - the terminal. No quotes displayed
(princ "|")
(format t "Hello World")
(princ "|")

;; Prefix string with a newline. Use ~%
(terpri)
(princ "|")
(format t "~%Hello World")
(princ "|")

;; Prefix and Append string with a newline. Use ~%
(terpri)
(princ "|")
(format t "~%Hello World~%")
(princ "|")
```

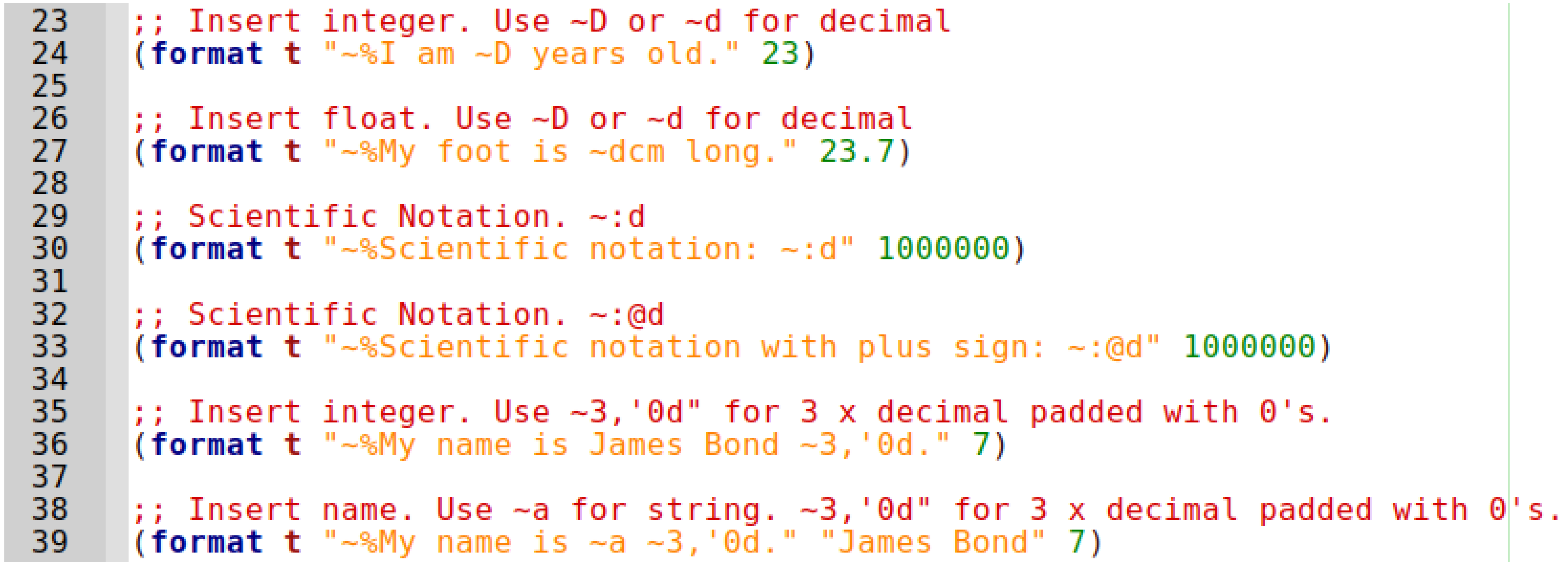
http://www.gigamonkeys.com/book/a-few-format-recipes.html



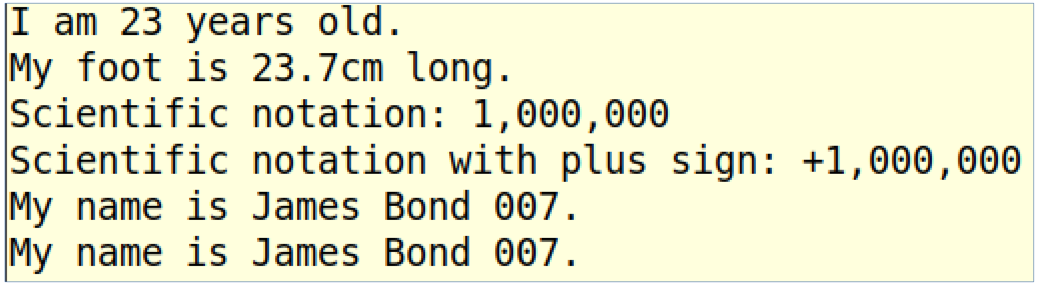
```
ian@ian:~/lisp/sbcl$ sbcl --script demo_format.lisp
|Hello World|
|
Hello World|
|
Hello World
|
```

File: demo_format.lisp

Output to Console. Format...



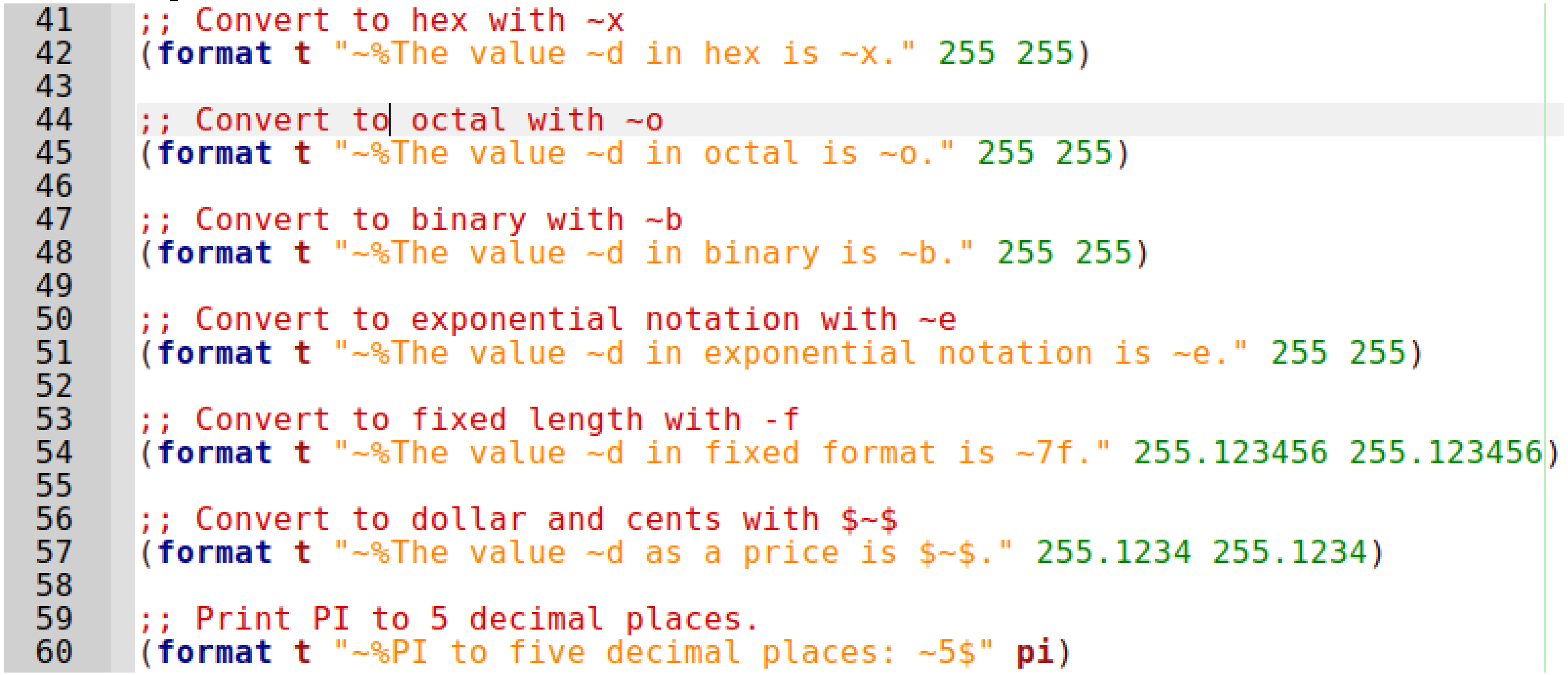
```lisp
;; Insert integer. Use ~D or ~d for decimal
(format t "~%I am ~D years old." 23)

;; Insert float. Use ~D or ~d for decimal
(format t "~%My foot is ~dcm long." 23.7)

;; Scientific Notation. ~:d
(format t "~%Scientific notation: ~:d" 1000000)

;; Scientific Notation. ~:@d
(format t "~%Scientific notation with plus sign: ~:@d" 1000000)

;; Insert integer. Use ~3,'0d" for 3 x decimal padded with 0's.
(format t "~%My name is James Bond ~3,'0d." 7)

;; Insert name. Use ~a for string. ~3,'0d" for 3 x decimal padded with 0's.
(format t "~%My name is ~a ~3,'0d." "James Bond" 7)
```



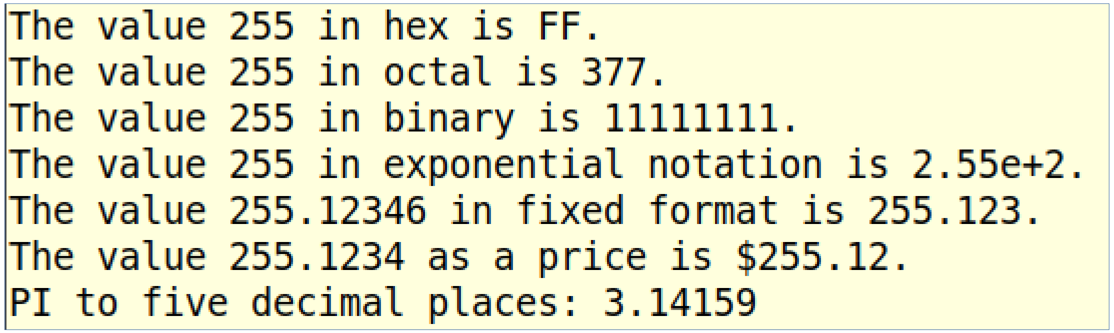
```
I am 23 years old.
My foot is 23.7cm long.
Scientific notation: 1,000,000
Scientific notation with plus sign: +1,000,000
My name is James Bond 007.
My name is James Bond 007.
```

File: demo_format.lisp

Output to Console. Format...



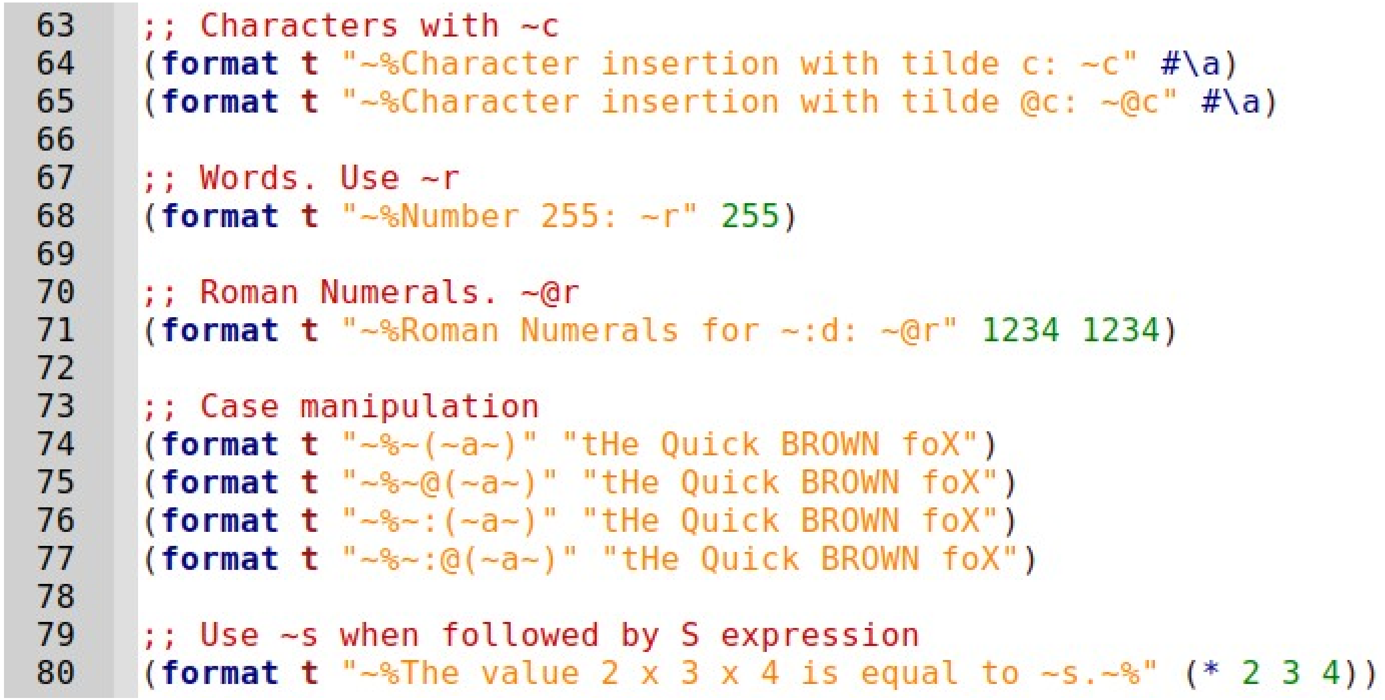
```lisp
;; Convert to hex with ~x
(format t "~%The value ~d in hex is ~x." 255 255)

;; Convert to octal with ~o
(format t "~%The value ~d in octal is ~o." 255 255)

;; Convert to binary with ~b
(format t "~%The value ~d in binary is ~b." 255 255)

;; Convert to exponential notation with ~e
(format t "~%The value ~d in exponential notation is ~e." 255 255)

;; Convert to fixed length with -f
(format t "~%The value ~d in fixed format is ~7f." 255.123456 255.123456)

;; Convert to dollar and cents with $~$
(format t "~%The value ~d as a price is $~$." 255.1234 255.1234)

;; Print PI to 5 decimal places.
(format t "~%PI to five decimal places: ~5$" pi)
```



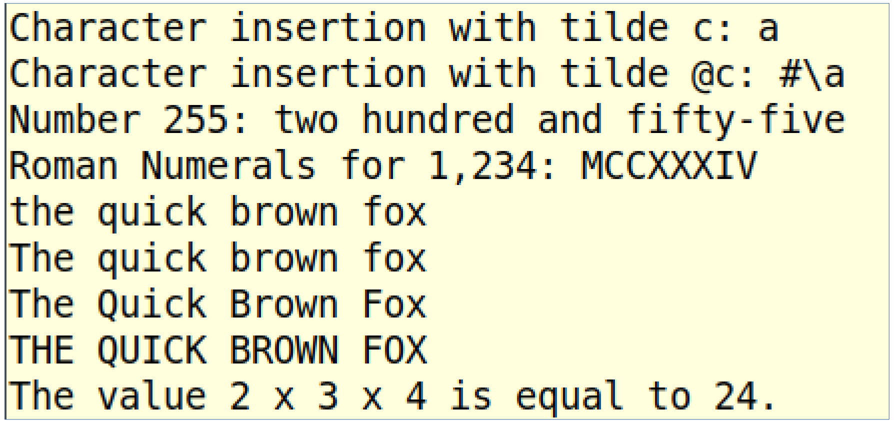
```
The value 255 in hex is FF.
The value 255 in octal is 377.
The value 255 in binary is 11111111.
The value 255 in exponential notation is 2.55e+2.
The value 255.12346 in fixed format is 255.123.
The value 255.1234 as a price is $255.12.
PI to five decimal places: 3.14159
```

File: demo_format.lisp

Output to Console. Format...



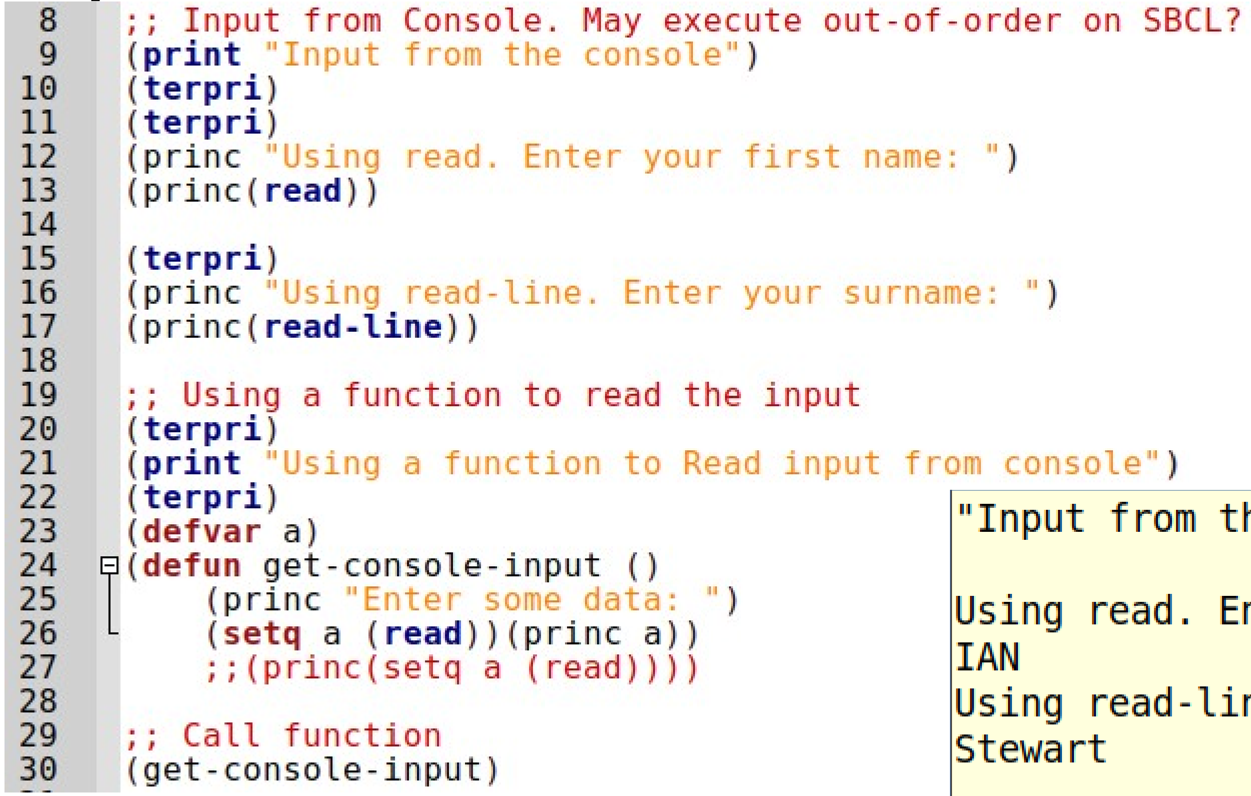
```lisp
;; Characters with ~c
(format t "~%Character insertion with tilde c: ~c" #\a)
(format t "~%Character insertion with tilde @c: ~@c" #\a)

;; Words. Use ~r
(format t "~%Number 255: ~r" 255)

;; Roman Numerals. ~@r
(format t "~%Roman Numerals for ~:d: ~@r" 1234 1234)

;; Case manipulation
(format t "~%~(~a~)" "tHe Quick BROWN foX")
(format t "~%~@(~a~)" "tHe Quick BROWN foX")
(format t "~%~:(~a~)" "tHe Quick BROWN foX")
(format t "~%~:@(~a~)" "tHe Quick BROWN foX")

