

LISP

INDIVIDUAL - PROJECT

*Submitted in partial fulfilment of the requirement for the
award of the degree
of*

MASTER'S IN DATA SCIENCE

In course

ARTIFICIAL INTELLIGENCE (CSCI 755 M02 Fall 2020)

by

MEHREET SINGH BAJAJ (1274698)



**New York Institute
of Technology**

TASK

You will turn in a paper with the following:

1. Cover page with your name, id, class, and semester
(CSCI-755-M02 Fall2020)
2. Each of the two exercises
 1. Snapshots of your input
 2. Snapshots of the output
 3. Any other supporting comments

First, you are to familiarize yourself with the basic common LISP environment and write some simple functions.

- Download a common LISP from the internet.
 - Install the common LISP that you downloaded
 - **DO NOT PAY FOR THIS**
-
1. Start it and play around with it
 - Type in some symbols and numbers (' a, 5, 3.183, nil, T, "hello", etc.) and see what results you get.
 - Call some basic functions (+ 5 3), (setf a 6), (setf b 12), (* a b), (set c(/ a b)), etc.
 - Write some simple functions like square, cube, etc.
 - Write a few list functions like determining if all elements of a list are numbers, characters, etc.
 2. Start a dribble session. Your first command should be (print "name") with your name in quotes.

❖ INSTALLING LISP:

Mac Ports

Download

Home Installing MacPorts Available Ports Documentation Support & Development Contact Us News

Available Downloads

The MacPorts Project Official Homepage

MacPorts FAQ

The MacPorts Project is an open-source community initiative to design an easy-to-use system for compiling, installing, and upgrading either command-line, X11 or Aqua based open-source software on the [Mac operating system](#). To that end we provide the command-line driven MacPorts software package under a [3-Clause BSD License](#), and through it easy access to thousands of ports that [greatly simplify](#) the task of [compiling and installing](#) open-source software on your Mac.

Report a Bug

Bug reporting Guidelines

Git Repository

We provide a single software tree that attempts to track the latest release of every software title (port) we distribute, without splitting them into "stable" Vs. "unstable" branches, targeting mainly macOS High Sierra v10.13 and later (including macOS Big Sur v11). There are [thousands of ports](#) in our tree, distributed among different categories, and more are being added on a regular basis.

MacPorts Team

Becoming a Member

Getting started

For information on installing MacPorts please see the [installation](#) section of this site and explore the myriad of download options we provide and our base system requirements.

If you run into any problems installing and/or using MacPorts we also have many options to help you, depending on how you wish to get [get in touch with us](#). Other important help resources are our online documentation, A.K.A [The MacPorts Guide](#), and our Trac [Wiki server & bug tracker](#).

Latest MacPorts [release](#): 2.6.4

Getting involved: Students

A good way for students to get involved is through the [Google Summer of Code](#). GSoC is a program to encourage students' participation in Open Source development and offers a stipend to work on the project with an organization for three months. MacPorts has been participating



Mac Ports

Download

Home **Installing MacPorts** Available Ports Documentation Support & Development Contact Us News

Available Downloads

Quickstart

MacPorts FAQ

1. Install [Xcode and the Xcode Command Line Tools](#)

Report a Bug

2. Agree to Xcode license in Terminal: `sudo xcodebuild -license`

Bug reporting Guidelines

3. Install MacPorts for your version of the Mac operating system:

◦ [macOS Big Sur v11](#)

◦ [macOS Catalina v10.15](#)

◦ [macOS Mojave v10.14](#)

◦ [macOS High Sierra v10.13](#)

◦ [Older OS? See here.](#)

Installing MacPorts

MacPorts version 2.6.4 is available in various formats for download and installation (note, if you are upgrading to a new major release of macOS, see the [migration info page](#)):

- "pkg" installers for [Big Sur](#), [Catalina](#), [Mojave](#), and [High Sierra](#), for use with the macOS Installer. This is the simplest installation procedure that most users should [follow](#) after meeting the requirements listed [below](#). Installers for legacy platforms [Sierra](#), [El Capitan](#), [Yosemite](#), [Mavericks](#), [Mountain Lion](#), [Lion](#), [Snow Leopard](#), [Leopard](#) and [Tiger](#) are also available.
- In [source form](#) as either a [tar.bz2](#) package or a [tar.gz](#) one for manual compilation, if you intend to customize your installation in any way.
- [Git clone](#) of the unpackaged sources, if you wish to follow MacPorts development.
- The [selfupdate](#) target of the `port(1)` command, for users who already have MacPorts installed and wish to upgrade to a newer release.

Checksums for our packaged [downloads](#) are contained in the corresponding [checksums file](#).

Install MacPorts

Welcome to the MacPorts Installer

Introduction

- Read Me
- License
- Destination Select
- Installation Type
- Installation
- Summary

Welcome to the MacPorts for macOS Installer
MacPorts provides the infrastructure that allows easy installation and management of freely available software on Mac OS X 10.15 systems.

<https://www.macports.org>

This installer guides you through the steps necessary to install MacPorts 2.6.4 for macOS. To get started, click Continue.

This package was made with:

Mac Ports

<http://www.macports.org/>

Go Back Continue

Install MacPorts

Select a Destination

Destination Select

- Introduction
- Read Me
- License
- Destination Select
- Installation Type
- Installation
- Summary

Select the disk where you want to install the MacPorts software.

 @NGYY
212.92 GB available
451 GB total

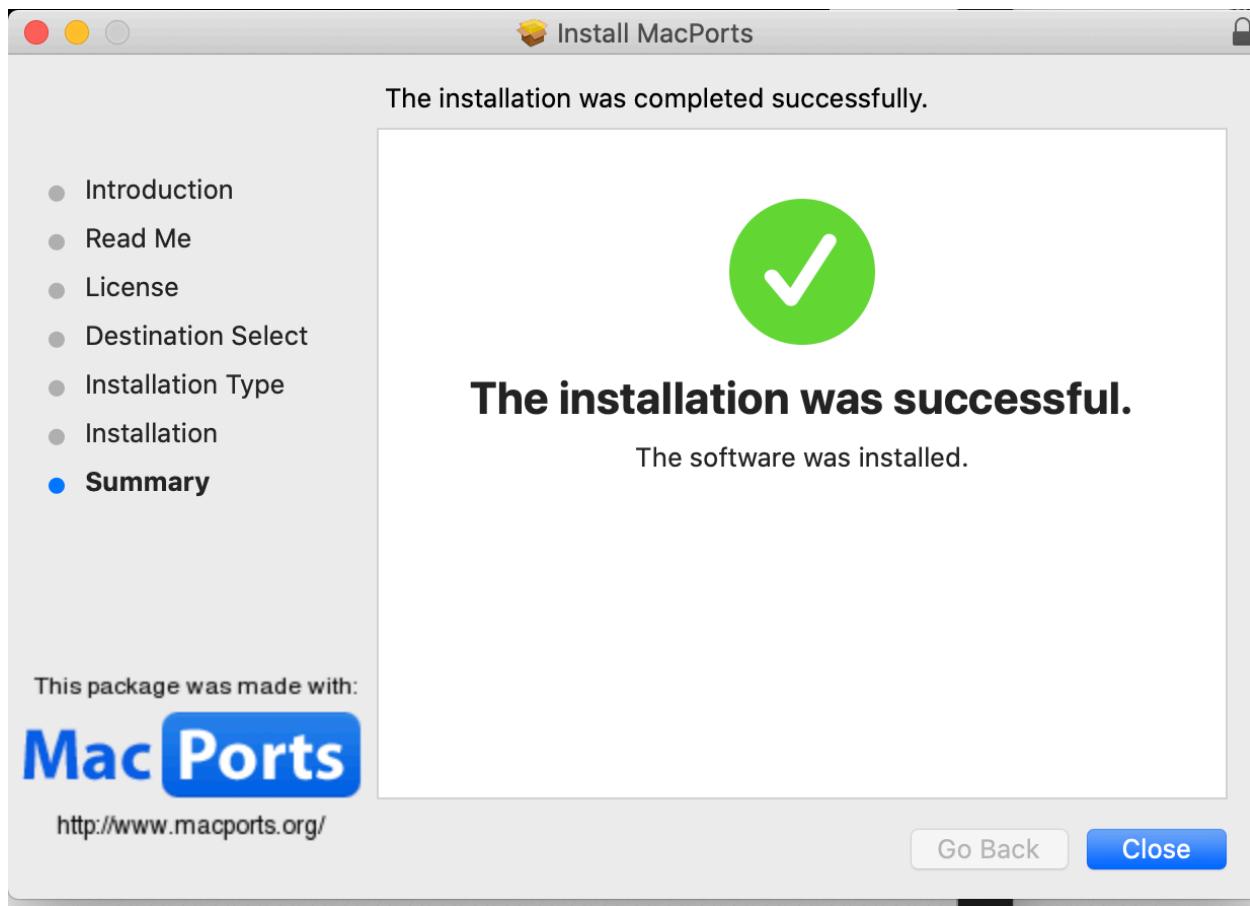
Installing this software requires 14.5 MB of space.
You have chosen to install this software on the disk "@NGYY".

This package was made with:

Mac Ports

<http://www.macports.org/>

Go Back Continue



The screenshot shows a terminal window titled "mehreetsinghbajaj — tclsh8.5 □ sudo — 80x24". The command "sudo port install clisp" is highlighted with a red oval. The terminal output shows the user entering their password three times, followed by dependency calculations and a list of packages to be installed: gettext, libiconv, libsigsegv, ncurses, readline. The final prompt is "Continue? [Y/n]".

```
mehreetsinghbajaj — tclsh8.5 □ sudo — 80x24
[mehreetsinghbajaj@MacBook-Pro ~ % sudo port install clisp
>Password:
Sorry, try again.
>Password:
Sorry, try again.
>Password:
---> Computing dependencies for clisp
The following dependencies will be installed:
gettext
libiconv
libsigsegv
ncurses
readline
Continue? [Y/n] :
```

```
mehreetsinghbajaj — -zsh — 80x24
---> Attempting to fetch clisp-2.49.92_0.darwin_19.x86_64.tbz2 from https://lil
.fr.packages.macports.org/clisp
---> Attempting to fetch clisp-2.49.92_0.darwin_19.x86_64.tbz2 from https://pac
kages.macports.org/clisp
---> Attempting to fetch clisp-2.49.92_0.darwin_19.x86_64.tbz2 from https://cph
.dk.packages.macports.org/clisp
---> Fetching distfiles for clisp
---> Attempting to fetch clisp-2.49.92.tar.bz2 from https://lil.fr.distfiles.ma
cports.org/clisp
---> Verifying checksums for clisp
---> Extracting clisp
---> Configuring clisp
---> Building clisp

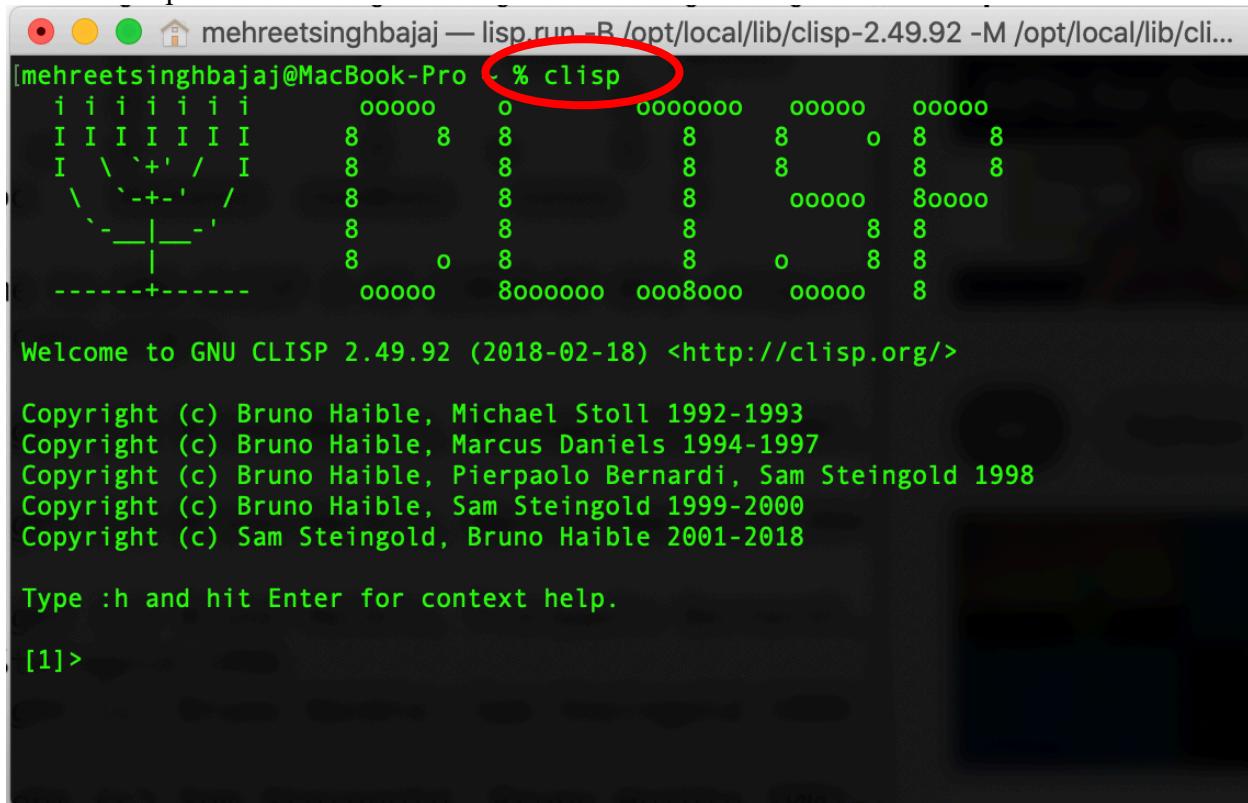
---> Staging clisp into destroot
---> Installing clisp @2.49.92_0
---> Activating clisp @2.49.92_0
---> Cleaning clisp
---> Updating database of binaries
---> Scanning binaries for linking errors
---> No broken files found.
---> No broken ports found.
mehreetsinghbajaj@MacBook-Pro ~ %
mehreetsinghbajaj@MacBook-Pro ~ %
```