;; Use ~s when followed by S expression
(format t "~%The value 2 x 3 x 4 is equal to ~s.~%" (* 2 3 4))
```



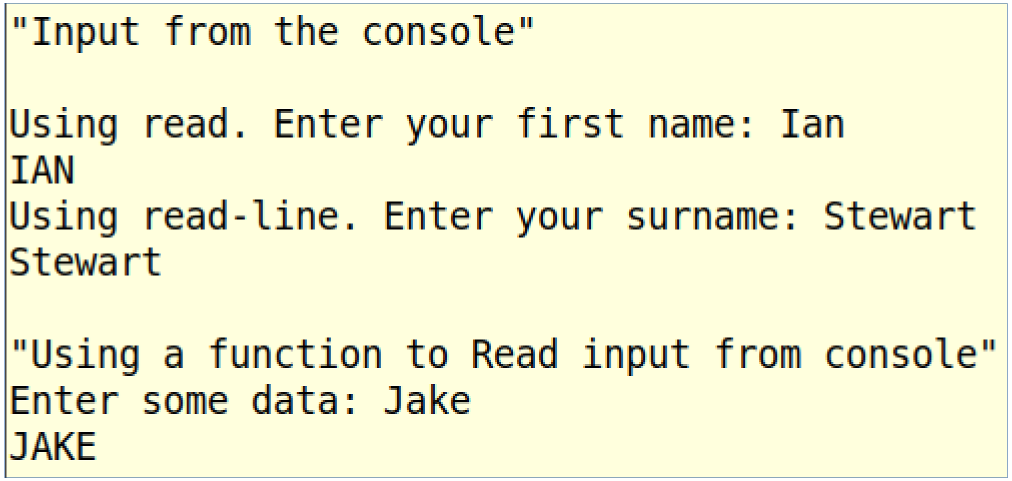
```
Character insertion with tilde c: a
Character insertion with tilde @c: #\a
Number 255: two hundred and fifty-five
Roman Numerals for 1,234: MCCXXXIV
the quick brown fox
The quick brown fox
The Quick Brown Fox
THE QUICK BROWN FOX
The value 2 x 3 x 4 is equal to 24.
```

File: demo_format.lisp

Input to Console.



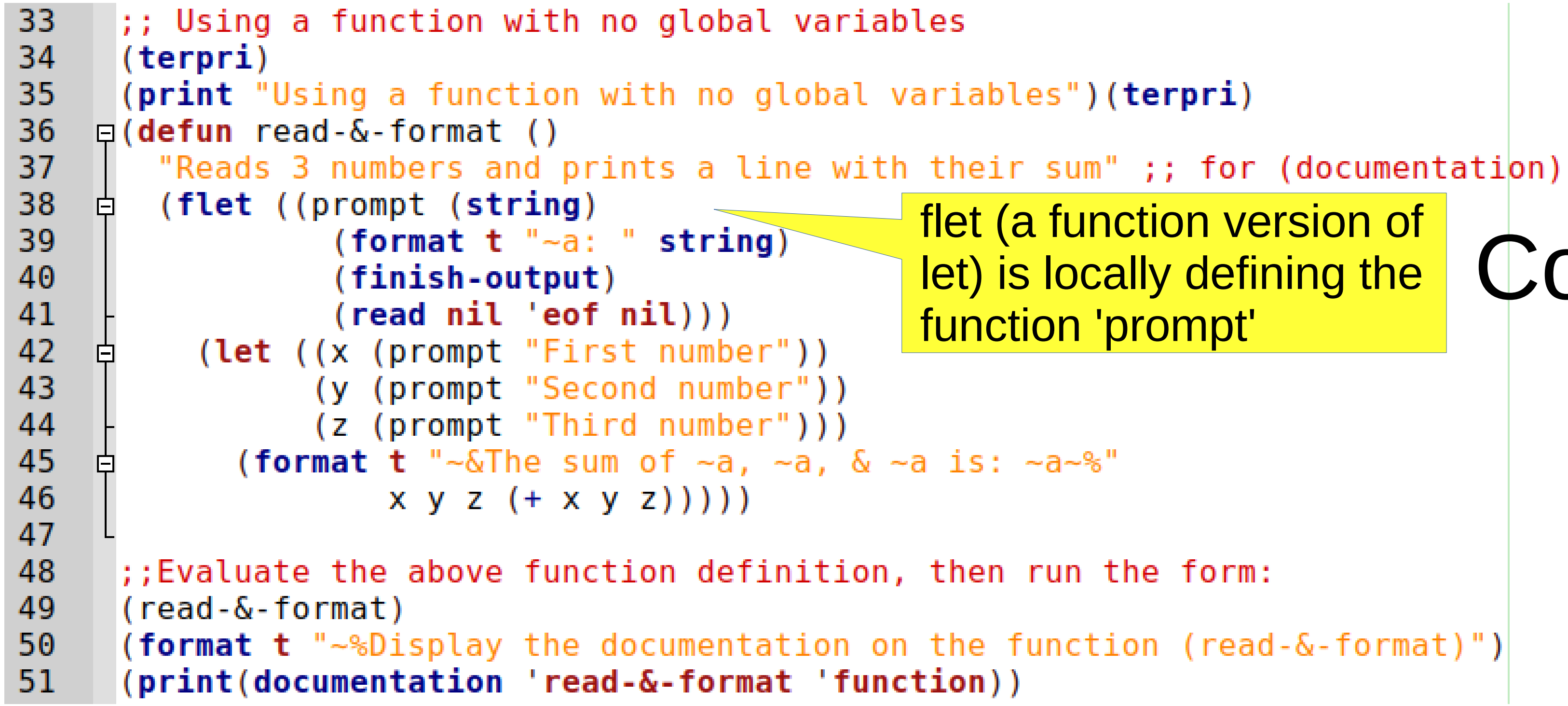
```lisp
;; Input from Console. May execute out-of-order on SBCL?
(print "Input from the console")
(terpri)
(terpri)
(princ "Using read. Enter your first name: ")
(princ(read))

(terpri)
(princ "Using read-line. Enter your surname: ")
(princ(read-line))

;; Using a function to read the input
(terpri)
(print "Using a function to Read input from console")
(terpri)
(defvar a)
(defun get-console-input ()
    (princ "Enter some data: ")
    (setq a (read))(princ a))
    ;;(princ(setq a (read))))

;; Call function
(get-console-input)
```



```
"Input from the console"

Using read. Enter your first name: Ian
IAN
Using read-line. Enter your surname: Stewart
Stewart

"Using a function to Read input from console"
Enter some data: Jake
JAKE
```

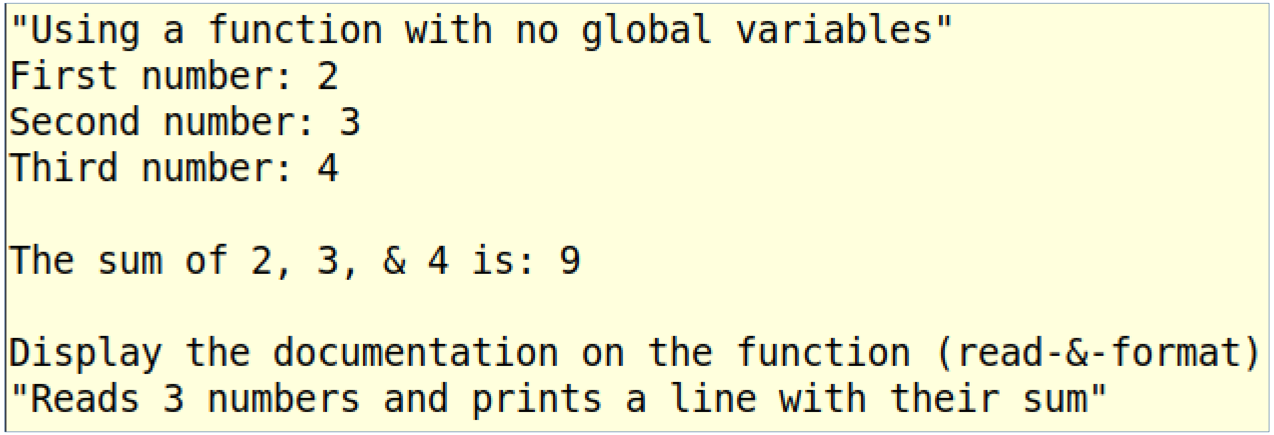
File: demo_input.lisp

Input to Console. Continued...



```lisp
;; Using a function with no global variables
(terpri)
(print "Using a function with no global variables")(terpri)
(defun read-&-format ()
  "Reads 3 numbers and prints a line with their sum" ;; for (documentation)
  (flet ((prompt (string)
           (format t "~a: " string)
           (finish-output)
           (read nil 'eof nil)))
    (let ((x (prompt "First number"))
          (y (prompt "Second number"))
          (z (prompt "Third number")))
      (format t "~&The sum of ~a, ~a, & ~a is: ~a~%"
              x y z (+ x y z)))))

;;Evaluate the above function definition, then run the form:
(read-&-format)
(format t "~%Display the documentation on the function (read-&-format)")
(print(documentation 'read-&-format 'function))
```

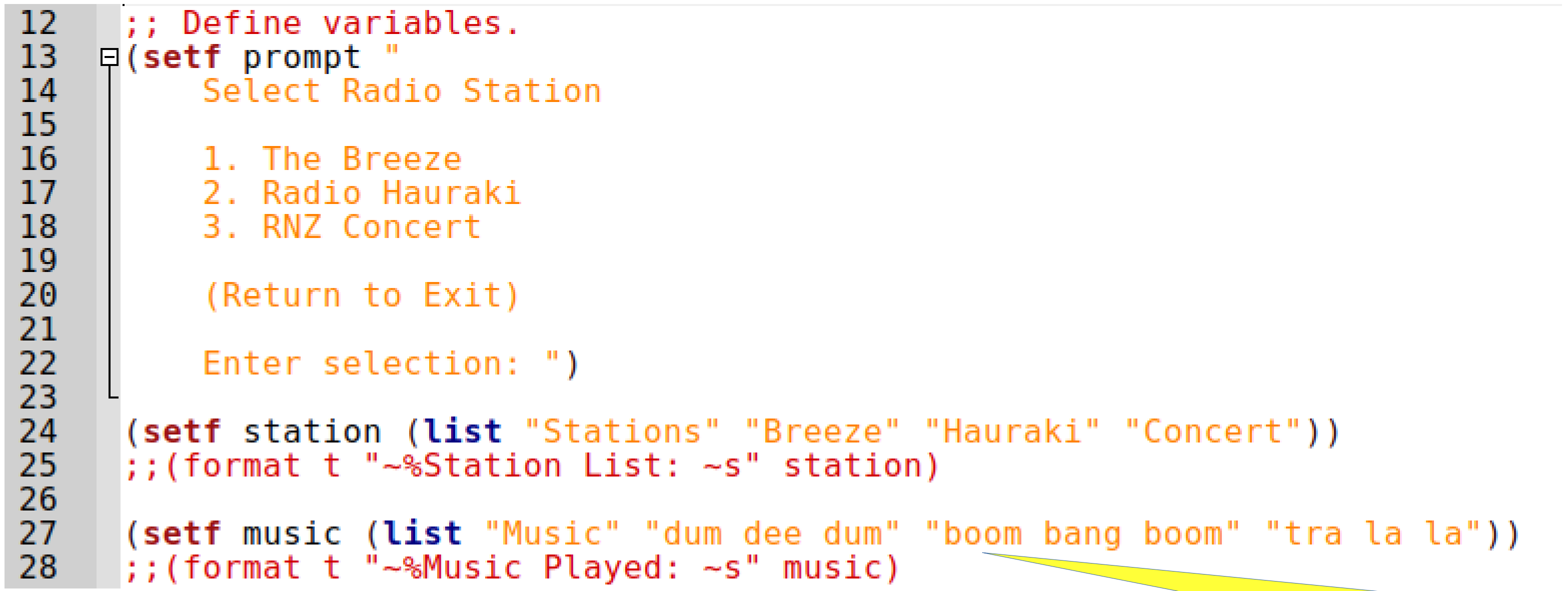
flet (a function version of let) is locally defining the function 'prompt'



```
"Using a function with no global variables"
First number: 2
Second number: 3
Third number: 4

The sum of 2, 3, & 4 is: 9

Display the documentation on the function (read-&-format)
"Reads 3 numbers and prints a line with their sum"
```

File: demo_input.lisp

Input to Console. Mock Radio Menu

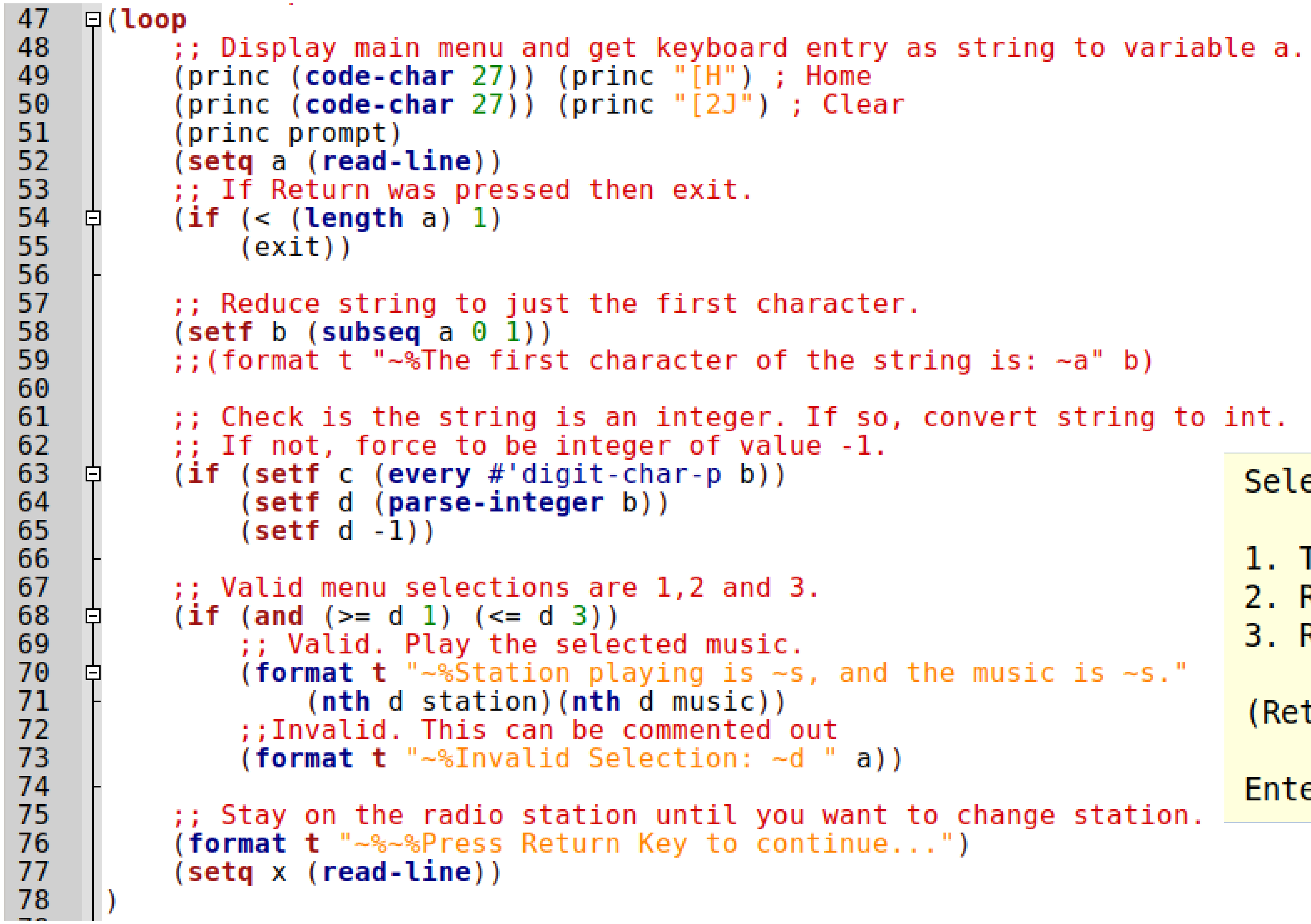


```lisp
;; Define variables.
(setf prompt "
    Select Radio Station

    1. The Breeze
    2. Radio Hauraki
    3. RNZ Concert

    (Return to Exit)

    Enter selection: ")

(setf station (list "Stations" "Breeze" "Hauraki" "Concert"))
;;(format t "~%Station List: ~s" station)

(setf music (list "Music" "dum dee dum" "boom bang boom" "tra la la"))
;;(format t "~%Music Played: ~s" music)
```

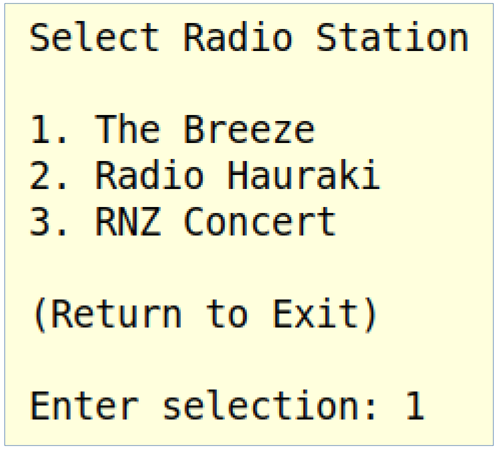
Should be a list of URL’s to Radio Stations, so audio is streamed.

File: demo_radio.lisp

Radio Menu...



```lisp
(loop
    ;; Display main menu and get keyboard entry as string to variable a.
    (princ (code-char 27)) (princ "[H") ; Home
    (princ (code-char 27)) (princ "[2J") ; Clear
    (princ prompt)
    (setq a (read-line))
    ;; If Return was pressed then exit.
    (if (< (length a) 1)
        (exit))

    ;; Reduce string to just the first character.
    (setf b (subseq a 0 1))
    ;;(format t "~%The first character of the string is: ~a" b)

    ;; Check is the string is an integer. If so, convert string to int.
    ;; If not, force to be integer of value -1.
    (if (setf c (every #'digit-char-p b))
        (setf d (parse-integer b))
        (setf d -1))

    ;; Valid menu selections are 1,2 and 3.
    (if (and (>= d 1) (<= d 3))
        ;; Valid. Play the selected music.
        (format t "~%Station playing is ~s, and the music is ~s."
            (nth d station)(nth d music))
        ;;Invalid. This can be commented out
        (format t "~%Invalid Selection: ~d " a))

    ;; Stay on the radio station until you want to change station.
    (format t "~%~%Press Return Key to continue...")
    (setq x (read-line))
)
```



```
Select Radio Station

1. The Breeze
2. Radio Hauraki
3. RNZ Concert

(Return to Exit)

Enter selection: 1
```

File: demo_radio.lisp

SBCL Command line Help

```
$ sbcl --help
Usage: sbcl [runtime-options] [toplevel-options] [user-options]
Common runtime options:
  --help                     Print this message and exit.
  --version                  Print version information and exit.
  --core <filename>          Use the specified core file instead of the default.
  --dynamic-space-size <MiB> Size of reserved dynamic space in megabytes.
  --control-stack-size <MiB> Size of reserved control stack in megabytes.
  --tls-limit                Maximum number of thread-local symbols.

Common toplevel options:
  --sysinit <filename>       System-wide init-file to use instead of default.
  --userinit <filename>      Per-user init-file to use instead of default.
  --no-sysinit               Inhibit processing of any system-wide init-file.
  --no-userinit              Inhibit processing of any per-user init-file.
  --disable-debugger         Invoke sb-ext:disable-debugger.
  --noprint                  Run a Read-Eval Loop without printing results.
  --script [<filename>]      Skip #! line, disable debugger, avoid verbosity.
  --quit                     Exit with code 0 after option processing.
  --non-interactive          Sets both --quit and --disable-debugger.
Common toplevel options that are processed in order:
  --eval <form>              Form to eval when processing this option.
  --load <filename>          File to load when processing this option.
```

Avoid error from Shebang

Error if Shebang

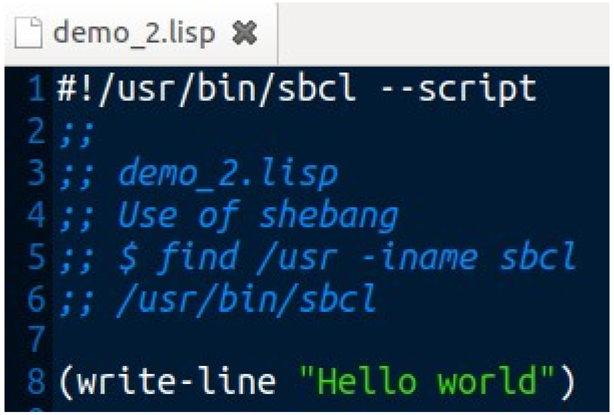
SBCL Command line help, continued...

```
User options are not processed by SBCL. All runtime options must
appear before toplevel options, and all toplevel options must
appear before user options.