Let's start working in our bash shell

```
mehreetsinghbajaj — -zsh — 80x24
[mehreetsinghbajaj@MacBook-Pro ~ % cat /etc/shells
# List of acceptable shells for chpass(1).
# Ftpd will not allow users to connect who are not using
# one of these shells.

/bin/bash
/bin/csh
/bin/dash
/bin/ksh
/bin/sh
/bin/tcsh
/bin/zsh
[mehreetsinghbajaj@MacBook-Pro ~ % chsh -s /bin/bash
Changing shell for mehreetsinghbajaj.
[Password for mehreetsinghbajaj:
mehreetsinghbajaj@MacBook-Pro ~ %
```

Start the clisp interface in the terminal



The screenshot shows a terminal window on a Mac OS X desktop. The title bar reads "mehreetsinghbajaj — lisp.run -B /opt/local/lib/clisp-2.49.92 -M /opt/local/lib/cli...". The command "% clisp" is highlighted with a red oval. The terminal displays the standard CLISP welcome message and copyright notices. At the bottom, it shows the prompt "[1]>".

```
[mehreetsinghbajaj@MacBook-Pro ~ % clisp
i i i i i i      oooooo   o     00000000  000000  000000
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     8   8     8     00000  80000
`-__|__-'        8     8   8     8     8   8   8
|               8   o   8   8     8   o   8   8
-----+-----  oooooo  8000000  0008000  00000  8

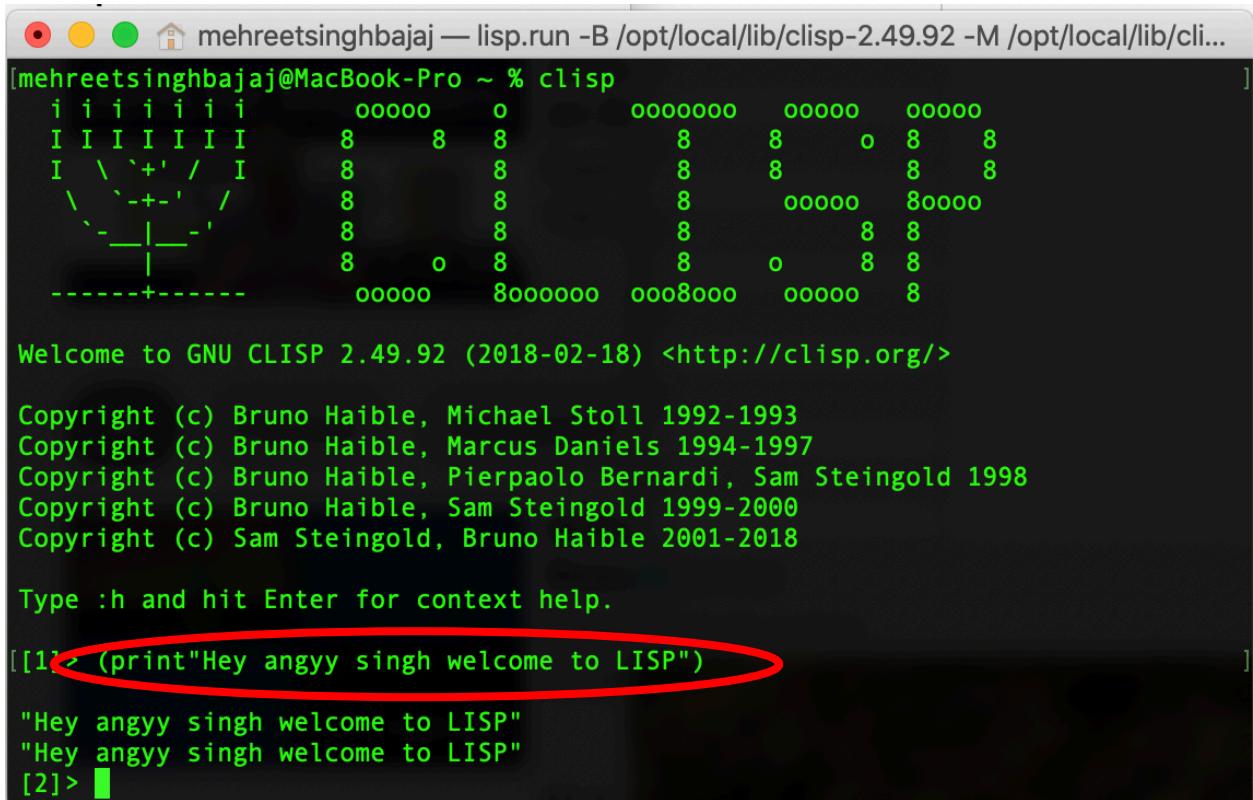
Welcome to GNU CLISP 2.49.92 (2018-02-18) <http://clisp.org/>

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 1999-2000
Copyright (c) Sam Steingold, Bruno Haible 2001-2018

Type :h and hit Enter for context help.

[1]>
```