For more information please refer to the SBCL User Manual, which
should be installed along with SBCL, and is also available from the
website <http://www.sbcl.org/>.
```

http://www.sbcl.org/manual/index.html

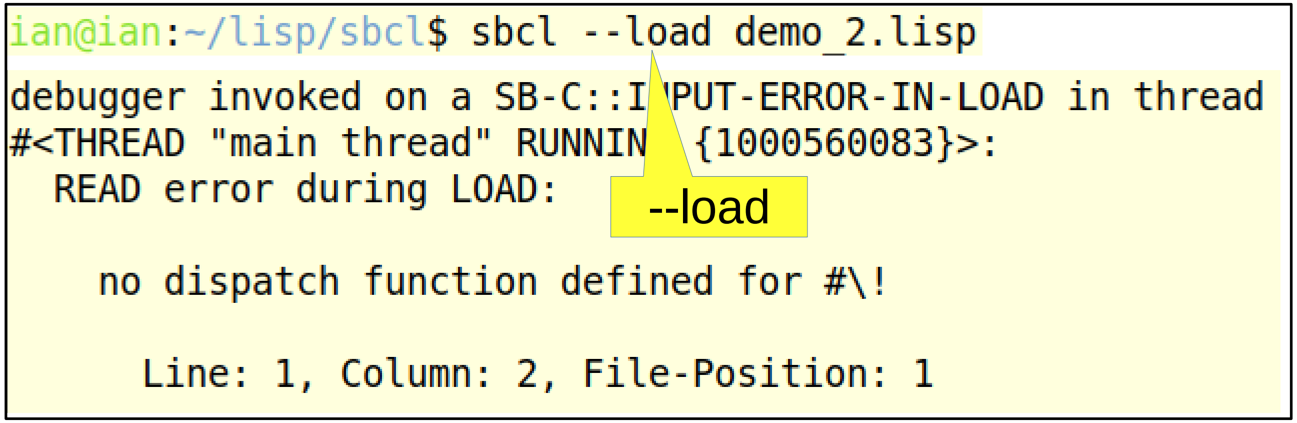
http://www.sbcl.org/manual/sbcl.pdf

SBCL Shebang #!/usr/bin/sbcl --script

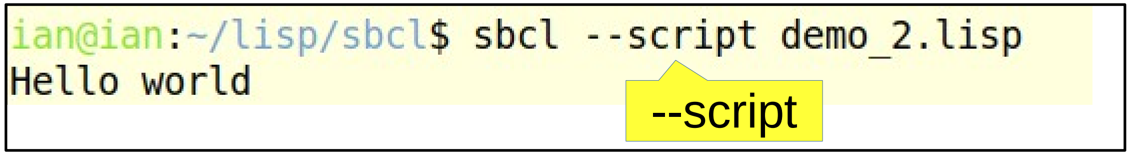


demo_2.lisp

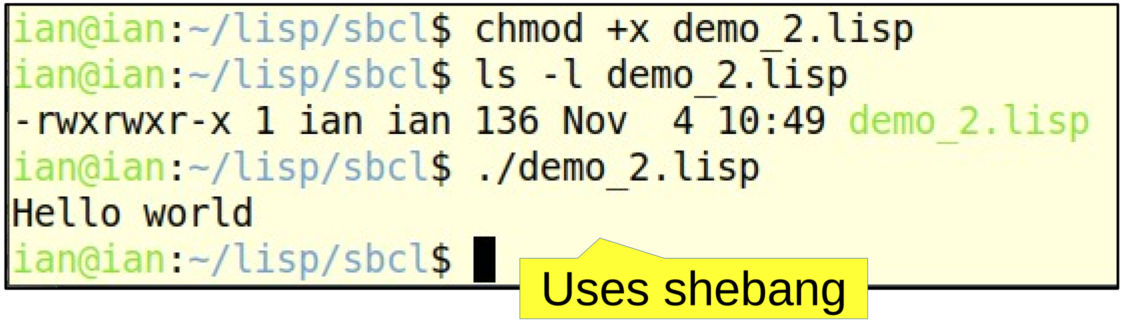
```
#!/usr/bin/sbcl --script
;;
;; demo_2.lisp
;; Use of shebang
;; $ find /usr -iname sbcl
;; /usr/bin/sbcl

(write-line "Hello world")
```



```
ian@ian:~/lisp/sbcl$ sbcl --load demo_2.lisp
debugger invoked on a SB-C::INPUT-ERROR-IN-LOAD in thread
#<THREAD "main thread" RUNNING {100560083}>:
  READ error during LOAD:

    no dispatch function defined for #\!

      Line: 1, Column: 2, File-Position: 1
```



```
ian@ian:~/lisp/sbcl$ sbcl --script demo_2.lisp
Hello world
```



```
ian@ian:~/lisp/sbcl$ chmod +x demo_2.lisp
ian@ian:~/lisp/sbcl$ ls -l demo_2.lisp
-rwxrwxr-x 1 ian ian 136 Nov  4 10:49 demo_2.lisp
ian@ian:~/lisp/sbcl$ ./demo_2.lisp
Hello world
ian@ian:~/lisp/sbcl$ 
```

File: demo_2.lisp

COMMON-LISP Package

To list the Packages:

```
    (print (list-all-packages))
```

The next 10 slides show symbols in the Common-Lisp packet.

- Sorted from Lisp command:

```
    (do-external-symbols (s (find-package "COMMON-LISP"))(print s))
```

- 978 Symbols
- Red for mathematical functions
- Green for Trigonometric functions

Common-Lisp 1/10

-1	*LOAD-VERBOSE*	*RANDOM-STATE*	&REST	ALPHANUMERICP
1	*MACROEXPAND-HOOK*	*READ-BASE*	&WHOLE	AND
-	*MODULES*	*READ-DEFAULT-FLOAT-FORMAT*	+	APPEND
*	*PACKAGE*	*READ-EVAL*	++	APPLY
**	*PRINT-ARRAY*	*READ-SUPPRESS*	+++	APROPOS
***	*PRINT-BASE*	*READTABLE*	<	APROPOS-LIST
BREAK-ON-SIGNALS	*PRINT-CASE*	*STANDARD-INPUT*	<=	AREF
COMPILE-FILE-PATHNAME	*PRINT-CIRCLE*	*STANDARD-OUTPUT*	>	ARITHMETIC-ERROR
COMPILE-FILE-TRUENAME	*PRINT-ESCAPE*	*TERMINAL-IO*	>=	ARITHMETIC-ERROR-OPERANDS
COMPILE-PRINT	*PRINT-GENSYM*	*TRACE-OUTPUT*	ABORT	ARITHMETIC-ERROR-OPERATION
COMPILE-VERBOSE	*PRINT-LENGTH*	/	ABS	ARRAY
DEBUG-IO	*PRINT-LEVEL*	//	ACONS	ARRAY-DIMENSION
DEBUGGER-HOOK	*PRINT-LINES*	///	ACOS	ARRAY-DIMENSION-LIMIT
DEFAULT-PATHNAME-DEFAULTS	*PRINT-MISER-WIDTH*	/=	ACOSH	ARRAY-DIMENSIONS
ERROR-OUTPUT	*PRINT-PPRINT-DISPATCH*	&ALLOW-OTHER-KEYS	ADD-METHOD	ARRAY-DISPLACEMENT
FEATURES	*PRINT-PRETTY*	&AUX	ADJOIN	ARRAY-ELEMENT-TYPE
GENSYM-COUNTER	*PRINT-RADIX*	&BODY	ADJUST-ARRAY	ARRAY-HAS-FILL-POINTER-P
LOAD-PATHNAME	*PRINT-READABLY*	&ENVIRONMENT	ADJUSTABLE-ARRAY-P	ARRAY-IN-BOUNDS-P
LOAD-PRINT	*PRINT-RIGHT-MARGIN*	&KEY	ALLOCATE-INSTANCE	ARRAY-RANK
LOAD-TRUENAME	*QUERY-IO*	&OPTIONAL	ALPHA-CHAR-P	ARRAY-RANK-LIMIT

Common-Lisp 2/10

ARRAY-ROW-MAJOR-INDEX	BIT-ANDC2	BOOLE-CLR	CAAAAR	CCASE
ARRAY-TOTAL-SIZE	BIT-EQV	BOOLE-EQV	CAAADR	CDAAAR
ARRAY-TOTAL-SIZE-LIMIT	BIT-IOR	BOOLE-IOR	CAAAR	CDAADR
ARRAYP	BIT-NAND	BOOLE-NAND	CAADAR	CDAAR
ASH	BIT-NOR	BOOLE-NOR	CAADDR	CDADAR
ASIN	BIT-NOT	BOOLE-ORC1	CAADR	CDADDR
ASINH	BIT-ORC1	BOOLE-ORC2	CAAR	CDADR
ASSERT	BIT-ORC2	BOOLE-SET	CADAAR	CDAR
ASSOC	BIT-VECTOR	BOOLE-XOR	CADADR	CDDAAR
ASSOC-IF	BIT-VECTOR-P	BOOLEAN	CADAR	CDDADR
ASSOC-IF-NOT	BIT-XOR	BOTH-CASE-P	CADDAR	CDDAR
ATAN	BLOCK	BOUNDP	CADDDR	CDDDAR
ATANH	BOOLE	BREAK	CADDR	CDDDDR
ATOM	BOOLE-1	BROADCAST-STREAM	CADR	CDDDR
BASE-CHAR	BOOLE-2	BROADCAST-STREAM-STREAMS	CALL-ARGUMENTS-LIMIT	CDDR
BASE-STRING	BOOLE-AND	BUILT-IN-CLASS	CALL-METHOD	CDR
BIGNUM	BOOLE-ANDC1	BUTLAST	CALL-NEXT-METHOD	CEILING
BIT	BOOLE-ANDC2	BYTE	CAR	CELL-ERROR
BIT-AND	BOOLE-C1	BYTE-POSITION	CASE	CELL-ERROR-NAME
BIT-ANDC1	BOOLE-C2	BYTE-SIZE	CATCH	CERROR

Common-Lisp 3/10

CHANGE-CLASS	CHARACTER	COMPILER-MACRO-FUNCTION	COPY-PPRINT-DISPATCH	DEFCONSTANT
CHAR	CHARACTERP	COMPLEMENT	COPY-READTABLE	DEFGENERIC
CHAR-CODE	CHECK-TYPE	COMPLEX	COPY-SEQ	DEFINE-COMPILER-MACRO
CHAR-CODE-LIMIT	CIS	COMPLEXP	COPY-STRUCTURE	DEFINE-CONDITION
CHAR-DOWNCASE	CLASS	COMPUTE-APPLICABLE-METHODS	COPY-SYMBOL	DEFINE-METHOD-COMBINATION
CHAR-EQUAL	CLASS-NAME	COMPUTE-RESTARTS	COPY-TREE	DEFINE-MODIFY-MACRO
CHAR-GREATERP	CLASS-OF	CONCATENATE	COS	DEFINE-SETF-EXPANDER
CHAR-INT	CLEAR-INPUT	CONCATENATED-STREAM	COSH	DEFINE-SYMBOL-MACRO
CHAR-LESSP	CLEAR-OUTPUT	CONCATENATED-STREAM-STREAMS	COUNT	DEFMACRO
CHAR-NAME	CLOSE	COND	COUNT-IF	DEFMETHOD
CHAR-NOT-EQUAL	CLRHASH	CONDITION	COUNT-IF-NOT	DEFPACKAGE
CHAR-NOT-GREATERP	CODE-CHAR	CONJUGATE	CTYPECASE	DEFPARAMETER
CHAR-NOT-LESSP	COERCE	CONS	DEBUG	DEFSETF
CHAR-UPCASE	COMPILATION-SPEED	CONSP	DECF	DEFSTRUCT
CHAR/=	COMPILE	CONSTANTLY	DECLAIM	DEFTYPE
CHAR<	COMPILE-FILE	CONSTANTP	DECLARATION	DEFUN
CHAR<=	COMPILE-FILE-PATHNAME	CONTINUE	DECLARE	DEFVAR
CHAR=	COMPILED-FUNCTION	CONTROL-ERROR	DECODE-FLOAT	DELETE
CHAR>	COMPILED-FUNCTION-P	COPY-ALIST	DECODE-UNIVERSAL-TIME	DELETE-DUPLICATES
CHAR>=	COMPILER-MACRO	COPY-LIST	DEFCLASS	DELETE-FILE

Common-Lisp 4/10

DELETE-IF	DOLIST	ENSURE-GENERIC-FUNCTION	FILE-AUTHOR	FINISH-OUTPUT
DELETE-IF-NOT	DOTIMES	EQ	FILE-ERROR	FIRST
DELETE-PACKAGE	DOUBLE-FLOAT	EQL	FILE-ERROR-PATHNAME	FIXNUM
DENOMINATOR	DOUBLE-FLOAT-EPSILON	EQUAL	FILE-LENGTH	FLET
DEPOSIT-FIELD	DOUBLE-FLOAT-NEGATIVE-EPSILON	EQUALP	FILE-NAMESTRING	FLOAT
DESCRIBE	DPB	ERROR	FILE-POSITION	FLOAT-DIGITS
DESCRIBE-OBJECT	DRIBBLE	ETYPECASE	FILE-STREAM	FLOAT-PRECISION
DESTRUCTURING-BIND	DYNAMIC-EXTENT	EVAL	FILE-STRING-LENGTH	FLOAT-RADIX
DIGIT-CHAR	ECASE	EVAL-WHEN	FILE-WRITE-DATE	FLOAT-SIGN
DIGIT-CHAR-P	ECHO-STREAM	EVENP	FILL	FLOATING-POINT-INEXACT
DIRECTORY	ECHO-STREAM-INPUT-STREAM	EVERY	FILL-POINTER	FLOATING-POINT-INVALID-OPERATION
DIRECTORY-NAMESTRING	ECHO-STREAM-OUTPUT-STREAM	EXP	FIND	FLOATING-POINT-OVERFLOW
DISASSEMBLE	ED	EXPORT	FIND-ALL-SYMBOLS	FLOATING-POINT-UNDERFLOW
DIVISION-BY-ZERO	EIGHTH	EXPT	FIND-CLASS	FLOATP
DO	ELT	EXTENDED-CHAR	FIND-IF	FLOOR
DO-ALL-SYMBOLS	ENCODE-UNIVERSAL-TIME	FBOUNDP	FIND-IF-NOT	FMAKUNBOUND
DO-EXTERNAL-SYMBOLS	END-OF-FILE	FCEILING	FIND-METHOD	FORCE-OUTPUT
DO-SYMBOLS	ENDP	FDEFINITION	FIND-PACKAGE	FORMAT
DO*	ENOUGH-NAMESTRING	FFLOOR	FIND-RESTART	FORMATTER
DOCUMENTATION	ENSURE-DIRECTORIES-EXIST	FIFTH	FIND-SYMBOL	FOURTH

Common-Lisp 5/10

FRESH-LINE	GET-PROPERTIES	IGNORE	INVOKE-RESTART	LEAST-NEGATIVE-SHORT-FLOAT
FROUND	GET-SETF-EXPANSION	IGNORE-ERRORS	INVOKE-RESTART-INTERACTIVELY	LEAST-NEGATIVE-SINGLE-FLOAT
FTRUNCATE	GET-UNIVERSAL-TIME	IMAGPART	ISQRT	LEAST-POSITIVE-DOUBLE-FLOAT
FTYPE	GETF	IMPORT	KEYWORD	LEAST-POSITIVE-LONG-FLOAT
FUNCALL	GETHASH	IN-PACKAGE	KEYWORDP	LEAST-POSITIVE-NORMALIZED-DOUBLE-FLOAT
FUNCTION	GO	INCF	LABELS	LEAST-POSITIVE-NORMALIZED-LONG-FLOAT
FUNCTION-KEYWORDS	GRAPHIC-CHAR-P	INITIALIZE-INSTANCE	LAMBDA	LEAST-POSITIVE-NORMALIZED-SHORT-FLOAT
FUNCTION-LAMBDA-EXPRESSION	HANDLER-BIND	INLINE	LAMBDA-LIST-KEYWORDS	LEAST-POSITIVE-NORMALIZED-SINGLE-FLOAT
FUNCTIONP	HANDLER-CASE	INPUT-STREAM-P	LAMBDA-PARAMETERS-LIMIT	LEAST-POSITIVE-SHORT-FLOAT
GCD	HASH-TABLE	INSPECT	LAST	LEAST-POSITIVE-SINGLE-FLOAT
GENERIC-FUNCTION	HASH-TABLE-COUNT	INTEGER	LCM	LENGTH
GENSYM	HASH-TABLE-P	INTEGER-DECODE-FLOAT	LDB	LET
GENTEMP	HASH-TABLE-REHASH-SIZE	INTEGER-LENGTH	LDB-TEST	LET*
GET	HASH-TABLE-REHASH-THRESHOLD	INTEGERP	LDIFF	LISP-IMPLEMENTATION-TYPE
GET-DECODED-TIME	HASH-TABLE-SIZE	INTERACTIVE-STREAM-P	LEAST-NEGATIVE-DOUBLE-FLOAT	LISP-IMPLEMENTATION-VERSION
GET-DISPATCH-MACRO-CHARACTER	HASH-TABLE-TEST	INTERN	LEAST-NEGATIVE-LONG-FLOAT	LIST
GET-INTERNAL-REAL-TIME	HOST-NAMESTRING	INTERNAL-TIME-UNITS-PER-SECOND	LEAST-NEGATIVE-NORMALIZED-DOUBLE-FLOAT	LIST-ALL-PACKAGES
GET-INTERNAL-RUN-TIME	IDENTITY	INTERSECTION	LEAST-NEGATIVE-NORMALIZED-LONG-FLOAT	LIST-LENGTH
GET-MACRO-CHARACTER	IF	INVALID-METHOD-ERROR	LEAST-NEGATIVE-NORMALIZED-SHORT-FLOAT	LIST*
GET-OUTPUT-STREAM-STRING	IGNORABLE	INVOKE-DEBUGGER	LEAST-NEGATIVE-NORMALIZED-SINGLE-FLOAT	LISTEN

Common-Lisp 6/10

LISTP	LOGTEST	MAKE-DISPATCH-MACRO-CHARACTER	MAP	MIN
LOAD	LOGXOR	MAKE-ECHO-STREAM	MAP-INTO	MINUSP
LOAD-LOGICAL-PATHNAME-TRANSLATIONS	LONG-FLOAT	MAKE-HASH-TABLE	MAPC	MISMATCH
LOAD-TIME-VALUE	LONG-FLOAT-EPSILON	MAKE-INSTANCE	MAPCAN	MOD
LOCALLY	LONG-FLOAT-NEGATIVE-EPSILON	MAKE-INSTANCES-OBSOLETE	MAPCAR	MOST-NEGATIVE-DOUBLE-FLOAT
LOG	LONG-SITE-NAME	MAKE-LIST	MAPCON	MOST-NEGATIVE-FIXNUM
LOGAND	LOOP	MAKE-LOAD-FORM	MAPHASH	MOST-NEGATIVE-LONG-FLOAT
LOGANDC1	LOOP-FINISH	MAKE-LOAD-FORM-SAVING-SLOTS	MAPL	MOST-NEGATIVE-SHORT-FLOAT
LOGANDC2	LOWER-CASE-P	MAKE-METHOD	MAPLIST	MOST-NEGATIVE-SINGLE-FLOAT
LOGBITP	MACHINE-INSTANCE	MAKE-PACKAGE	MASK-FIELD	MOST-POSITIVE-DOUBLE-FLOAT
LOGCOUNT	MACHINE-TYPE	MAKE-PATHNAME	MAX	MOST-POSITIVE-FIXNUM
LOGEQV	MACHINE-VERSION	MAKE-RANDOM-STATE	MEMBER	MOST-POSITIVE-LONG-FLOAT
LOGICAL-PATHNAME	MACRO-FUNCTION	MAKE-SEQUENCE	MEMBER-IF	MOST-POSITIVE-SHORT-FLOAT
LOGICAL-PATHNAME-TRANSLATIONS	MACROEXPAND	MAKE-STRING	MEMBER-IF-NOT	MOST-POSITIVE-SINGLE-FLOAT
LOGIOR	MACROEXPAND-1	MAKE-STRING-INPUT-STREAM	MERGE	MUFFLE-WARNING
LOGNAND	MACROLET	MAKE-STRING-OUTPUT-STREAM	MERGE-PATHNAMES	MULTIPLE-VALUE-BIND
LOGNOR	MAKE-ARRAY	MAKE-SYMBOL	METHOD	MULTIPLE-VALUE-CALL
LOGNOT	MAKE-BROADCAST-STREAM	MAKE-SYNONYM-STREAM	METHOD-COMBINATION	MULTIPLE-VALUE-LIST
LOGORC1	MAKE-CONCATENATED-STREAM	MAKE-TWO-WAY-STREAM	METHOD-COMBINATION-ERROR	MULTIPLE-VALUE-PROG1
LOGORC2	MAKE-CONDITION	MAKUNBOUND	METHOD-QUALIFIERS	MULTIPLE-VALUE-SETQ

Common-Lisp 7/10

MULTIPLE-VALUES-LIMIT	NSTRING-DOWNCASE	OPTIMIZE	PATHNAME-HOST	PPRINT-LOGICAL-BLOCK
NAME-CHAR	NSTRING-UPCASE	OR	PATHNAME-MATCH-P	PPRINT-NEWLINE
NAMESTRING	NSUBLIS	OTHERWISE	PATHNAME-NAME	PPRINT-POP
NBUTLAST	NSUBST	OUTPUT-STREAM-P	PATHNAME-TYPE	PPRINT-TAB
NCONC	NSUBST-IF	PACKAGE	PATHNAME-VERSION	PPRINT-TABULAR
NEXT-METHOD-P	NSUBST-IF-NOT	PACKAGE-ERROR	PATHNAMEP	PRIN1
NIL	NSUBSTITUTE	PACKAGE-ERROR-PACKAGE	PEEK-CHAR	PRIN1-TO-STRING
NINTERSECTION	NSUBSTITUTE-IF	PACKAGE-NAME	PHASE	PRINC
NINTH	NSUBSTITUTE-IF-NOT	PACKAGE-NICKNAMES	PI	PRINC-TO-STRING
NO-APPLICABLE-METHOD	NTH	PACKAGE-SHADOWING-SYMBOLS	PLUSP	PRINT
NO-NEXT-METHOD	NTH-VALUE	PACKAGE-USE-LIST	POP	PRINT-NOT-READABLE
NOT	NTHCDR	PACKAGE-USED-BY-LIST	POSITION	PRINT-NOT-READABLE-OBJECT
NOTANY	NULL	PACKAGEP	POSITION-IF	PRINT-OBJECT
NOTEVERY	NUMBER	PAIRLIS	POSITION-IF-NOT	PRINT-UNREADABLE-OBJECT
NOTINLINE	NUMBERP	PARSE-ERROR	PPRINT	PROBE-FILE
NRECONC	NUMERATOR	PARSE-INTEGER	PPRINT-DISPATCH	PROCLAIM
NREVERSE	NUNION	PARSE-NAMESTRING	PPRINT-EXIT-IF-LIST-EXHAUSTED	PROG
NSET-DIFFERENCE	ODDP	PATHNAME	PPRINT-FILL	PROG*
NSET-EXCLUSIVE-OR	OPEN	PATHNAME-DEVICE	PPRINT-INDENT	PROG1
NSTRING-CAPITALIZE	OPEN-STREAM-P	PATHNAME-DIRECTORY	PPRINT-LINEAR	PROG2

Common-Lisp 8/10

PROGN	READ-BYTE	REMOVE	ROTATEF	SET-SYNTAX-FROM-CHAR
PROGRAM-ERROR	READ-CHAR	REMOVE-DUPLICATES	ROUND	SETF
PROGV	READ-CHAR-NO-HANG	REMOVE-IF	ROW-MAJOR-AREF	SETQ
PROVIDE	READ-DELIMITED-LIST	REMOVE-IF-NOT	RPLACA	SEVENTH
PSETF	READ-FROM-STRING	REMOVE-METHOD	RPLACD	SHADOW
PSETQ	READ-LINE	REMPROP	SAFETY	SHADOWING-IMPORT
PUSH	READ-PRESERVING-WHITESPACE	RENAME-FILE	SATISFIES	SHARED-INITIALIZE
PUSHNEW	READ-SEQUENCE	RENAME-PACKAGE	SBIT	SHIFTF
QUOTE	READER-ERROR	REPLACE	SCALE-FLOAT	SHORT-FLOAT
RANDOM	READTABLE	REQUIRE	SCHAR	SHORT-FLOAT-EPSILON
RANDOM-STATE	READTABLE-CASE	REST	SEARCH	SHORT-FLOAT-NEGATIVE-EPSILON
RANDOM-STATE-P	READTABLEP	RESTART	SECOND	SHORT-SITE-NAME
RASSOC	REAL	RESTART-BIND	SEQUENCE	SIGNAL
RASSOC-IF	REALP	RESTART-CASE	SERIOUS-CONDITION	SIGNED-BYTE
RASSOC-IF-NOT	REALPART	RESTART-NAME	SET	SIGNUM
RATIO	REDUCE	RETURN	SET-DIFFERENCE	SIMPLE-ARRAY
RATIONAL	REINITIALIZE-INSTANCE	RETURN-FROM	SET-DISPATCH-MACRO-CHARACTER	SIMPLE-BASE-STRING
RATIONALIZE	REM	REVAPPEND	SET-EXCLUSIVE-OR	SIMPLE-BIT-VECTOR
RATIONALP	REMF	REVERSE	SET-MACRO-CHARACTER	SIMPLE-BIT-VECTOR-P
READ	REMHASH	ROOM	SET-PPRINT-DISPATCH	SIMPLE-CONDITION

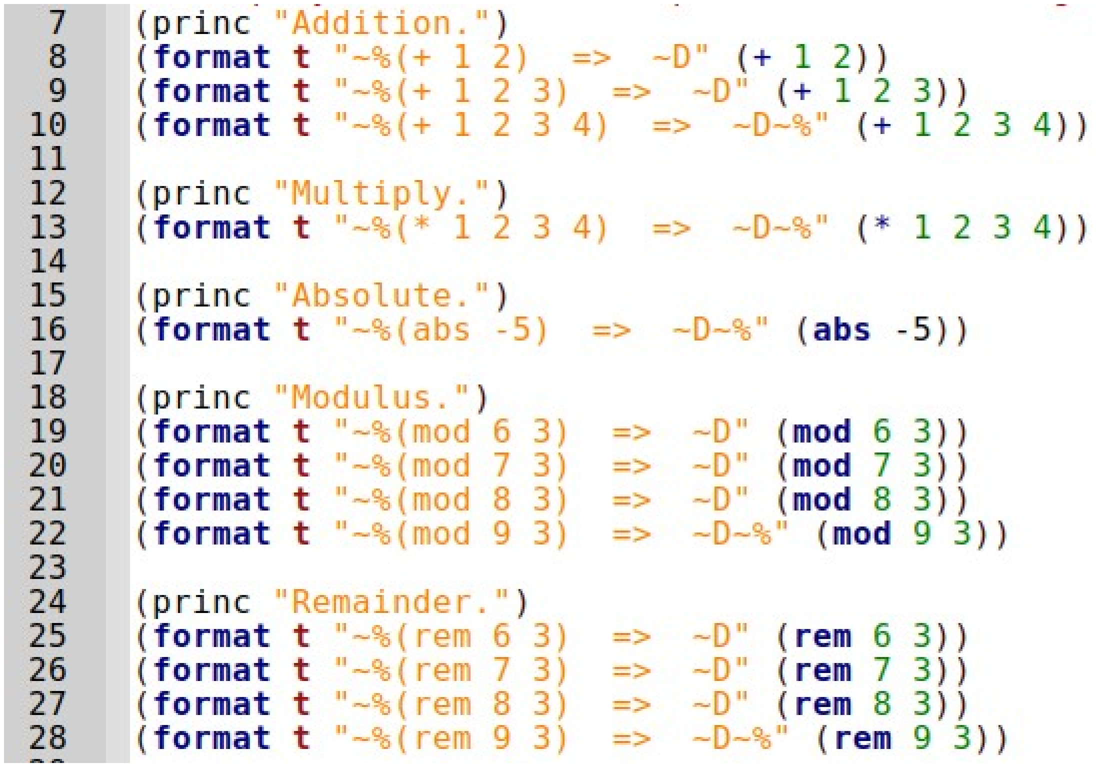
Common-Lisp 9/10

SIMPLE-CONDITION-FORMAT-ARGUMENTS	SLOT-UNBOUND	STORAGE-CONDITION	STRING-TRIM	SUBSTITUTE-IF
SIMPLE-CONDITION-FORMAT-CONTROL	SLOT-VALUE	STORE-VALUE	STRING-UPCASE	SUBSTITUTE-IF-NOT
SIMPLE-ERROR	SOFTWARE-TYPE	STREAM	STRING/=	SUBTYPEP
SIMPLE-STRING	SOFTWARE-VERSION	STREAM-ELEMENT-TYPE	STRING<	SVREF
SIMPLE-STRING-P	SOME	STREAM-ERROR	STRING<=	SXHASH
SIMPLE-TYPE-ERROR	SORT	STREAM-ERROR-STREAM	STRING=	SYMBOL
SIMPLE-VECTOR	SPACE	STREAM-EXTERNAL-FORMAT	STRING>	SYMBOL-FUNCTION
SIMPLE-VECTOR-P	SPECIAL	STREAMP	STRING>=	SYMBOL-MACROLET
SIMPLE-WARNING	SPECIAL-OPERATOR-P	STRING	STRINGP	SYMBOL-NAME
SIN	SPEED	STRING-CAPITALIZE	STRUCTURE	SYMBOL-PACKAGE
SINGLE-FLOAT	SQRT	STRING-DOWNCASE	STRUCTURE-CLASS	SYMBOL-PLIST
SINGLE-FLOAT-EPSILON	STABLE-SORT	STRING-EQUAL	STRUCTURE-OBJECT	SYMBOL-VALUE
SINGLE-FLOAT-NEGATIVE-EPSILON	STANDARD	STRING-GREATERP	STYLE-WARNING	SYMBOLP
SINH	STANDARD-CHAR	STRING-LEFT-TRIM	SUBLIS	SYNONYM-STREAM
SIXTH	STANDARD-CHAR-P	STRING-LESSP	SUBSEQ	SYNONYM-STREAM-SYMBOL
SLEEP	STANDARD-CLASS	STRING-NOT-EQUAL	SUBSETP	T
SLOT-BOUNDP	STANDARD-GENERIC-FUNCTION	STRING-NOT-GREATERP	SUBST	TAGBODY
SLOT-EXISTS-P	STANDARD-METHOD	STRING-NOT-LESSP	SUBST-IF	TAILP
SLOT-MAKUNBOUND	STANDARD-OBJECT	STRING-RIGHT-TRIM	SUBST-IF-NOT	TAN
SLOT-MISSING	STEP	STRING-STREAM	SUBSTITUTE	TANH

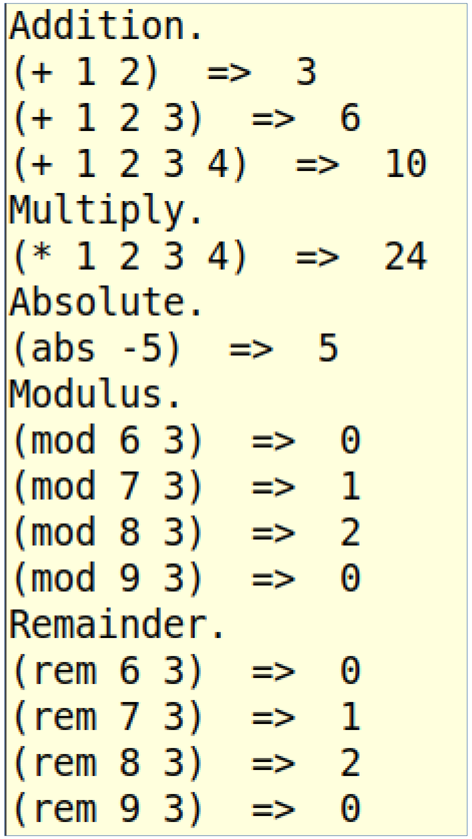
Common-Lisp 10/10

TENTH	TYPECASE	USE-PACKAGE	WITH-OPEN-FILE	
TERPRI	TYPEP	USE-VALUE	WITH-OPEN-STREAM	
THE	UNBOUND-SLOT	USER-HOMEDIR-PATHNAME	WITH-OUTPUT-TO-STRING	
THIRD	UNBOUND-SLOT-INSTANCE	VALUES	WITH-PACKAGE-ITERATOR	
THROW	UNBOUND-VARIABLE	VALUES-LIST	WITH-SIMPLE-RESTART	
TIME	UNDEFINED-FUNCTION	VARIABLE	WITH-SLOTS	
TRACE	UNEXPORT	VECTOR	WITH-STANDARD-IO-SYNTAX	
TRANSLATE-LOGICAL-PATHNAME	UNINTERN	VECTOR-POP	WRITE	
TRANSLATE-PATHNAME	UNION	VECTOR-PUSH	WRITE-BYTE	
TREE-EQUAL	UNLESS	VECTOR-PUSH-EXTEND	WRITE-CHAR	
TRUENAME	UNREAD-CHAR	VECTORP	WRITE-LINE	
TRUNCATE	UNSIGNED-BYTE	WARN	WRITE-SEQUENCE	
TWO-WAY-STREAM	UNTRACE	WARNING	WRITE-STRING	
TWO-WAY-STREAM-INPUT-STREAM	UNUSE-PACKAGE	WHEN	WRITE-TO-STRING	
TWO-WAY-STREAM-OUTPUT-STREAM	UNWIND-PROTECT	WILD-PATHNAME-P	Y-OR-N-P	
TYPE	UPDATE-INSTANCE-FOR-DIFFERENT-CLASS	WITH-ACCESSORS	YES-OR-NO-P	
TYPE-ERROR	UPDATE-INSTANCE-FOR-REDEFINED-CLASS	WITH-COMPILATION-UNIT	ZEROP	
TYPE-ERROR-DATUM	UPGRADED-ARRAY-ELEMENT-TYPE	WITH-CONDITION-RESTARTS	Err:520	
TYPE-ERROR-EXPECTED-TYPE	UPGRADED-COMPLEX-PART-TYPE	WITH-HASH-TABLE-ITERATOR		
TYPE-OF	UPPER-CASE-P	WITH-INPUT-FROM-STRING		

Math Functions



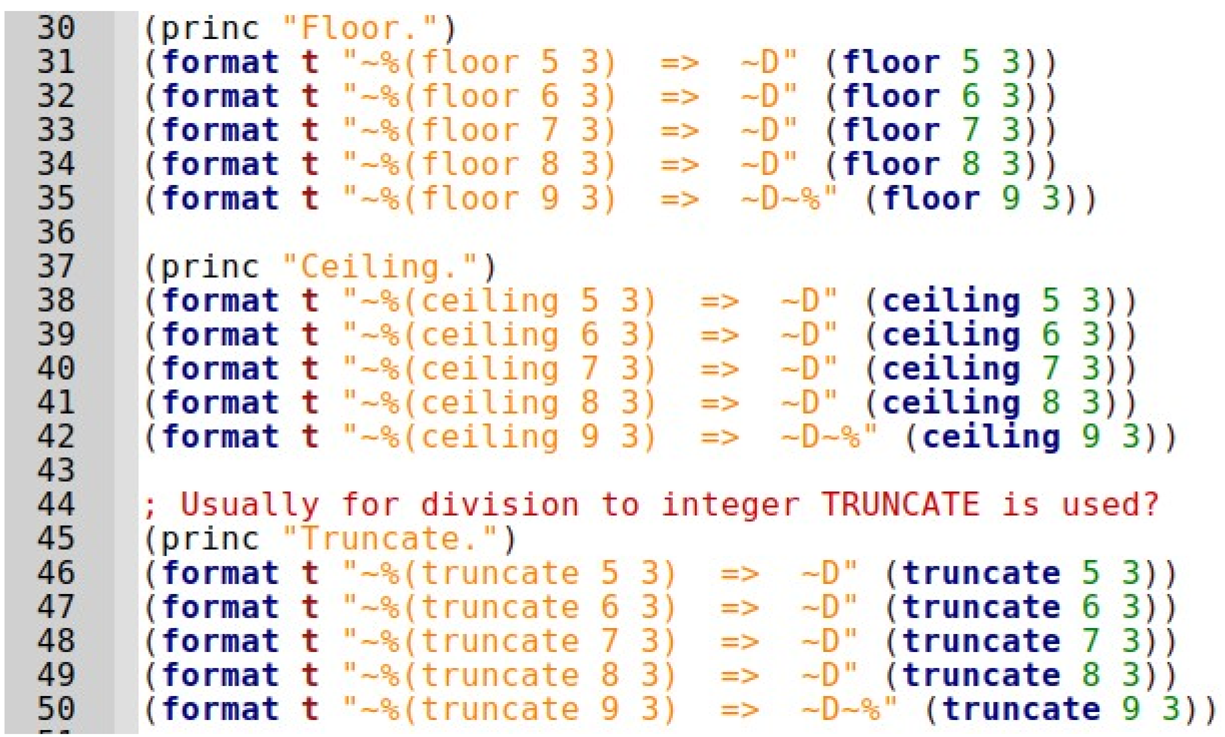
```lisp
(princ "Addition.")
(format t "~%(+ 1 2)  =>  ~D" (+ 1 2))
(format t "~%(+ 1 2 3)  =>  ~D" (+ 1 2 3))
(format t "~%(+ 1 2 3 4)  =>  ~D~%" (+ 1 2 3 4))