1. Print function



The screenshot shows a terminal window on a Mac OS X desktop. The title bar reads "mehreetsinghbajaj — lisp.run -B /opt/local/lib/clisp-2.49.92 -M /opt/local/lib/cli...". The command "% clisp" is highlighted with a red oval. The terminal displays the standard CLISP welcome message and copyright notices. A line of code "[1]< (print "Hey angyy singh welcome to LISP")" is highlighted with a red oval. The output shows the string being printed twice. At the bottom, it shows the prompt "[2]>".

```
[mehreetsinghbajaj@MacBook-Pro ~ % clisp
i i i i i i      oooooo   o     00000000  000000  000000
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     8   8     8     00000  80000
`-__|__-'        8     8   8     8     8   8   8
|               8   o   8   8     8   o   8   8
-----+-----  oooooo  8000000  0008000  00000  8

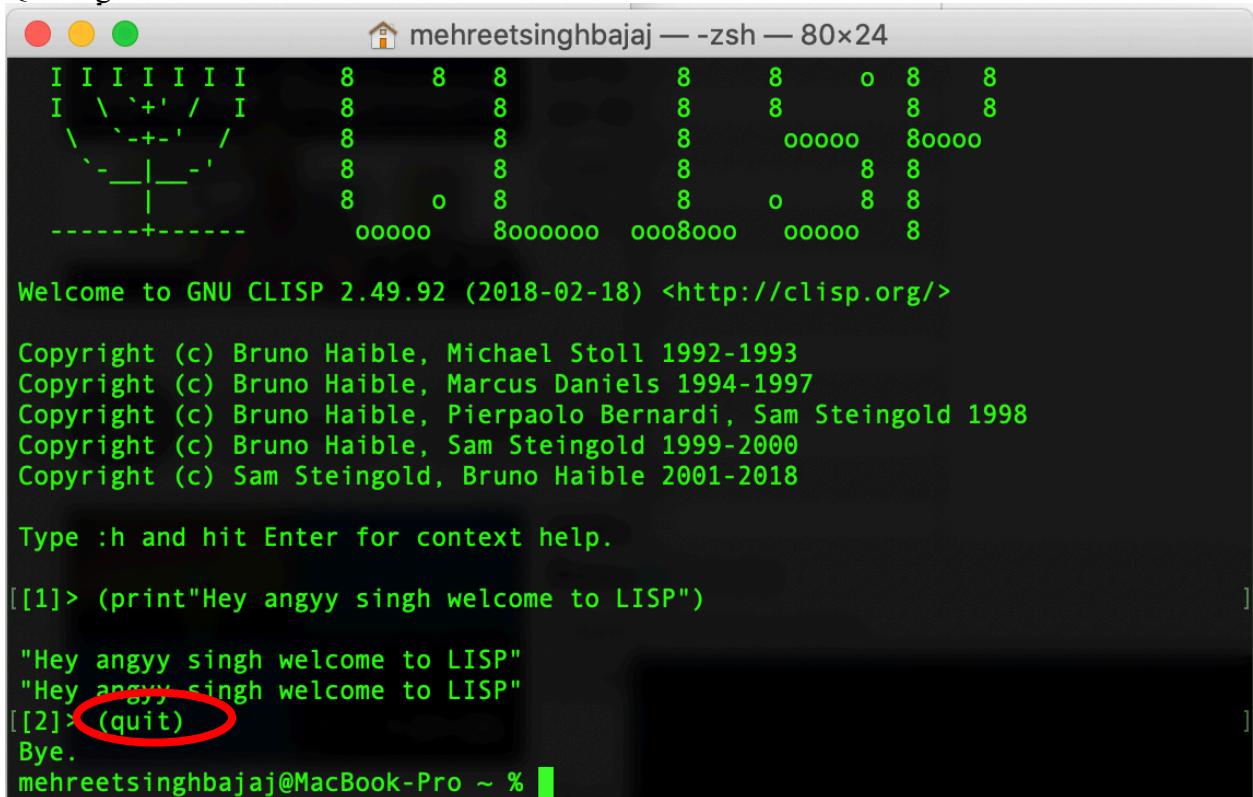
Welcome to GNU CLISP 2.49.92 (2018-02-18) <http://clisp.org/>

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 1999-2000
Copyright (c) Sam Steingold, Bruno Haible 2001-2018

Type :h and hit Enter for context help.

[[1]< (print "Hey angyy singh welcome to LISP")
"Hey angyy singh welcome to LISP"
"Hey angyy singh welcome to LISP"
[2]> ]
```

2. Quitting from the interface



```
mehreetsinghbajaj — zsh — 80x24
I I I I I I I   8   8   8   8   o   8   8
I \ `+' / I   8   8   8   8   8   8   8
\ `--' /   8   8   8   00000   80000
`-__|__-'   8   8   8   8   8   8   8
|   8   o   8   8   o   8   8   8
-----+----- 00000   8000000   0008000   00000   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 1999-2000
Copyright (c) Sam Steingold, Bruno Haible 2001-2018

Type :h and hit Enter for context help.

[[1]> (print "Hey angyy singh welcome to LISP")
[2]> "Hey angyy singh welcome to LISP"
[2]> "Hey angyy singh welcome to LISP"
[[2]> (quit)
Bye.
mehreetsinghbajaj@MacBook-Pro ~ %
```

So , lets start writing our programs in some text editor

;;;
;;; Describe Program

;;; Basic comment

; comment indent to your code

; Comment that comes after a line of code

||

Multiline comments

||#

3. Let's write a basic code in our text editor ,

The screenshot shows a text editor window with the following details:

- Currently Open Documents: LispPrac.clisp, untitled text 4
- File Path: ~/Desktop/Fall 2020/AI/6.Projects/Individual/LispPrac.clisp
- Code in LispPrac.clisp:

```
(format t "Hey angyy singh this side ~%")|
```

- Naming it LispPrac.clisp → extension is .clisp as we will run this on our terminal
- Everything will be encapsulated in ()

```
mehreetsinghbajaj@MacBook-Pro Individual % clisp LispPrac.clisp
Hey angyy singh this side
mehreetsinghbajaj@MacBook-Pro Individual %
```

4. Read and write input / output

The screenshot shows a text editor window with the following details:

- Currently Open Documents: LispPrac.clisp, untitled text 4
- File Path: ~/Desktop/Fall 2020/AI/6.Projects/Individual/LispPrac.clisp
- Code in LispPrac.clisp:

```
(print "Please say your name ")
(defvar *name* (read))
(defun hello-your(*name*)
  (format t "Welcome to the world pf lisp ~a! ~%" *name*)
)
(setq *print-case* :capitalize )
(hello-your *name*)
```

```
Individual — zsh — 80x24
~/Desktop/Fall 2020/AI/6.Projects/Individual — zsh
mehreetsinghbajaj@MacBook-Pro Individual % clisp LispPrac.clisp

"Please say your name " angyysingh
Welcome to the world pf lisp Angyysingh!
mehreetsinghbajaj@MacBook-Pro Individual %
```

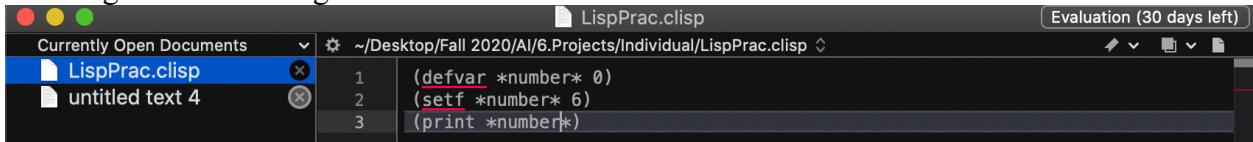
Defining a variable using defvar

Definig a function using defun

format -t helps printing info on terminal

now ~a	: shows the vale
~s	: shows the quote around values
~10a	: add 10 space to the value with extra space to the right
~10@a	: add 10 space to the value with extra space to the left