(princ "Multiply.")
(format t "~%(* 1 2 3 4)  =>  ~D~%" (* 1 2 3 4))

(princ "Absolute.")
(format t "~%(abs -5)  =>  ~D~%" (abs -5))

(princ "Modulus.")
(format t "~%(mod 6 3)  =>  ~D" (mod 6 3))
(format t "~%(mod 7 3)  =>  ~D" (mod 7 3))
(format t "~%(mod 8 3)  =>  ~D" (mod 8 3))
(format t "~%(mod 9 3)  =>  ~D~%" (mod 9 3))

(princ "Remainder.")
(format t "~%(rem 6 3)  =>  ~D" (rem 6 3))
(format t "~%(rem 7 3)  =>  ~D" (rem 7 3))
(format t "~%(rem 8 3)  =>  ~D" (rem 8 3))
(format t "~%(rem 9 3)  =>  ~D~%" (rem 9 3))
```



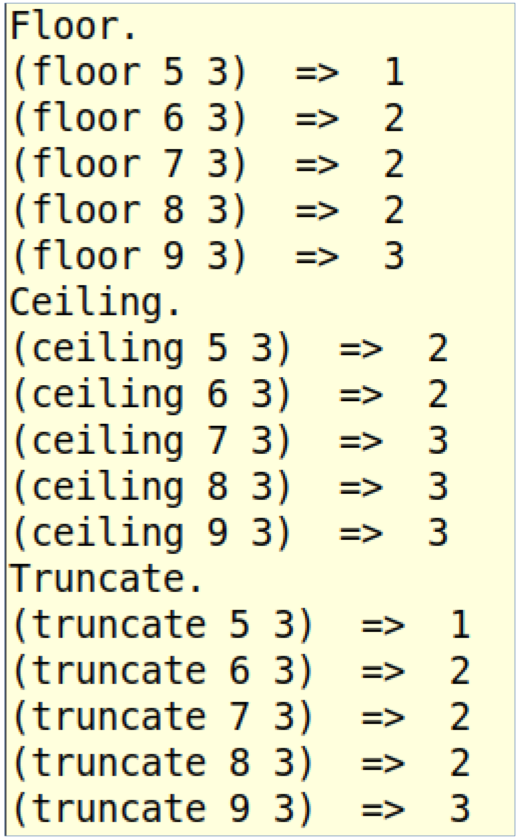
```
Addition.
(+ 1 2)  =>  3
(+ 1 2 3)  =>  6
(+ 1 2 3 4)  =>  10
Multiply.
(* 1 2 3 4)  =>  24
Absolute.
(abs -5)  =>  5
Modulus.
(mod 6 3)  =>  0
(mod 7 3)  =>  1
(mod 8 3)  =>  2
(mod 9 3)  =>  0
Remainder.
(rem 6 3)  =>  0
(rem 7 3)  =>  1
(rem 8 3)  =>  2
(rem 9 3)  =>  0
```

File: demo_math.lisp

Math Functions



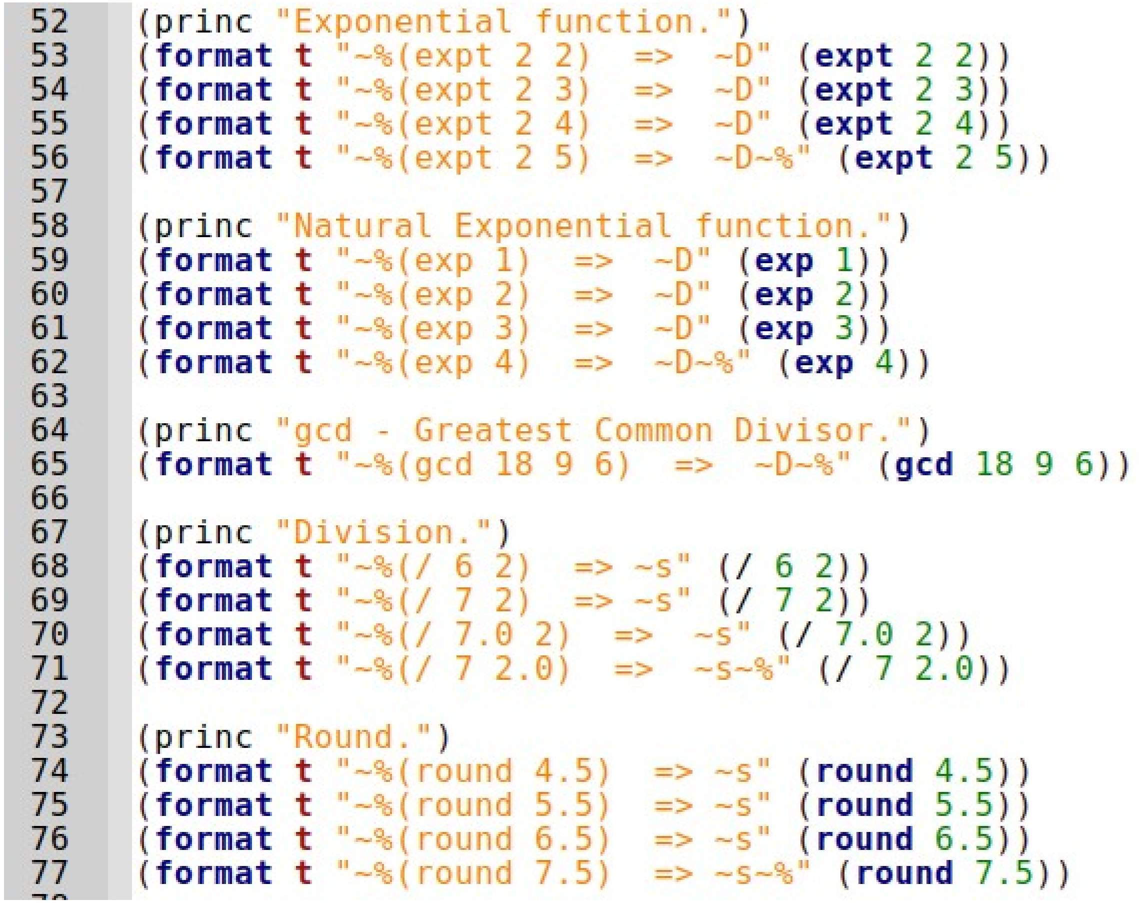
```lisp
(princ "Floor.")
(format t "~%(floor 5 3)  =>  ~D" (floor 5 3))
(format t "~%(floor 6 3)  =>  ~D" (floor 6 3))
(format t "~%(floor 7 3)  =>  ~D" (floor 7 3))
(format t "~%(floor 8 3)  =>  ~D" (floor 8 3))
(format t "~%(floor 9 3)  =>  ~D~%" (floor 9 3))

(princ "Ceiling.")
(format t "~%(ceiling 5 3)  =>  ~D" (ceiling 5 3))
(format t "~%(ceiling 6 3)  =>  ~D" (ceiling 6 3))
(format t "~%(ceiling 7 3)  =>  ~D" (ceiling 7 3))
(format t "~%(ceiling 8 3)  =>  ~D" (ceiling 8 3))
(format t "~%(ceiling 9 3)  =>  ~D~%" (ceiling 9 3))

; Usually for division to integer TRUNCATE is used?
(princ "Truncate.")
(format t "~%(truncate 5 3)  =>  ~D" (truncate 5 3))
(format t "~%(truncate 6 3)  =>  ~D" (truncate 6 3))
(format t "~%(truncate 7 3)  =>  ~D" (truncate 7 3))
(format t "~%(truncate 8 3)  =>  ~D" (truncate 8 3))
(format t "~%(truncate 9 3)  =>  ~D~%" (truncate 9 3))
```



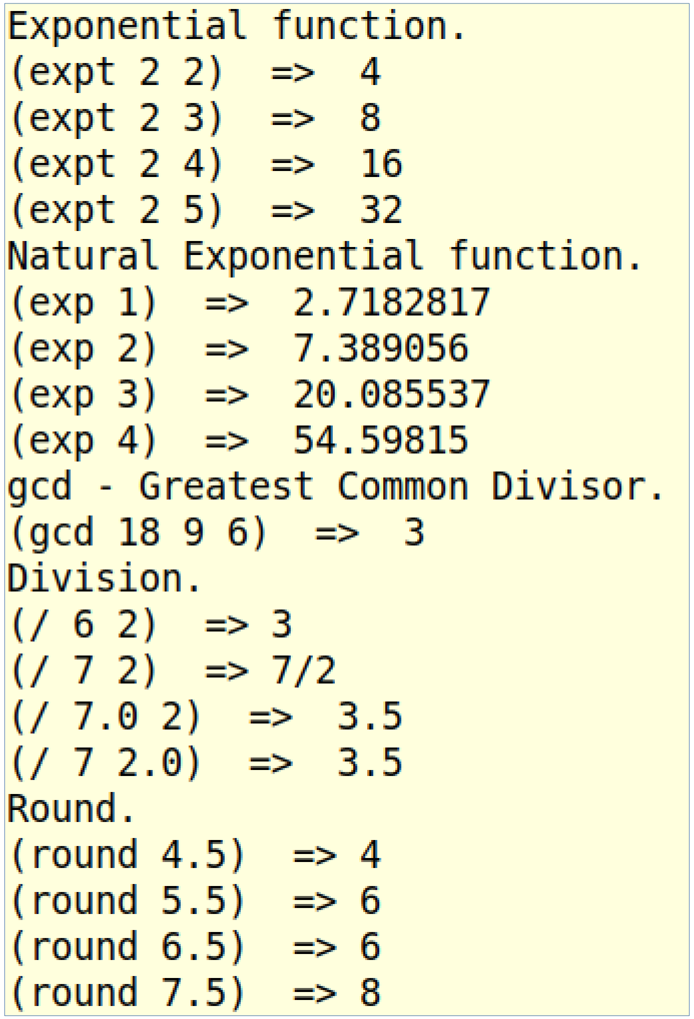
```
Floor.
(floor 5 3)  =>  1
(floor 6 3)  =>  2
(floor 7 3)  =>  2
(floor 8 3)  =>  2
(floor 9 3)  =>  3
Ceiling.
(ceiling 5 3)  =>  2
(ceiling 6 3)  =>  2
(ceiling 7 3)  =>  3
(ceiling 8 3)  =>  3
(ceiling 9 3)  =>  3
Truncate.
(truncate 5 3)  =>  1
(truncate 6 3)  =>  2
(truncate 7 3)  =>  2
(truncate 8 3)  =>  2
(truncate 9 3)  =>  3
```

File: demo_math.lisp

Math Functions



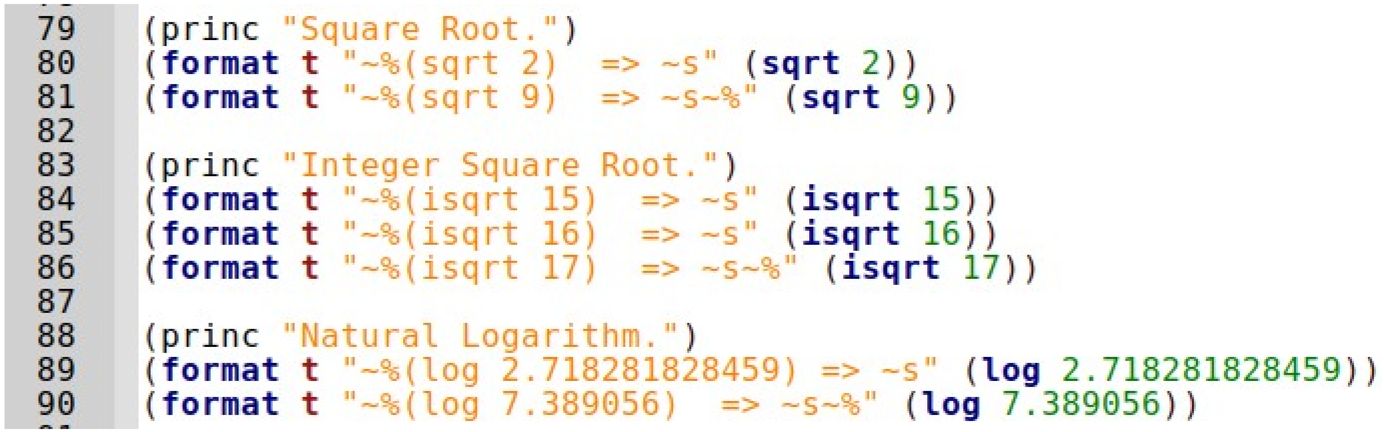
```lisp
(princ "Exponential function.")
(format t "~%(expt 2 2)  =>  ~D" (expt 2 2))
(format t "~%(expt 2 3)  =>  ~D" (expt 2 3))
(format t "~%(expt 2 4)  =>  ~D" (expt 2 4))
(format t "~%(expt 2 5)  =>  ~D~%" (expt 2 5))

(princ "Natural Exponential function.")
(format t "~%(exp 1)  =>  ~D" (exp 1))
(format t "~%(exp 2)  =>  ~D" (exp 2))
(format t "~%(exp 3)  =>  ~D" (exp 3))
(format t "~%(exp 4)  =>  ~D~%" (exp 4))

(princ "gcd - Greatest Common Divisor.")
(format t "~%(gcd 18 9 6)  =>  ~D~%" (gcd 18 9 6))

(princ "Division.")
(format t "~%(/ 6 2)  => ~s" (/ 6 2))
(format t "~%(/ 7 2)  => ~s" (/ 7 2))
(format t "~%(/ 7.0 2)  =>  ~s" (/ 7.0 2))
(format t "~%(/ 7 2.0)  =>  ~s~%" (/ 7 2.0))

(princ "Round.")
(format t "~%(round 4.5)  => ~s" (round 4.5))
(format t "~%(round 5.5)  => ~s" (round 5.5))
(format t "~%(round 6.5)  => ~s" (round 6.5))
(format t "~%(round 7.5)  => ~s~%" (round 7.5))
```



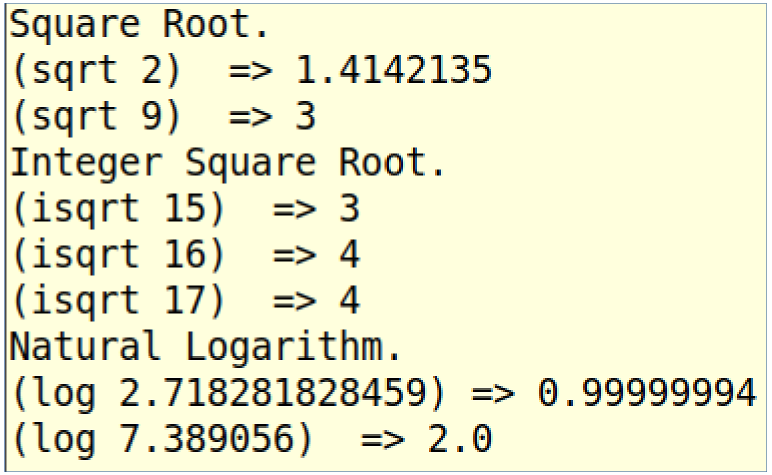
```
Exponential function.
(expt 2 2)  =>  4
(expt 2 3)  =>  8
(expt 2 4)  =>  16
(expt 2 5)  =>  32
Natural Exponential function.
(exp 1)  =>  2.7182817
(exp 2)  =>  7.389056
(exp 3)  =>  20.085537
(exp 4)  =>  54.59815
gcd - Greatest Common Divisor.
(gcd 18 9 6)  =>  3
Division.
(/ 6 2)  => 3
(/ 7 2)  => 7/2
(/ 7.0 2)  =>  3.5
(/ 7 2.0)  =>  3.5
Round.
(round 4.5)  => 4
(round 5.5)  => 6
(round 6.5)  => 6
(round 7.5)  => 8
```

File: demo_math.lisp

Math Functions



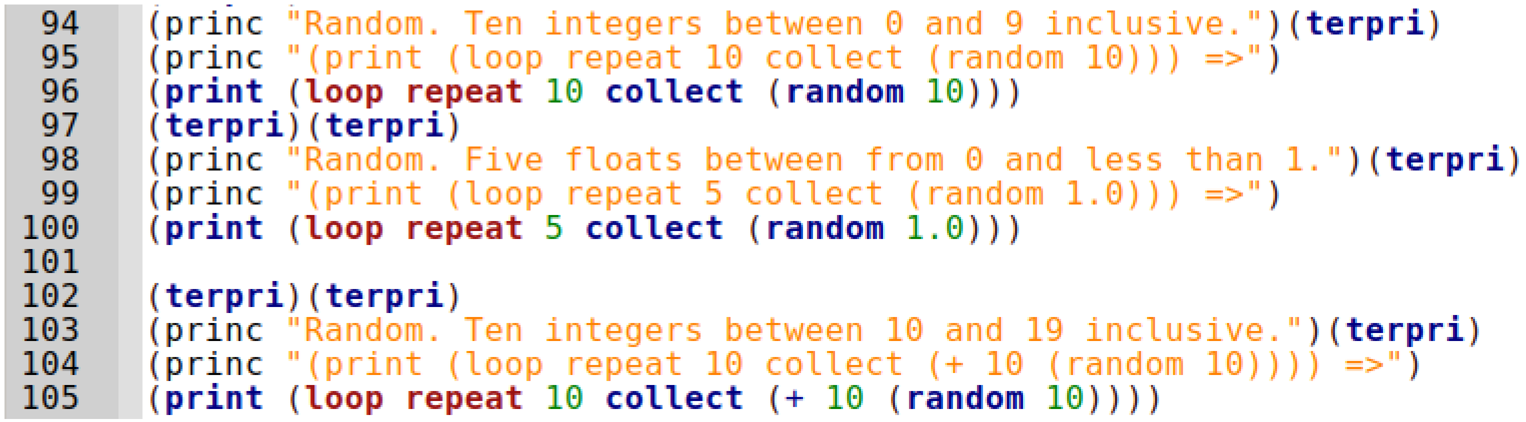
```
(princ "Square Root.")
(format t "~%(sqrt 2)  => ~s" (sqrt 2))
(format t "~%(sqrt 9)  => ~s~%" (sqrt 9))

(princ "Integer Square Root.")
(format t "~%(isqrt 15)  => ~s" (isqrt 15))
(format t "~%(isqrt 16)  => ~s" (isqrt 16))
(format t "~%(isqrt 17)  => ~s~%" (isqrt 17))

(princ "Natural Logarithm.")
(format t "~%(log 2.718281828459) => ~s" (log 2.718281828459))
(format t "~%(log 7.389056)  => ~s~%" (log 7.389056))
```



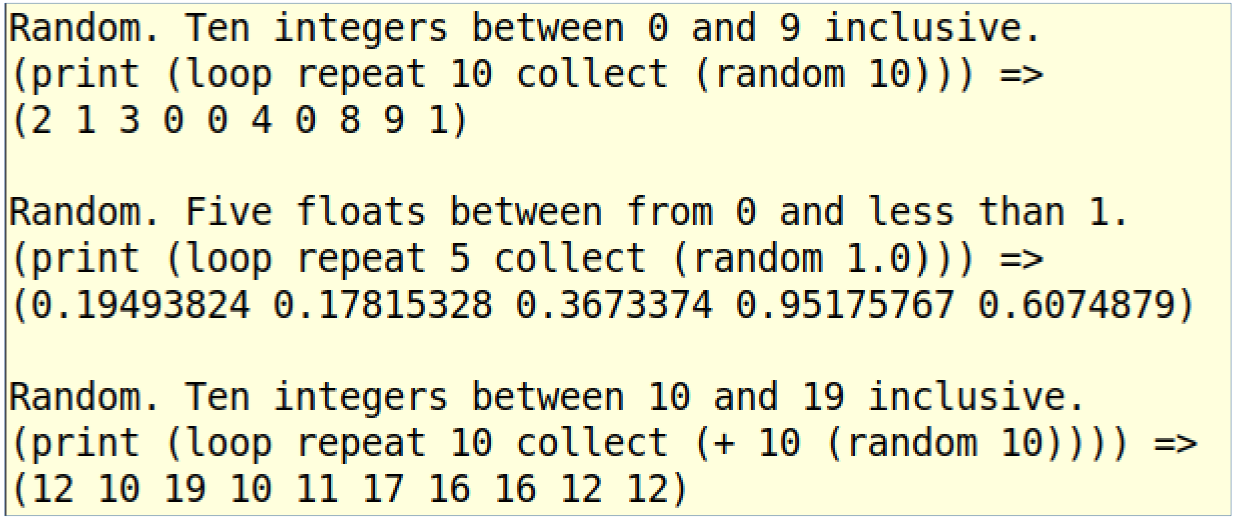
```
Square Root.
(sqrt 2)  => 1.4142135
(sqrt 9)  => 3
Integer Square Root.
(isqrt 15)  => 3
(isqrt 16)  => 4
(isqrt 17)  => 4
Natural Logarithm.
(log 2.718281828459) => 0.99999994
(log 7.389056)  => 2.0
```

File: demo_math.lisp

Math Functions



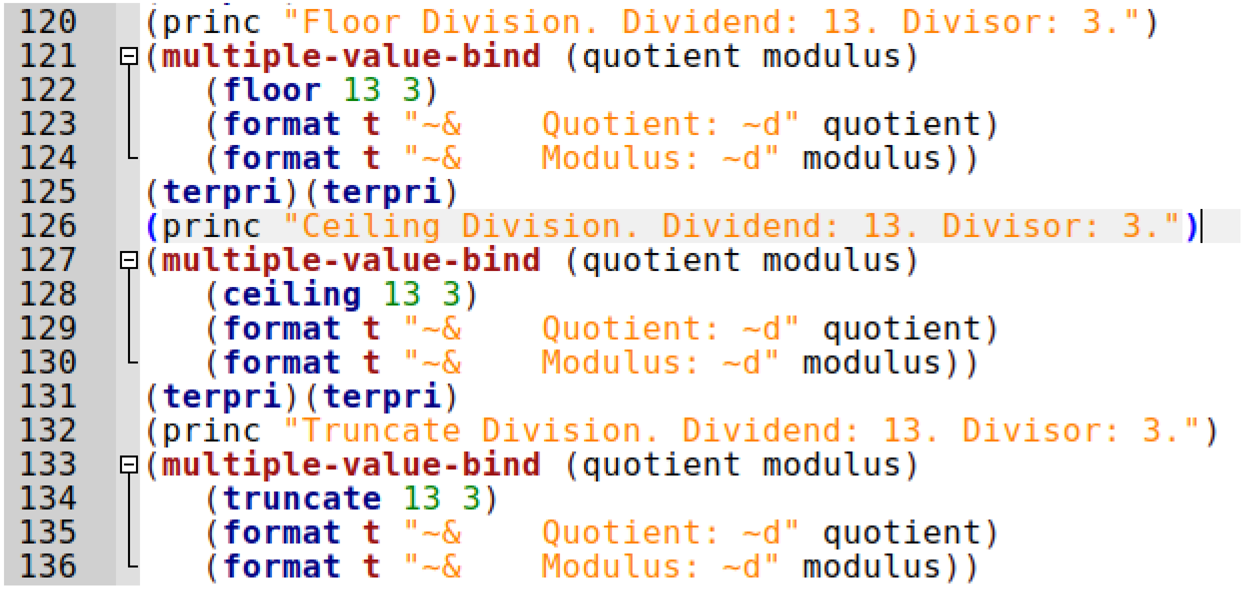
```
(princ "Random. Ten integers between 0 and 9 inclusive.")(terpri)
(princ "(print (loop repeat 10 collect (random 10))) =>")
(print (loop repeat 10 collect (random 10)))
(terpri)(terpri)
(princ "Random. Five floats between from 0 and less than 1.")(terpri)
(princ "(print (loop repeat 5 collect (random 1.0))) =>")
(print (loop repeat 5 collect (random 1.0)))

(terpri)(terpri)
(princ "Random. Ten integers between 10 and 19 inclusive.")(terpri)
(princ "(print (loop repeat 10 collect (+ 10 (random 10)))) =>")
(print (loop repeat 10 collect (+ 10 (random 10))))
```



```
Random. Ten integers between 0 and 9 inclusive.
(print (loop repeat 10 collect (random 10))) =>
(2 1 3 0 0 4 0 8 9 1)

Random. Five floats between from 0 and less than 1.
(print (loop repeat 5 collect (random 1.0))) =>
(0.19493824 0.17815328 0.3673374 0.95175767 0.6074879)

Random. Ten integers between 10 and 19 inclusive.
(print (loop repeat 10 collect (+ 10 (random 10)))) =>
(12 10 19 10 11 17 16 16 12 12)
```

File: demo_math.lisp

Math Functions



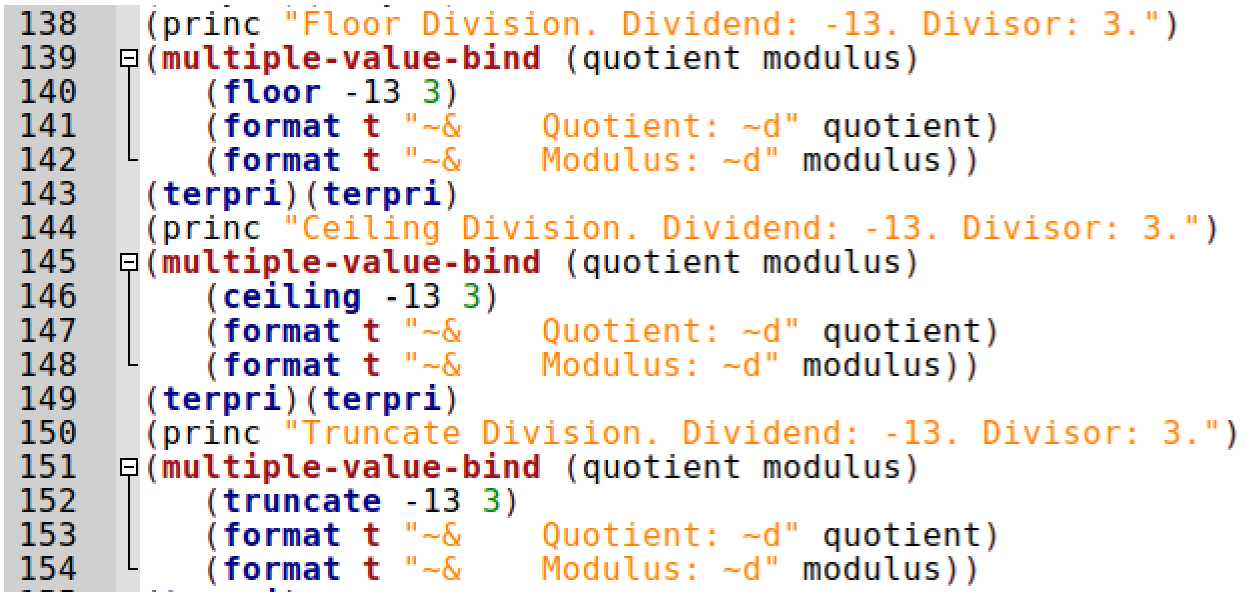
```lisp
(princ "Floor Division. Dividend: 13. Divisor: 3.")
(multiple-value-bind (quotient modulus)
   (floor 13 3)
   (format t "~&    Quotient: ~d" quotient)
   (format t "~&    Modulus: ~d" modulus))
(terpri)(terpri)
(princ "Ceiling Division. Dividend: 13. Divisor: 3.")
(multiple-value-bind (quotient modulus)
   (ceiling 13 3)
   (format t "~&    Quotient: ~d" quotient)
   (format t "~&    Modulus: ~d" modulus))
(terpri)(terpri)
(princ "Truncate Division. Dividend: 13. Divisor: 3.")
(multiple-value-bind (quotient modulus)
   (truncate 13 3)
   (format t "~&    Quotient: ~d" quotient)
   (format t "~&    Modulus: ~d" modulus))
```