5. Defining a variable using defvar



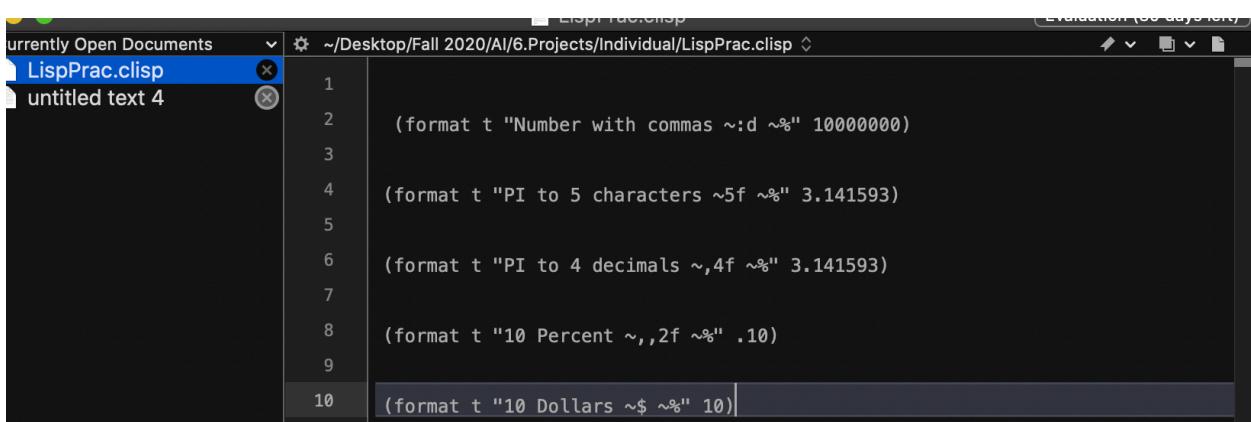
LispPrac.clisp

```
(defvar *number* 0)
(setf *number* 6)
(print *number*)
```

Setf allows us to change the value of variable

```
mehreetsinghbajaj@MacBook-Pro Individual % clisp LispPrac.clisp
6
mehreetsinghbajaj@MacBook-Pro Individual %
```

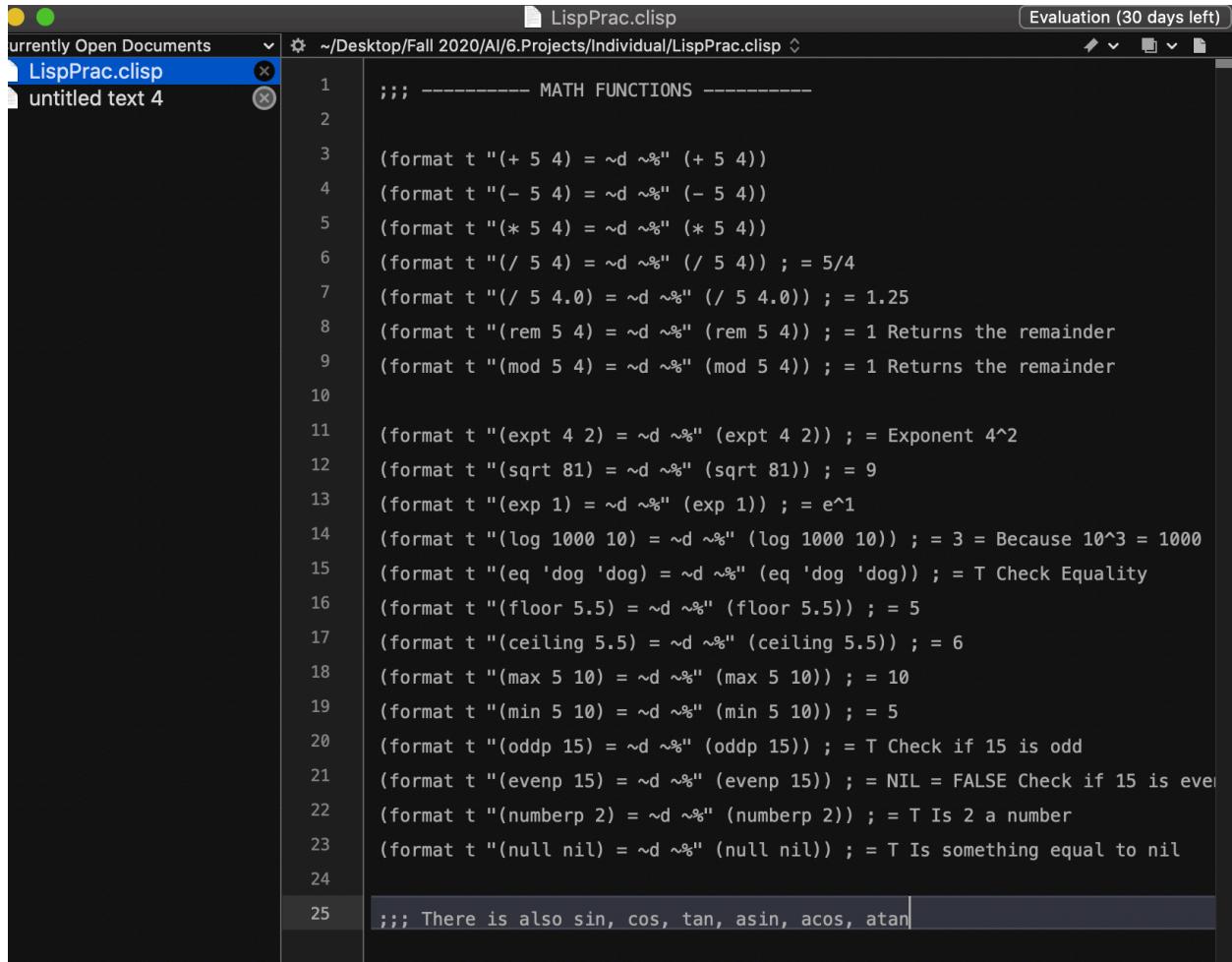
6. Playing with factor method



```
(format t "Number with commas ~:d ~%" 10000000)
(format t "PI to 5 characters ~5f ~%" 3.141593)
(format t "PI to 4 decimals ~,4f ~%" 3.141593)
(format t "10 Percent ~,,2f ~%" .10)
(format t "10 Dollars ~$ ~%" 10)
```

```
mehreetsinghbajaj@MacBook-Pro Individual % clisp LispPrac.clisp
Number with commas 10,000,000
PI to 5 characters 3.142
PI to 4 decimals 3.1416
10 Percent 10.0
10 Dollars 10.00
mehreetsinghbajaj@MacBook-Pro Individual %
```