```
Floor Division. Dividend: 13. Divisor: 3.
        Quotient: 4
        Modulus: 1

Ceiling Division. Dividend: 13. Divisor: 3.
        Quotient: 5
        Modulus: -2

Truncate Division. Dividend: 13. Divisor: 3.
        Quotient: 4
        Modulus: 1
```

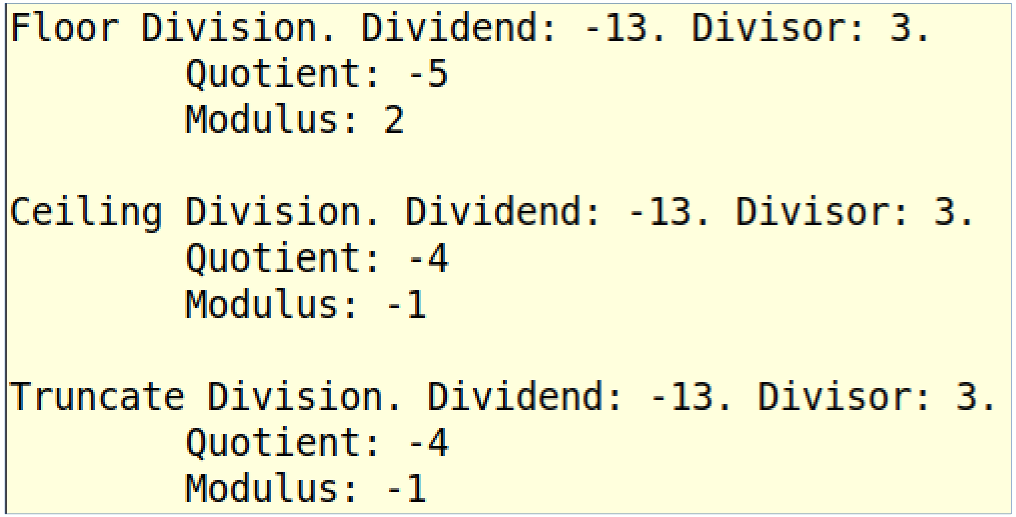
Positive: Truncate is like Floor.

File: demo_math.lisp

Math Functions



```
(princ "Floor Division. Dividend: -13. Divisor: 3.")
(multiple-value-bind (quotient modulus)
   (floor -13 3)
   (format t "~&    Quotient: ~d" quotient)
   (format t "~&    Modulus: ~d" modulus))
(terpri)(terpri)
(princ "Ceiling Division. Dividend: -13. Divisor: 3.")
(multiple-value-bind (quotient modulus)
   (ceiling -13 3)
   (format t "~&    Quotient: ~d" quotient)
   (format t "~&    Modulus: ~d" modulus))
(terpri)(terpri)
(princ "Truncate Division. Dividend: -13. Divisor: 3.")
(multiple-value-bind (quotient modulus)
   (truncate -13 3)
   (format t "~&    Quotient: ~d" quotient)
   (format t "~&    Modulus: ~d" modulus))
```



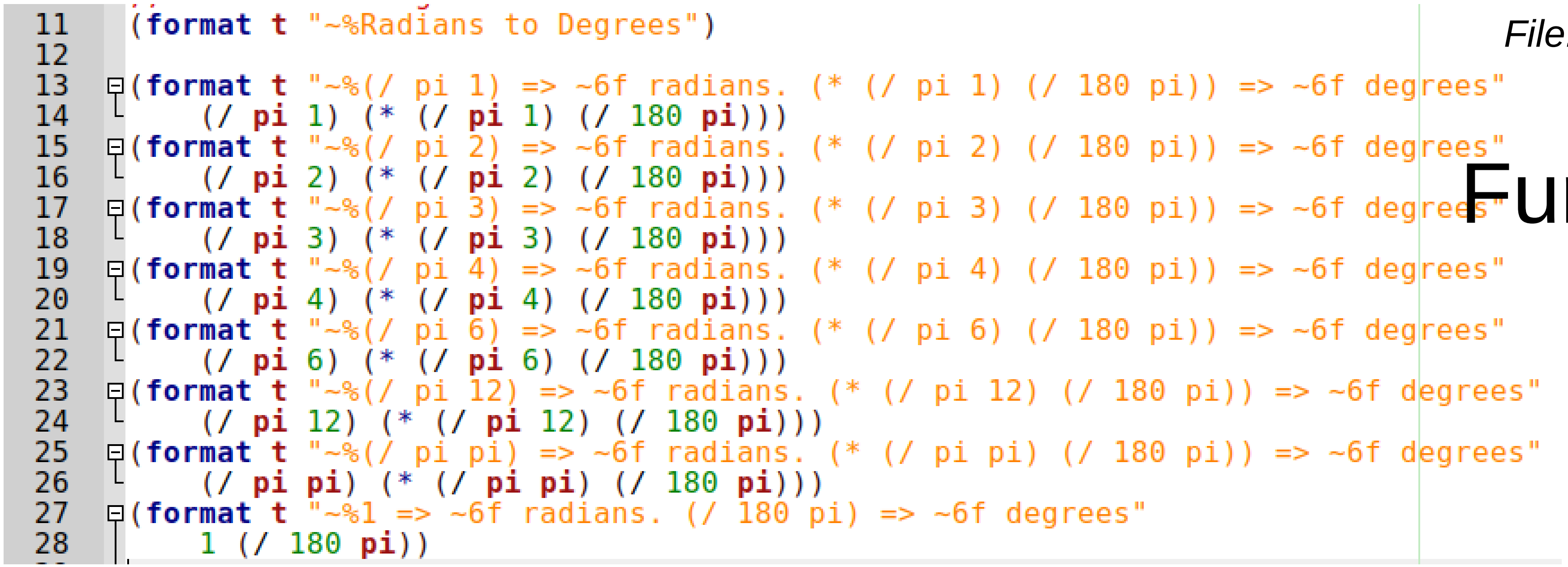
```
Floor Division. Dividend: -13. Divisor: 3.
        Quotient: -5
        Modulus: 2

Ceiling Division. Dividend: -13. Divisor: 3.
        Quotient: -4
        Modulus: -1

Truncate Division. Dividend: -13. Divisor: 3.
        Quotient: -4
        Modulus: -1
```

Negative: Truncate is like Ceiling.

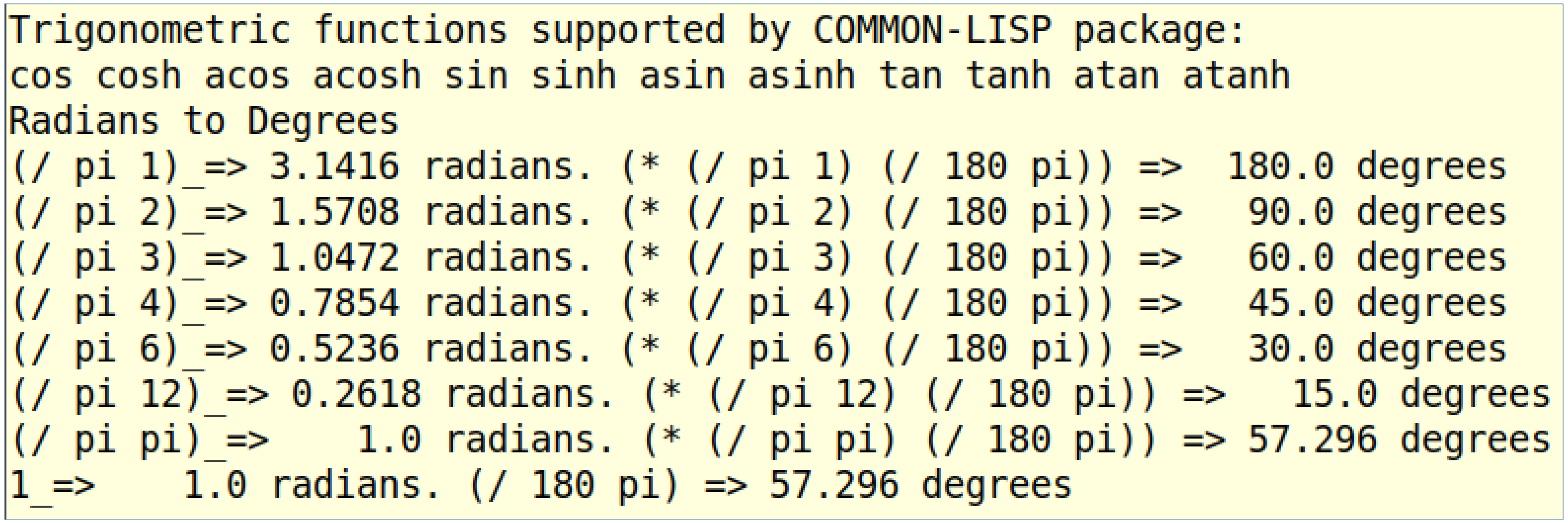
File: demo_math.lisp

File: demo_trig.lisp

Trig Functions

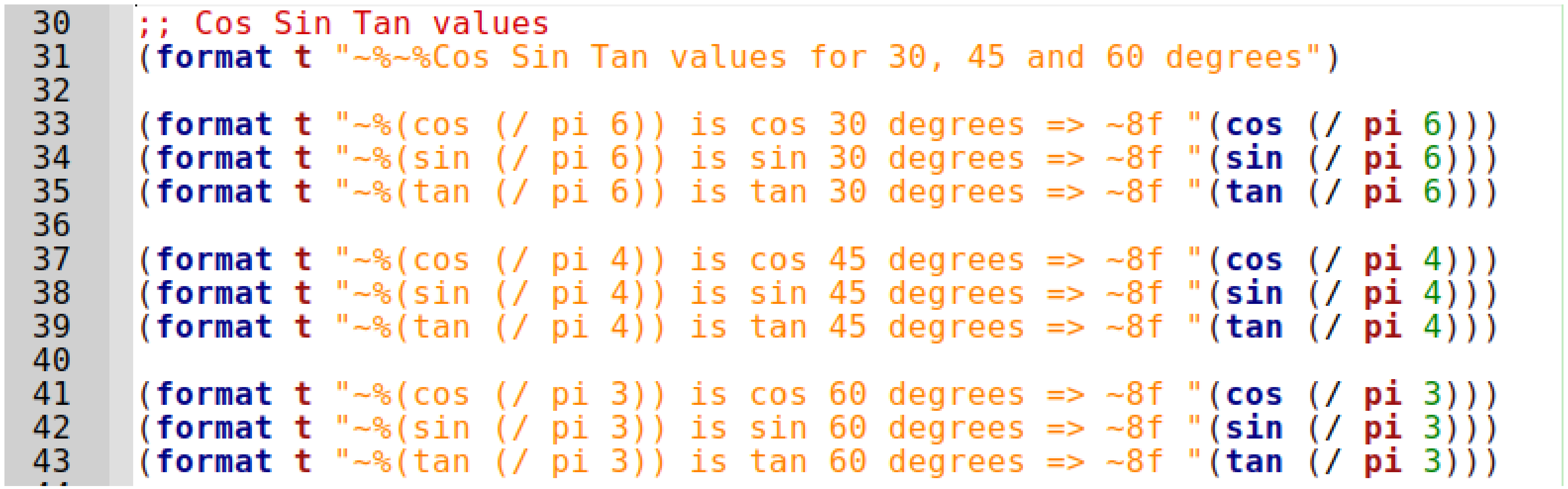


```lisp
(format t "~%Radians to Degrees")

(format t "~%(/ pi 1) => ~6f radians. (* (/ pi 1) (/ 180 pi)) => ~6f degrees"
    (/ pi 1) (* (/ pi 1) (/ 180 pi)))
(format t "~%(/ pi 2) => ~6f radians. (* (/ pi 2) (/ 180 pi)) => ~6f degrees"
    (/ pi 2) (* (/ pi 2) (/ 180 pi)))
(format t "~%(/ pi 3) => ~6f radians. (* (/ pi 3) (/ 180 pi)) => ~6f degrees"
    (/ pi 3) (* (/ pi 3) (/ 180 pi)))
(format t "~%(/ pi 4) => ~6f radians. (* (/ pi 4) (/ 180 pi)) => ~6f degrees"
    (/ pi 4) (* (/ pi 4) (/ 180 pi)))
(format t "~%(/ pi 6) => ~6f radians. (* (/ pi 6) (/ 180 pi)) => ~6f degrees"
    (/ pi 6) (* (/ pi 6) (/ 180 pi)))
(format t "~%(/ pi 12) => ~6f radians. (* (/ pi 12) (/ 180 pi)) => ~6f degrees"
    (/ pi 12) (* (/ pi 12) (/ 180 pi)))
(format t "~%(/ pi pi) => ~6f radians. (* (/ pi pi) (/ 180 pi)) => ~6f degrees"
    (/ pi pi) (* (/ pi pi) (/ 180 pi)))
(format t "~%1 => ~6f radians. (/ 180 pi) => ~6f degrees"
    1 (/ 180 pi))
```

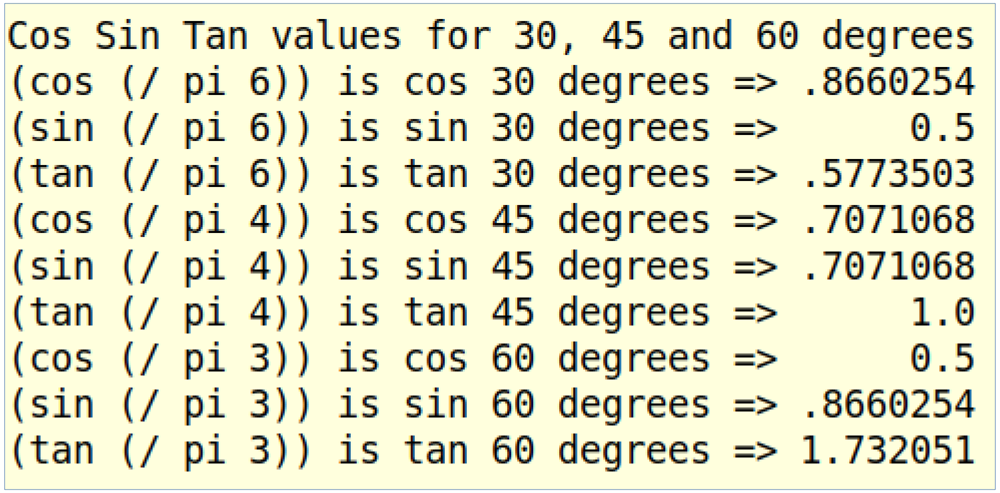


```
Trigonometric functions supported by COMMON-LISP package:
cos cosh acos acosh sin sinh asin asinh tan tanh atan atanh
Radians to Degrees
(/ pi 1)_=> 3.1416 radians. (* (/ pi 1) (/ 180 pi)) =>  180.0 degrees
(/ pi 2)_=> 1.5708 radians. (* (/ pi 2) (/ 180 pi)) =>   90.0 degrees
(/ pi 3)_=> 1.0472 radians. (* (/ pi 3) (/ 180 pi)) =>   60.0 degrees
(/ pi 4)_=> 0.7854 radians. (* (/ pi 4) (/ 180 pi)) =>   45.0 degrees
(/ pi 6)_=> 0.5236 radians. (* (/ pi 6) (/ 180 pi)) =>   30.0 degrees
(/ pi 12)_=> 0.2618 radians. (* (/ pi 12) (/ 180 pi)) =>   15.0 degrees
(/ pi pi)_=>    1.0 radians. (* (/ pi pi) (/ 180 pi)) => 57.296 degrees
1_=>    1.0 radians. (/ 180 pi) => 57.296 degrees
```

Trig Functions



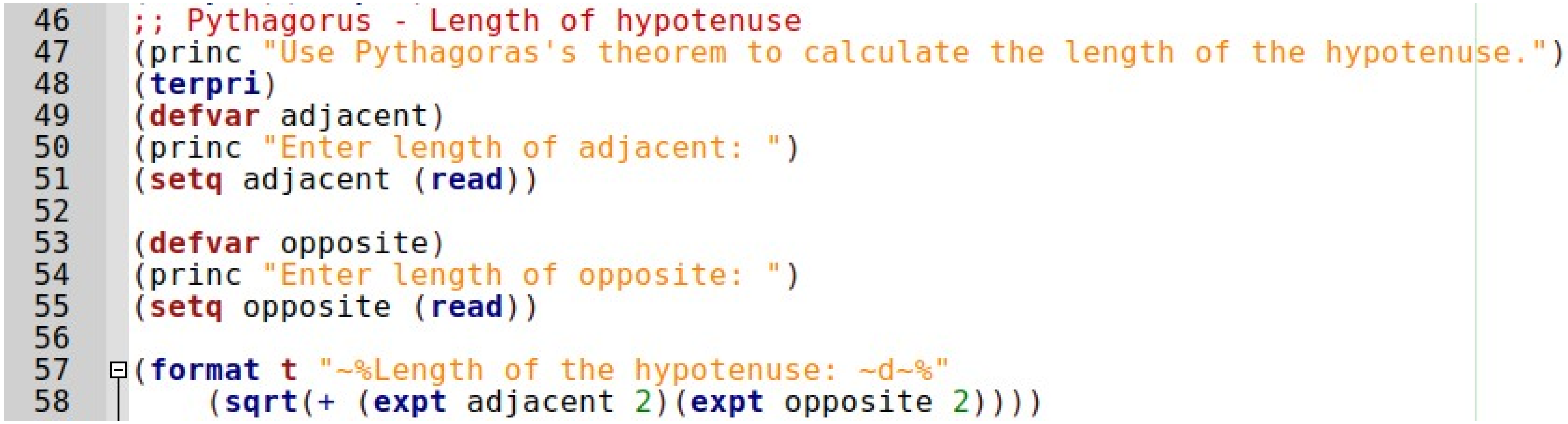
```lisp
;; Cos Sin Tan values
(format t "~%~%Cos Sin Tan values for 30, 45 and 60 degrees")

(format t "~%(cos (/ pi 6)) is cos 30 degrees => ~8f "(cos (/ pi 6)))
(format t "~%(sin (/ pi 6)) is sin 30 degrees => ~8f "(sin (/ pi 6)))
(format t "~%(tan (/ pi 6)) is tan 30 degrees => ~8f "(tan (/ pi 6)))

(format t "~%(cos (/ pi 4)) is cos 45 degrees => ~8f "(cos (/ pi 4)))
(format t "~%(sin (/ pi 4)) is sin 45 degrees => ~8f "(sin (/ pi 4)))
(format t "~%(tan (/ pi 4)) is tan 45 degrees => ~8f "(tan (/ pi 4)))

(format t "~%(cos (/ pi 3)) is cos 60 degrees => ~8f "(cos (/ pi 3)))
(format t "~%(sin (/ pi 3)) is sin 60 degrees => ~8f "(sin (/ pi 3)))
(format t "~%(tan (/ pi 3)) is tan 60 degrees => ~8f "(tan (/ pi 3)))
```



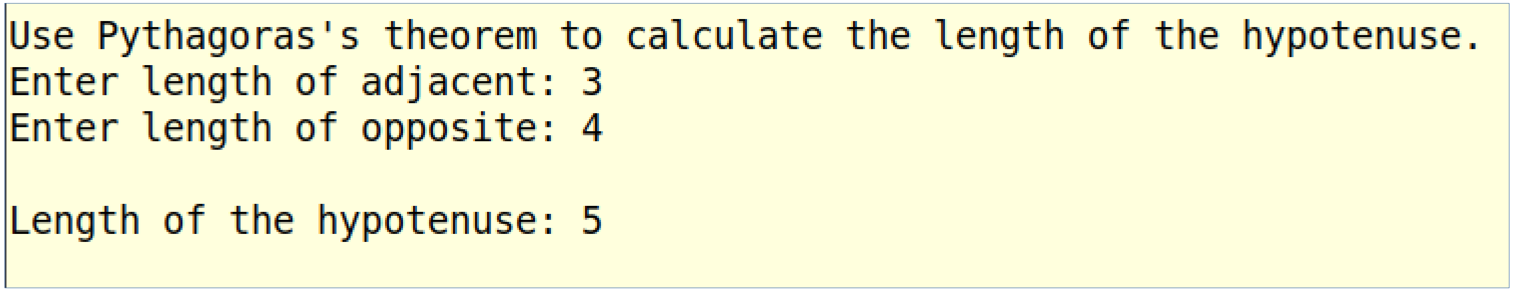
```
Cos Sin Tan values for 30, 45 and 60 degrees
(cos (/ pi 6)) is cos 30 degrees => .8660254
(sin (/ pi 6)) is sin 30 degrees =>      0.5
(tan (/ pi 6)) is tan 30 degrees => .5773503
(cos (/ pi 4)) is cos 45 degrees => .7071068
(sin (/ pi 4)) is sin 45 degrees => .7071068
(tan (/ pi 4)) is tan 45 degrees =>      1.0
(cos (/ pi 3)) is cos 60 degrees =>      0.5
(sin (/ pi 3)) is sin 60 degrees => .8660254
(tan (/ pi 3)) is tan 60 degrees => 1.732051
```

File: demo_trig.lisp

Trig Functions



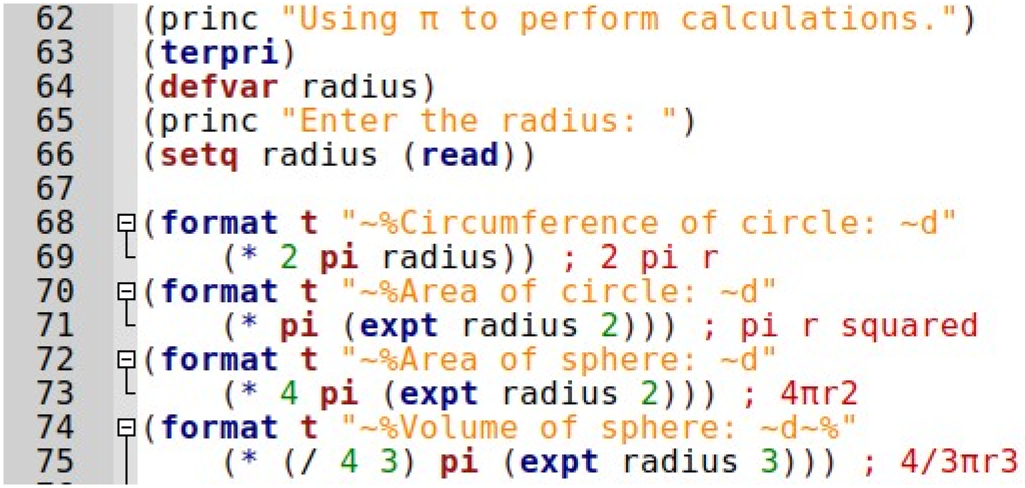
```
;; Pythagorus - Length of hypotenuse
(princ "Use Pythagoras's theorem to calculate the length of the hypotenuse.")
(terpri)
(defvar adjacent)
(princ "Enter length of adjacent: ")
(setq adjacent (read))

(defvar opposite)
(princ "Enter length of opposite: ")
(setq opposite (read))

(format t "~%Length of the hypotenuse: ~d~%"
    (sqrt(+ (expt adjacent 2)(expt opposite 2))))
```



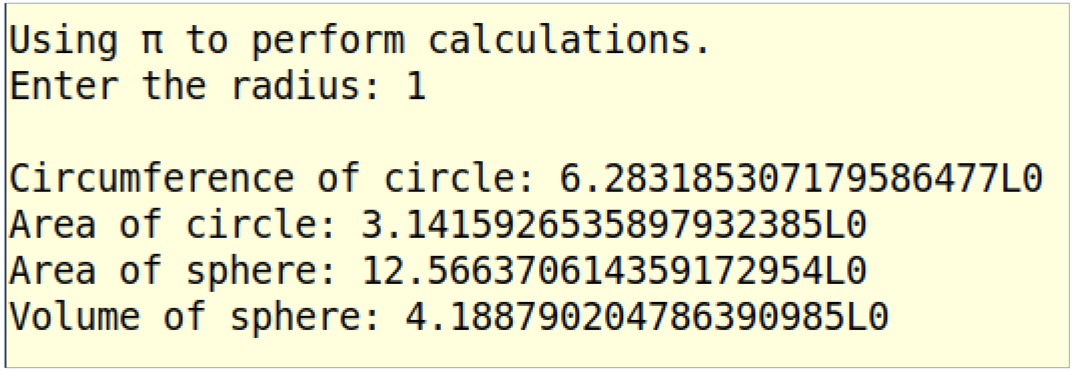
```
Use Pythagoras's theorem to calculate the length of the hypotenuse.
Enter length of adjacent: 3
Enter length of opposite: 4

Length of the hypotenuse: 5
```

File: demo_trig.lisp

Trig Functions



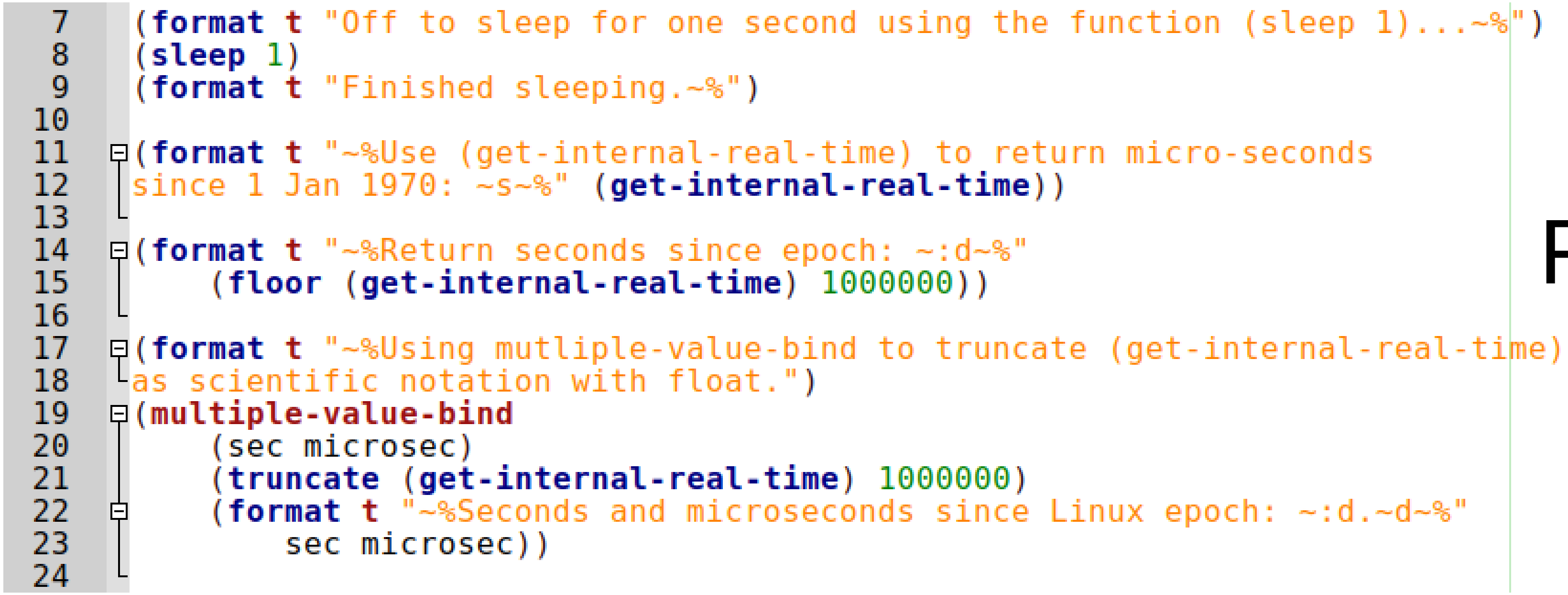
```
(princ "Using π to perform calculations.")
(terpri)
(defvar radius)
(princ "Enter the radius: ")
(setq radius (read))

(format t "~%Circumference of circle: ~d"
    (* 2 pi radius)) ; 2 pi r
(format t "~%Area of circle: ~d"
    (* pi (expt radius 2))) ; pi r squared
(format t "~%Area of sphere: ~d"
    (* 4 pi (expt radius 2))) ; 4πr2
(format t "~%Volume of sphere: ~d~%"
    (* (/ 4 3) pi (expt radius 3))) ; 4/3πr3
```



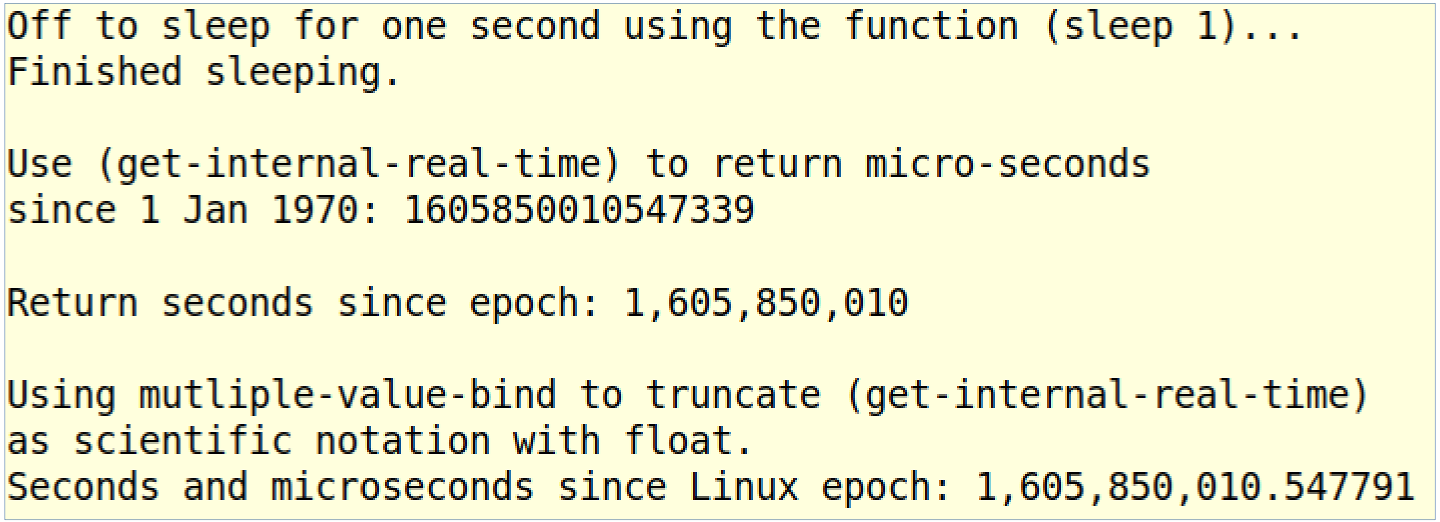
```
Using π to perform calculations.
Enter the radius: 1

Circumference of circle: 6.283185307179586477L0
Area of circle: 3.141592653589793238 5L0
Area of sphere: 12.566370614359172954L0
Volume of sphere: 4.188790204786390985L0
```

File: demo_trig.lisp

Date and Time Functions



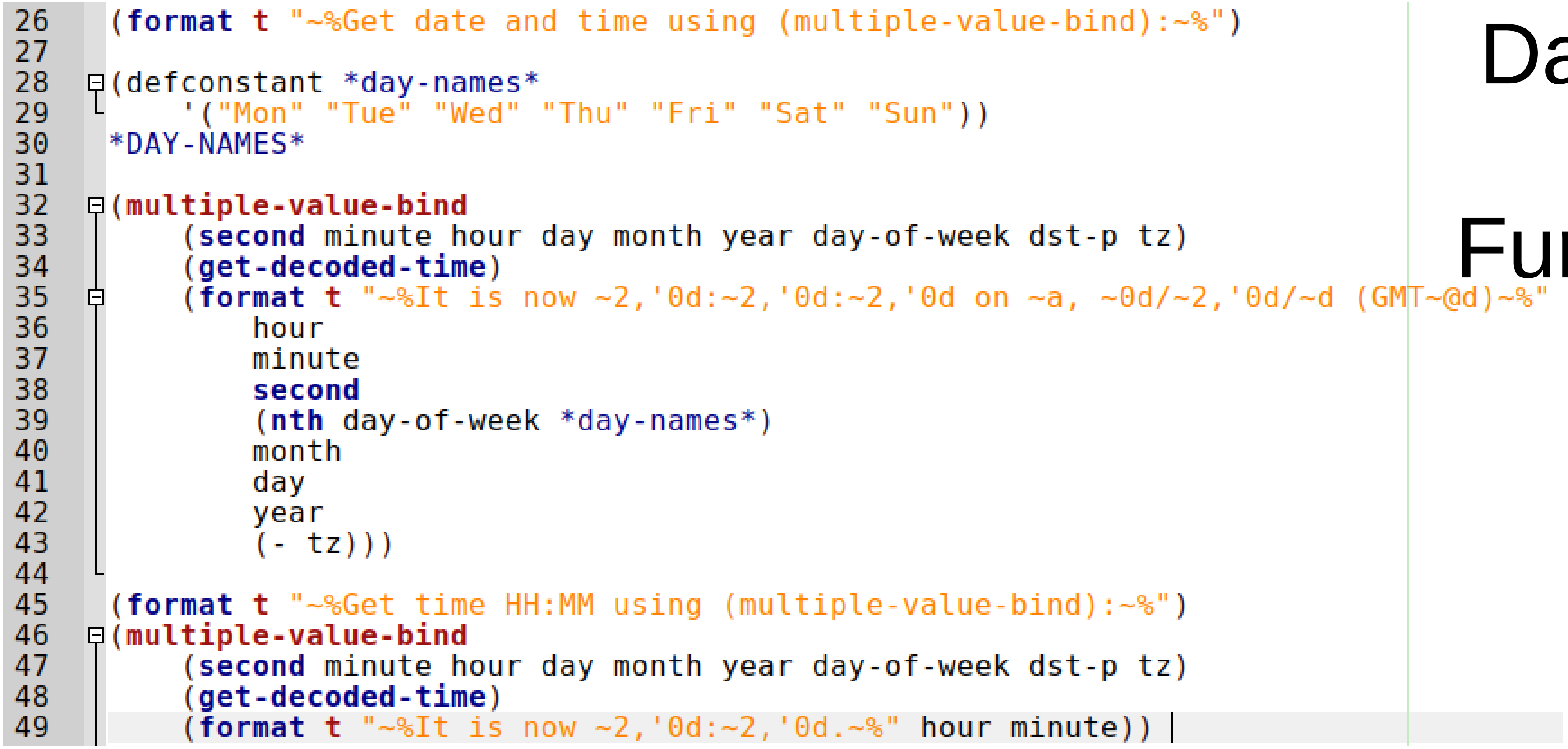
```lisp
(format t "Off to sleep for one second using the function (sleep 1)...~%")
(sleep 1)
(format t "Finished sleeping.~%")

(format t "~%Use (get-internal-real-time) to return micro-seconds
since 1 Jan 1970: ~s~%" (get-internal-real-time))

(format t "~%Return seconds since epoch: ~:d~%"
    (floor (get-internal-real-time) 1000000))

(format t "~%Using mutliple-value-bind to truncate (get-internal-real-time)
as scientific notation with float.")
(multiple-value-bind
    (sec microsec)
    (truncate (get-internal-real-time) 1000000)
    (format t "~%Seconds and microseconds since Linux epoch: ~:d.~d~%"
        sec microsec))
```



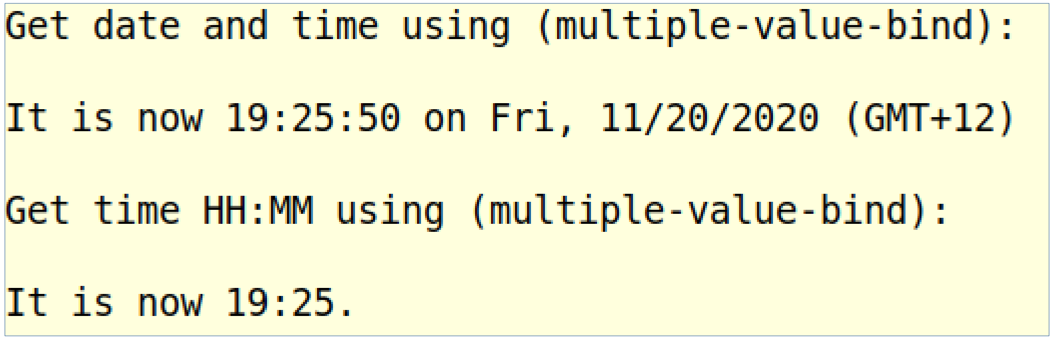
```
Off to sleep for one second using the function (sleep 1)...
Finished sleeping.

Use (get-internal-real-time) to return micro-seconds
since 1 Jan 1970: 1605850010547339

Return seconds since epoch: 1,605,850,010

Using mutliple-value-bind to truncate (get-internal-real-time)
as scientific notation with float.
Seconds and microseconds since Linux epoch: 1,605,850,010.547791
```

File: demo_date.lisp

Date and Time Functions



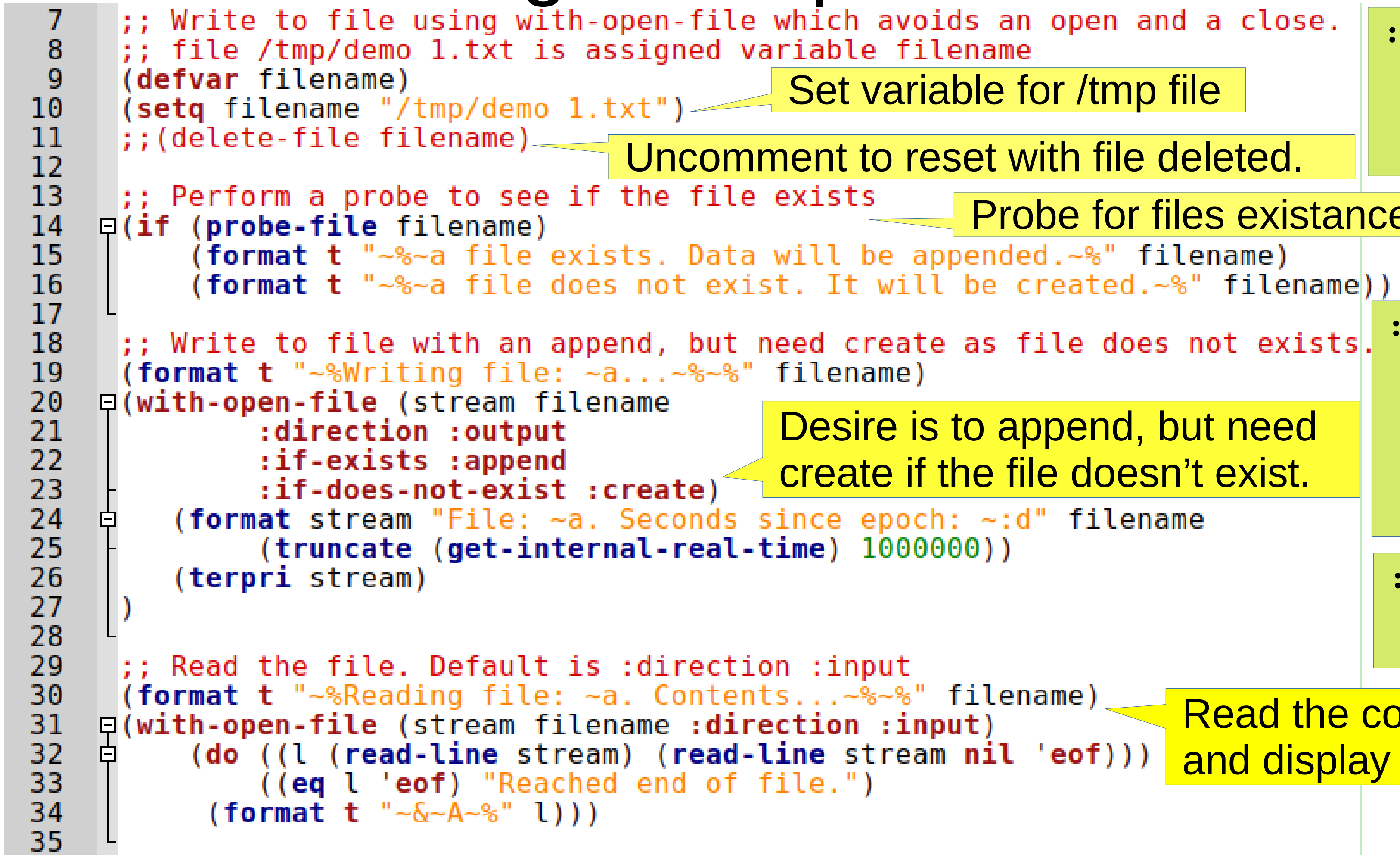
```lisp
(format t "~%Get date and time using (multiple-value-bind):~%")

(defconstant *day-names*
    '("Mon" "Tue" "Wed" "Thu" "Fri" "Sat" "Sun"))
*DAY-NAMES*

(multiple-value-bind
    (second minute hour day month year day-of-week dst-p tz)
    (get-decoded-time)
    (format t "~%It is now ~2,'0d:~2,'0d:~2,'0d on ~a, ~0d/~2,'0d/~d (GMT~@d)~%"
        hour
        minute
        second
        (nth day-of-week *day-names*)
        month
        day
        year
        (- tz)))

(format t "~%Get time HH:MM using (multiple-value-bind):~%")
(multiple-value-bind
    (second minute hour day month year day-of-week dst-p tz)
    (get-decoded-time)
    (format t "~%It is now ~2,'0d:~2,'0d.~%" hour minute))
```



```
Get date and time using (multiple-value-bind):

It is now 19:25:50 on Fri, 11/20/2020 (GMT+12)

Get time HH:MM using (multiple-value-bind):

It is now 19:25.
```

File: demo_date.lisp

File I/O. Using with-open-file.