7. Mathematical Functions



The screenshot shows a Lisp editor window titled "LispPrac.clisp". The status bar indicates "Evaluation (30 days left)". The code in the buffer is as follows:

```
1  ;;; ----- MATH FUNCTIONS -----
2
3  (format t "(+ 5 4) = ~d ~%" (+ 5 4))
4  (format t "(- 5 4) = ~d ~%" (- 5 4))
5  (format t "(* 5 4) = ~d ~%" (* 5 4))
6  (format t "(/ 5 4) = ~d ~%" (/ 5 4)) ; = 5/4
7  (format t "(/ 5 4.0) = ~d ~%" (/ 5 4.0)) ; = 1.25
8  (format t "(rem 5 4) = ~d ~%" (rem 5 4)) ; = 1 Returns the remainder
9  (format t "(mod 5 4) = ~d ~%" (mod 5 4)) ; = 1 Returns the remainder
10
11 (format t "(expt 4 2) = ~d ~%" (expt 4 2)) ; = Exponent 4^2
12 (format t "(sqrt 81) = ~d ~%" (sqrt 81)) ; = 9
13 (format t "(exp 1) = ~d ~%" (exp 1)) ; = e^1
14 (format t "(log 1000 10) = ~d ~%" (log 1000 10)) ; = 3 = Because 10^3 = 1000
15 (format t "(eq 'dog 'dog) = ~d ~%" (eq 'dog 'dog)) ; = T Check Equality
16 (format t "(floor 5.5) = ~d ~%" (floor 5.5)) ; = 5
17 (format t "(ceiling 5.5) = ~d ~%" (ceiling 5.5)) ; = 6
18 (format t "(max 5 10) = ~d ~%" (max 5 10)) ; = 10
19 (format t "(min 5 10) = ~d ~%" (min 5 10)) ; = 5
20 (format t "(oddp 15) = ~d ~%" (oddp 15)) ; = T Check if 15 is odd
21 (format t "(evenp 15) = ~d ~%" (evenp 15)) ; = NIL = FALSE Check if 15 is even
22 (format t "(numberp 2) = ~d ~%" (numberp 2)) ; = T Is 2 a number
23 (format t "(null nil) = ~d ~%" (null nil)) ; = T Is something equal to nil
24
25 ;;; There is also sin, cos, tan, asin, acos, atan|
```

```
mehreetsinghbajaj@MacBook-Pro Individual % clisp LispPrac.clisp
(+ 5 4) = 9
(- 5 4) = 1
(* 5 4) = 20
(/ 5 4) = 5/4
(/ 5 4.0) = 1.25
(rem 5 4) = 1
(mod 5 4) = 1
(expt 4 2) = 16
(sqrt 81) = 9
(exp 1) = 2.7182817
(log 1000 10) = 3
(eq 'dog 'dog) = T
(floor 5.5) = 5
(ceiling 5.5) = 6
(max 5 10) = 10
(min 5 10) = 5
(oddp 15) = T
(evenp 15) = NIL
(numberp 2) = T
(null nil) = T
mehreetsinghbajaj@MacBook-Pro Individual %
```

8. Equality checking

The screenshot shows a Lisp IDE interface with two panes. The top pane displays a Lisp source file named `LispPrac.clisp`. The code defines a parameter `*name*` and compares various data types using `eq`, `equal`, and `equalp`. The bottom pane shows the terminal output of running the code, demonstrating the results of these comparisons.

```
LispPrac.clisp
;; ----- EQUALITY -----
;;
;; Symbols are compared with eq
(defparameter *name* 'Derek)
(format t "(eq *name* 'Derek) = ~d ~%" (eq *name* 'Derek))

;; Everything else is compared with equal for the most part
(format t "(equal 'car 'truck) = ~d ~%" (equal 'car 'truck))
(format t "(equal 10 10) = ~d ~%" (equal 10 10))
(format t "(equal 5.5 5.3) = ~d ~%" (equal 5.5 5.3))
(format t "(equal \"string\" \"String\") = ~d ~%" (equal "string" "String"))
(format t "(equal (list 1 2 3) (list 1 2 3)) = ~d ~%" 
(equal (list 1 2 3) (list 1 2 3)))
(equal (list 1 2 3) (list 1 2 3)))

~/Desktop/Fall 2020/AI/6.Projects/Individual --zsh
mehreetsinghbajaj@MacBook-Pro Individual % clisp LispPrac.clisp
(eq *name* 'Derek) = T
(equal 'car 'truck) = NIL
(equal 10 10) = T
(equal 5.5 5.3) = NIL
(equal "string" "String") = NIL
(equal (list 1 2 3) (list 1 2 3)) = T
mehreetsinghbajaj@MacBook-Pro Individual %
```

9. Comparing Strings

The screenshot shows a Lisp IDE interface with two panes. The top pane displays a Lisp source file named `LispPrac.clisp`. The code demonstrates the use of `equalp` to compare strings, integers, and floats. The bottom pane shows the terminal output of running the code, illustrating the results of these comparisons.

```
LispPrac.clisp
;; equalp can compare strings of any case and integers to floats
(format t "(equalp 1.0 1) = ~d ~%" (equalp 1.0 1))
(format t "(equalp \"Derek\" \"derek\") = ~d ~%" (equalp "Derek" "derek"))

mehreetsinghbajaj@MacBook-Pro Individual % clisp LispPrac.clisp
(equalp 1.0 1) = T
(equalp "Derek" "derek") = T
mehreetsinghbajaj@MacBook-Pro Individual %
```