```lisp
;; Write to file using with-open-file which avoids an open and a close.
;; file /tmp/demo 1.txt is assigned variable filename
(defvar filename)
(setq filename "/tmp/demo 1.txt")
;;(delete-file filename)

;; Perform a probe to see if the file exists
(if (probe-file filename)
    (format t "~%~a file exists. Data will be appended.~%" filename)
    (format t "~%~a file does not exist. It will be created.~%" filename))

;; Write to file with an append, but need create as file does not exists.
(format t "~%Writing file: ~a...~%~%" filename)
(with-open-file (stream filename
        :direction :output
        :if-exists :append
        :if-does-not-exist :create)
   (format stream "File: ~a. Seconds since epoch: ~:d" filename
        (truncate (get-internal-real-time) 1000000))
   (terpri stream)
)

;; Read the file. Default is :direction :input
(format t "~%Reading file: ~a. Contents...~%~%" filename)
(with-open-file (stream filename :direction :input)
    (do ((l (read-line stream) (read-line stream nil 'eof)))
        ((eq l 'eof) "Reached end of file.")
     (format t "~&~A~%" l)))
```

Set variable for /tmp file

Uncomment to reset with file deleted.

Probe for files existance. Just curious.

Desire is to append, but need create if the file doesn't exist.

Read the contents of the file and display on the console.

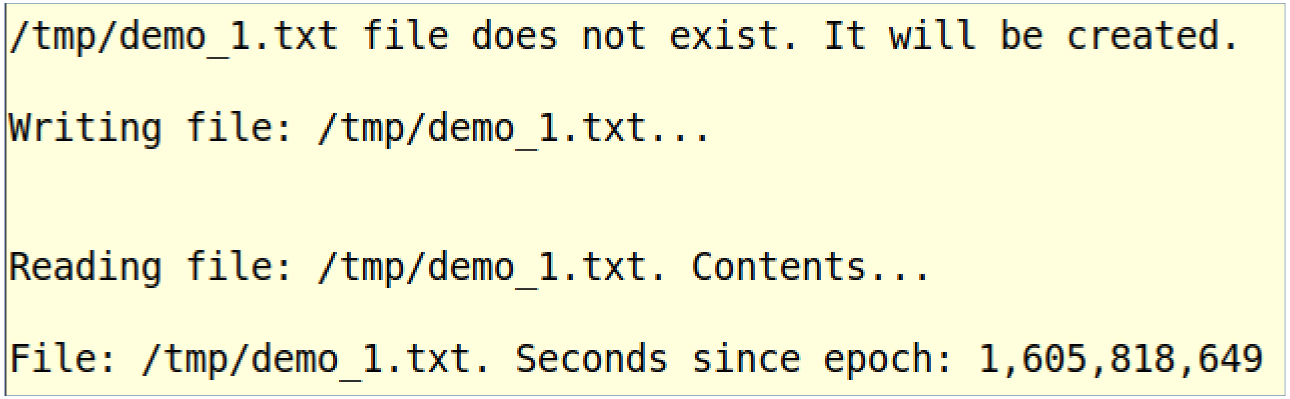
```
:direction
    :input
    :output
    :io
    :probe
```

```
:if exists
    :error
    :new-version
    :rename
    :rename-and-delete
    :append
    :supersede
```

```
:if-does-not-exist
    :error
    :create
```

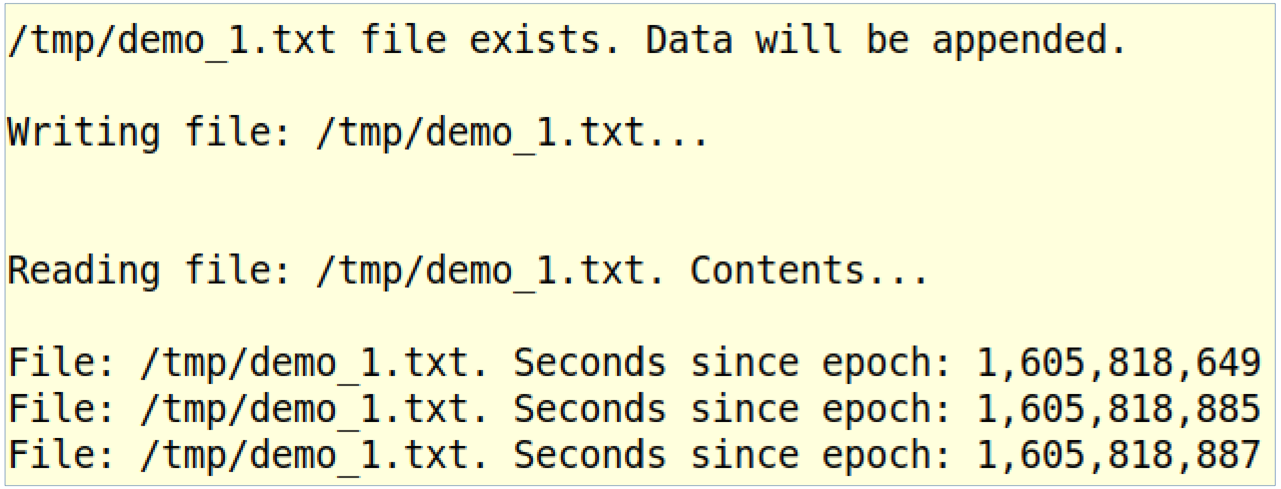
File: demo_file.lisp

File I/O. Using with-open-file.



```
/tmp/demo_1.txt file does not exist. It will be created.

Writing file: /tmp/demo_1.txt...


Reading file: /tmp/demo_1.txt. Contents...

File: /tmp/demo_1.txt. Seconds since epoch: 1,605,818,649
```

First time. File is created.



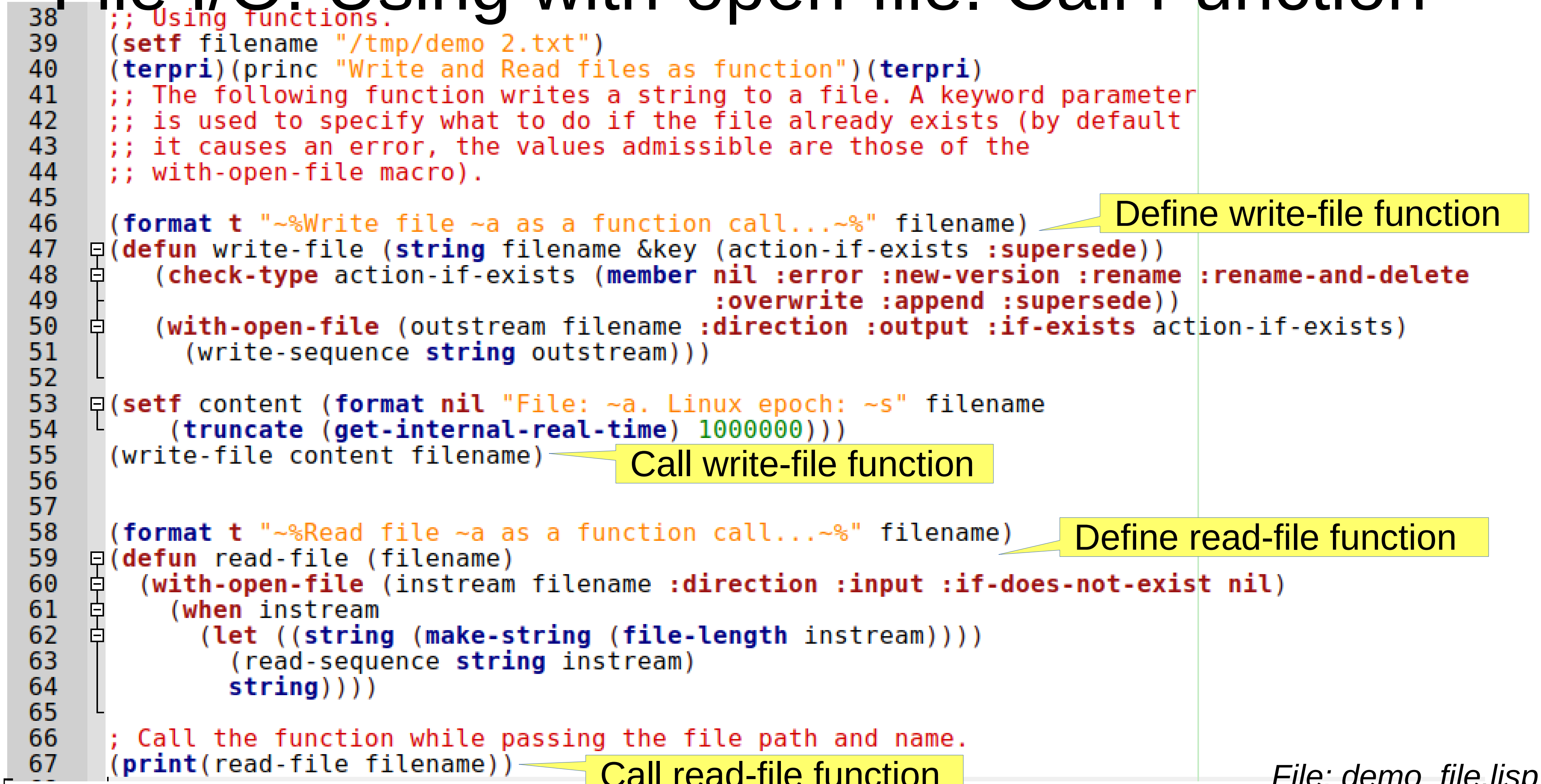
```
/tmp/demo_1.txt file exists. Data will be appended.

Writing file: /tmp/demo_1.txt...


Reading file: /tmp/demo_1.txt. Contents...

File: /tmp/demo_1.txt. Seconds since epoch: 1,605,818,649
File: /tmp/demo_1.txt. Seconds since epoch: 1,605,818,885
File: /tmp/demo_1.txt. Seconds since epoch: 1,605,818,887
```

Third time. File is appended.

Three lines of data. One for each time the program was run.

File: demo_file.lisp

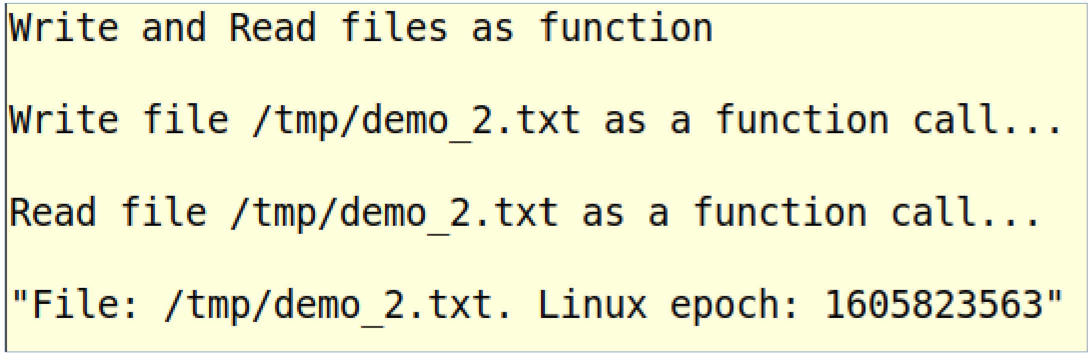
File I/O. Using with-open-file. Call Function



```lisp
;; Using functions.
(setf filename "/tmp/demo 2.txt")
(terpri)(princ "Write and Read files as function")(terpri)
;; The following function writes a string to a file. A keyword parameter
;; is used to specify what to do if the file already exists (by default
;; it causes an error, the values admissible are those of the
;; with-open-file macro).

(format t "~%Write file ~a as a function call...~%" filename)
(defun write-file (string filename &key (action-if-exists :supersede))
   (check-type action-if-exists (member nil :error :new-version :rename :rename-and-delete
                                        :overwrite :append :supersede))
   (with-open-file (outstream filename :direction :output :if-exists action-if-exists)
     (write-sequence string outstream)))

(setf content (format nil "File: ~a. Linux epoch: ~s" filename
    (truncate (get-internal-real-time) 1000000)))
(write-file content filename)


(format t "~%Read file ~a as a function call...~%" filename)
(defun read-file (filename)
  (with-open-file (instream filename :direction :input :if-does-not-exist nil)
    (when instream
      (let ((string (make-string (file-length instream))))
        (read-sequence string instream)
        string))))

; Call the function while passing the file path and name.
(print(read-file filename))
```

File: demo_file.lisp

File I/O. Using with-open-file. Call Function



```
Write and Read files as function

Write file /tmp/demo_2.txt as a function call...

Read file /tmp/demo_2.txt as a function call...

"File: /tmp/demo_2.txt. Linux epoch: 1605823563"
```

File is superseded.

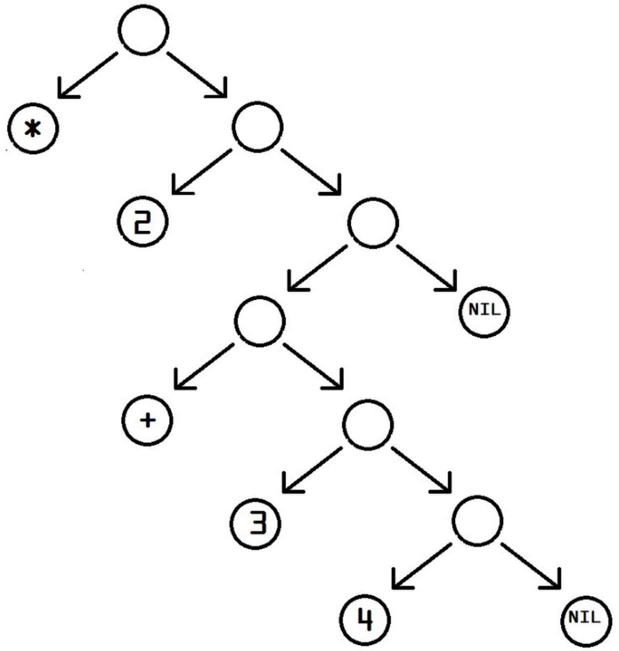
Content of file.

File: demo_file.lisp

Lisp: Any Questions? Run Demo Code?

Not Covered:

- String manipulation – Only “format” function. See https://lispcookbook.github.io/cl-cookbook/strings.html
- Packages. All 15000 of them!
- The LISP kernel addon for Jupyter-Notebook.
- Compiling executable code.
- Common lisp is an interpreted language ?
- "dumping an image" in lisp
- Setf is very important in lisp
- asdf (more or less the lisp equivalent to gnu make)
- Lambdas and closures
- The common lisp object system.
- Macroexpand and disassemble



Tree data structure representing the S-expression (* 2 (+ 3 4))

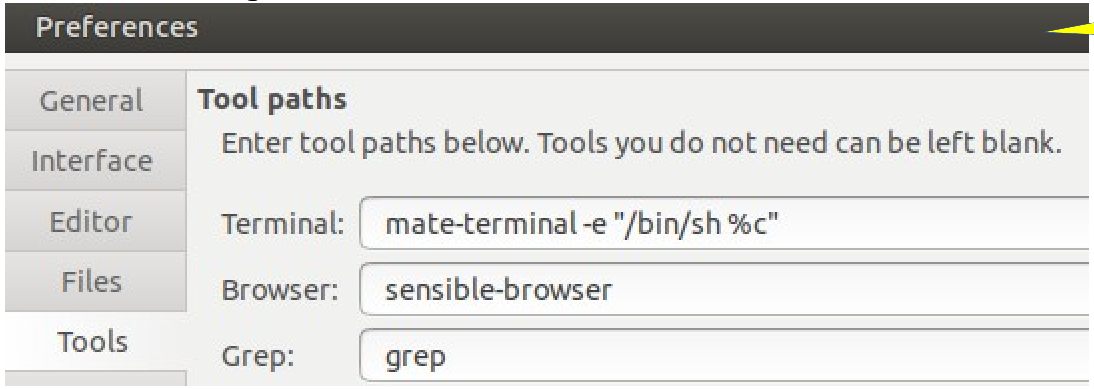
Geany IDE Modificatons. See Appendix 1. Next two slides
Apropos function. See Appendix 2.

Appendix 1: Geany IDE Modifications.

- Supports LISP
- Install from debian repositories: `$ sudo apt install geany`
- Default console is: uxterm. Change to Mate-terminal.
  Edit -> Preferences -> Tools -> Terminal: `mate-terminal -e "/bin/sh %c"`
- Select whether to launch SBCL or CLISP.
  Build -> Set Build Commands -> Execute Commands:
  CLISP: `clisp "%f"`
  SBCL: `sbcl --script "%f"`
- “Underscore” bug. https://www.geany.org/documentation/faq/
  Fixed in Geany 1.37. For 1.36 and below...
  Tools -> Configuration Files -> filetypes.common -> then uncomment:

```
[styling]
line_height=0;2;
```

Geany IDE Modifications



Preferences

General | Interface | Editor | Files | Tools

Tool paths

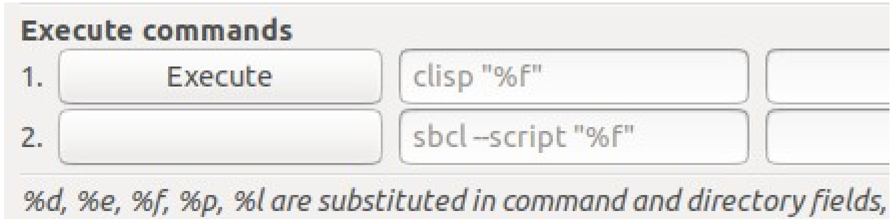
Enter tool paths below. Tools you do not need can be left blank.

Terminal: mate-terminal -e "/bin/sh %c"

Browser: sensible-browser

Grep: grep

Edit -> Preferences -> Tools ->

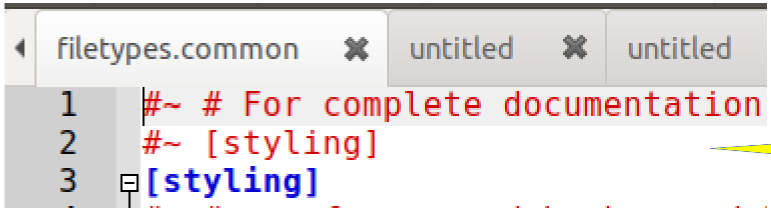


Execute commands

1. Execute | clisp "%f"
2. | sbcl --script "%f"

%d, %e, %f, %p, %l are substituted in command and directory fields,

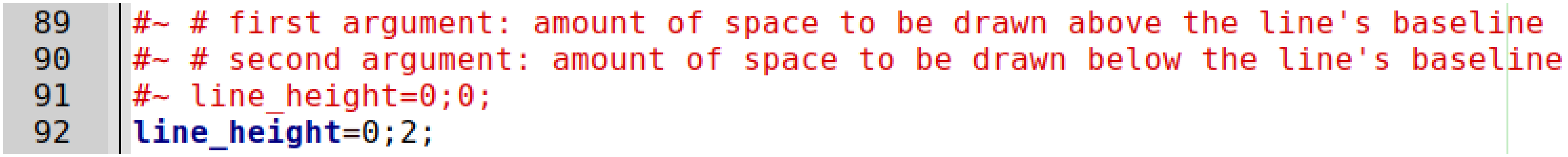
Build -> Set Build Commands ->



filetypes.common | untitled | untitled

```
#~ # For complete documentation
#~ [styling]
[styling]
```

Tools -> Configuration Files -> filetypes.common ->



```
#~ # first argument: amount of space to be drawn above the line's baseline
#~ # second argument: amount of space to be drawn below the line's baseline
#~ line_height=0;0;
line_height=0;2;
```

Appendix 2: apropos and apropos-list functions

- Find your way around all the functions.
- https://dept-info.labri.fr/~strandh/Teaching/MTP/Common/David-Lamkins/chapter10.html
- http://www.lispworks.com/documentation/HyperSpec/Body/f_apropo.htm
- Examples:

```
* (apropos "COS" :common-lisp)
ACOS (fbound)
ACOSH (fbound)
COS (fbound)
COSH (fbound)
```

In common-lisp package find the trigonometric functions related to COS

```
* (apropos "ACOSH")
SB-C::ACOSH-DERIVE-TYPE-OPTIMIZER (fbound)
SB-KERNEL::COMPLEX-ACOSH (fbound)
SB-KERNEL:%ACOSH (fbound)
ACOSH (fbound)
```

In any package find functions related to ACOSH

```
* (apropos-list "COS" :common-lisp)
(ACOS ACOSH COS COSH)
```

Return a list from common-lisp package of any functions related to COS