10. .Conditionals, not equal , logical operators, multiple statements using if

```
Currently Open Documents ~/Desktop/Fall 2020/AI/6.Projects/Individual/LispPrac.clisp
LispPrac.clisp
untitled text 4

1  ;; ----- CONDITIONALS -----
2
3  (defparameter *age* 18) ; Create variable age
4
5  ;;; Relational Operators > < >= <= =
6
7  ;;; Check if age is greater than or equal to 18
8
9  (if (= *age* 18)
10    (format t "You can vote~%")
11    (format t "You can't vote~%"))
12
13  ;;; How to check for not equal
14
15  (if (not (= *age* 18))
16    (format t "You can vote~%")
17    (format t "You can't vote~%"))
18
19  ;;; Logical Operators : and, or, not
20
21  (if (and (>= *age* 18) (<= *age* 67) )
22    (format t "Time for work~%")
23    (format t "Work if you want~%"))
24
25  (if (or (<= *age* 14) (>= *age* 67) )
26    (format t "You shouldn't work~%")
27    (format t "You should work~%"))
28
29  (defparameter *num* 2)
30  (defparameter *num-2* 2)
31  (defparameter *num-3* 2)
32
33  ;;; You can execute multiple statements in an if with progn
34
35  (if (= *num* 2)
36    (progn
37      (setf *num-2* (* *num-2* 2))
38      (setf *num-3* (* *num-3* 3)))
39    )
40    (format t "Not equal to 2~%"))
41
42  (format t "*num-2* = ~d ~%" *num-2*)
43  (format t "*num-3* = ~d ~%" *num-3*)
44
```

```
45     ;;; Case performs certain actions depending on conditions
46     (defun get-school (age)
47         (case age
48             (5 (print "Kindergarten"))
49             (6 (print "First Grade"))
50             (otherwise '(middle school))
51         )))
52
53     (get-school 5)
54
55     (terpri) ; Newline
56
57     ;;; when allows you to execute multiple statements by default
58
59     (when (= *age* 18)
60         (setf *num-3* 18)
61         (format t "Go to college you're ~d ~%" *num-3*))
62 )
```

```
mehreetsinghbajaj@MacBook-Pro Individual % clisp LispPrac.clisp
You can vote
You can't vote
Time for work
You should work
*num-2* = 4
*num-3* = 6

"Kindergarten"
Go to college you're 18
mehreetsinghbajaj@MacBook-Pro Individual %
```

❖ EXERCISE – 1: DRIBBLE SESSION A

```
mehreetsinghbajaj — lisp.run -B /opt/local/lib/clisp-2.49.92 -M /opt/local/lib/clisp-2.49.92/base/lispinit.mem -N /opt/local/share/locale — 80x24
~ — lisp.run -B /opt/local/lib/clisp-2.49.92 -M /opt/local/lib/clisp-2.49.92/base/lispinit.mem -N /opt/local/share/locale

|          8      o     8          8      o     8   8
-----+----- 0ooooo    8ooooooo  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 1999-2000
Copyright (c) Sam Steingold, Bruno Haible 2001-2018

Type :h and hit Enter for context help.

[[1]> 'a'
A
[[2]> 5
5
[[3]> 3.183
3.183
[[4]> nil
NIL
[[5]> "hello"
"hello"
[[6]>
```

- Write the normal char and they will be printed in capital
- Input the numerical and it will echo
- Input the information you want to print in between “ ” and it will print it like that

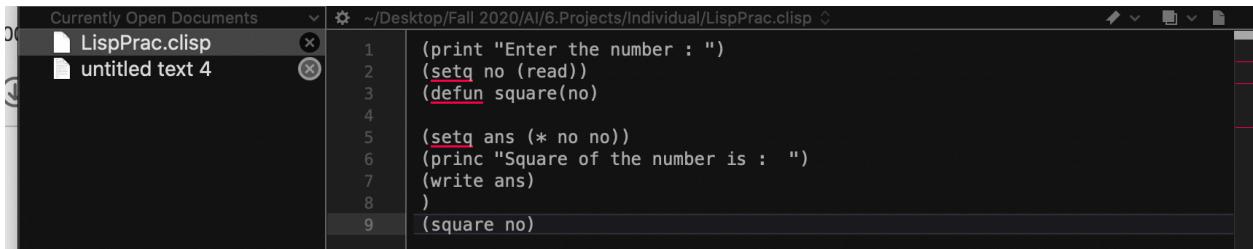
❖ EXERCISE – 1: DRIBBLE SESSION B (BUILT IN FUNCTION)

```
mehreetsinghbajaj — lisp.run -B /opt/local/lib/clisp-2.49.92 -M /opt/local/lib/clisp-2.49.92/base/lispinit.mem -N /opt/local/share/locale — 80x24
~ — lisp.run -B /opt/local/lib/clisp-2.49.92 -M /opt/local/lib/clisp-2.49.92/base/lispinit.mem -N /opt/local/share/locale +
```

```
[[1]> (+ 5 3)
8
[[2]> (setf a 6)
6
[[3]> (setf b 12)
12
[[4]> a
6
[[5]> b
12
[[6]> (* a b)
72
[[7]> (setf c (/ a b))
1/2
[[8]> c
1/2
[[9]> (* a a)
36
[[10]> (* b b b)
1728
[[11]> (sqrt a)
2.4494898
[[12]> (sqrt (* a a))
6
[[13]> (/ (* b b b) b b)
12
[14]> █
```

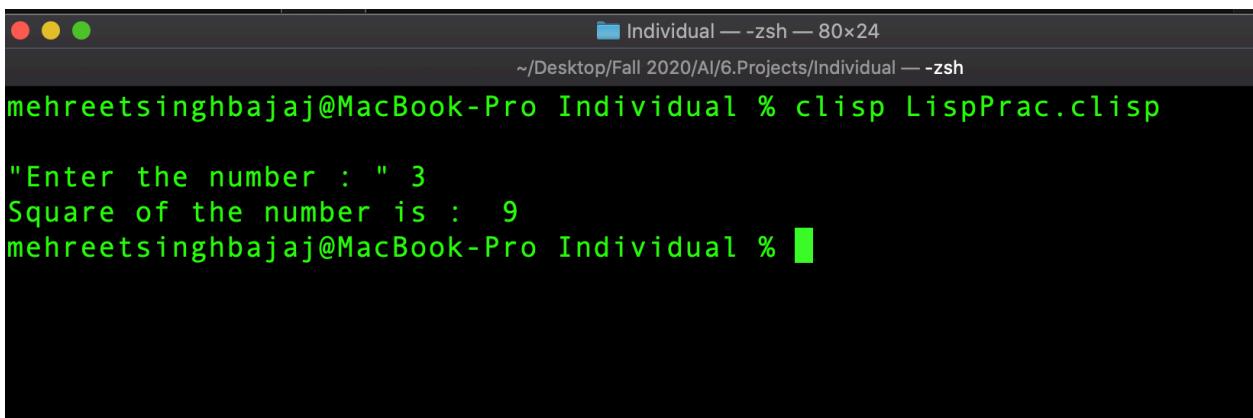
- (operator values)
- Setf- sets the values of function (setf variable value)
- Nested operator – (sqrt (operator var var))

❖ EXERCISE – 1: DRIBBLE SESSION C (SQUARE AND CUBE FUNCTIONS)



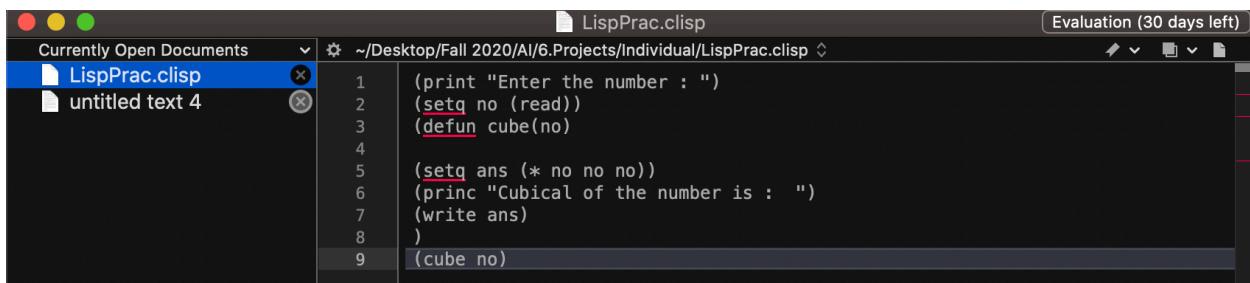
```
Currently Open Documents ~/Desktop/Fall 2020/AI/6.Projects/Individual/LispPrac.clisp
LispPrac.clisp
untitled text 4

1 (print "Enter the number : ")
2 (setq no (read))
3 (defun square(no)
4
5   (setq ans (* no no))
6   (princ "Square of the number is : ")
7   (write ans)
8 )
9 (square no)
```



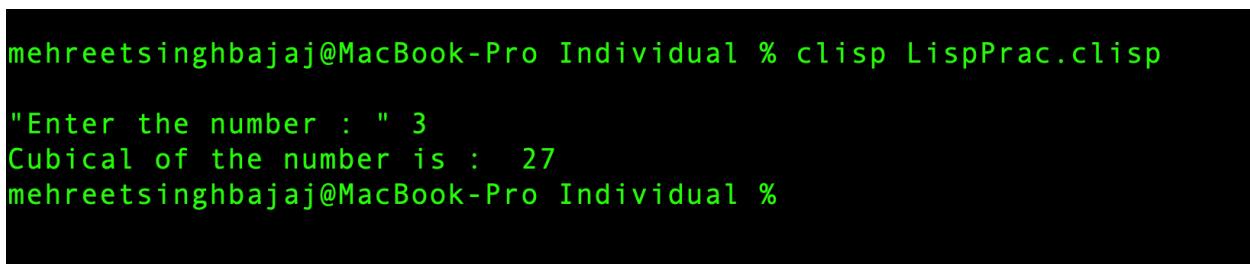
```
Individual — -zsh — 80x24
~/Desktop/Fall 2020/AI/6.Projects/Individual — -zsh
mehreetsinghbajaj@MacBook-Pro Individual % clisp LispPrac.clisp

"Enter the number : " 3
Square of the number is :  9
mehreetsinghbajaj@MacBook-Pro Individual %
```



```
Currently Open Documents ~/Desktop/Fall 2020/AI/6.Projects/Individual/LispPrac.clisp
LispPrac.clisp
untitled text 4

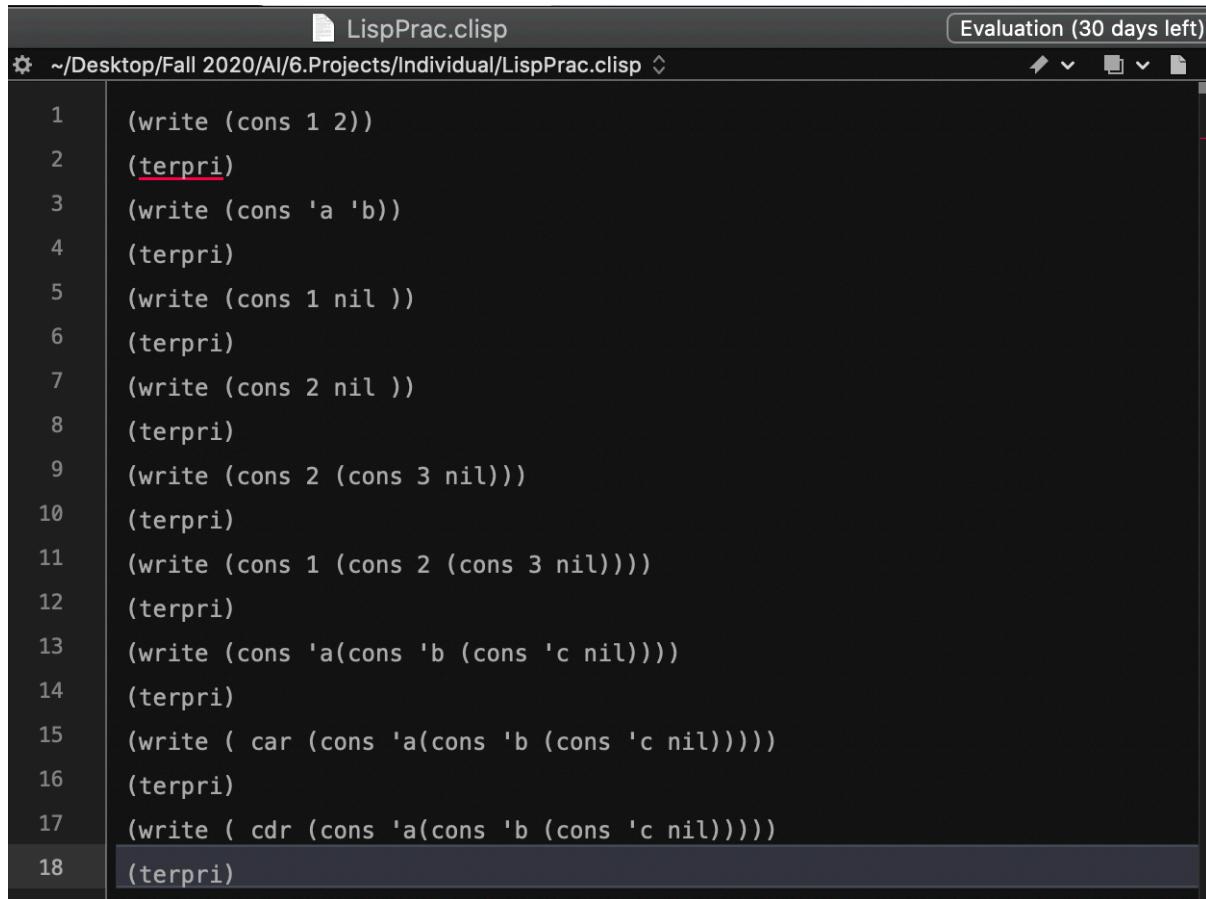
1 (print "Enter the number : ")
2 (setq no (read))
3 (defun cube(no)
4
5   (setq ans (* no no no))
6   (princ "Cubical of the number is : ")
7   (write ans)
8 )
9 (cube no)
```



```
mehreetsinghbajaj@MacBook-Pro Individual % clisp LispPrac.clisp

"Enter the number : " 3
Cubical of the number is :  27
mehreetsinghbajaj@MacBook-Pro Individual %
```

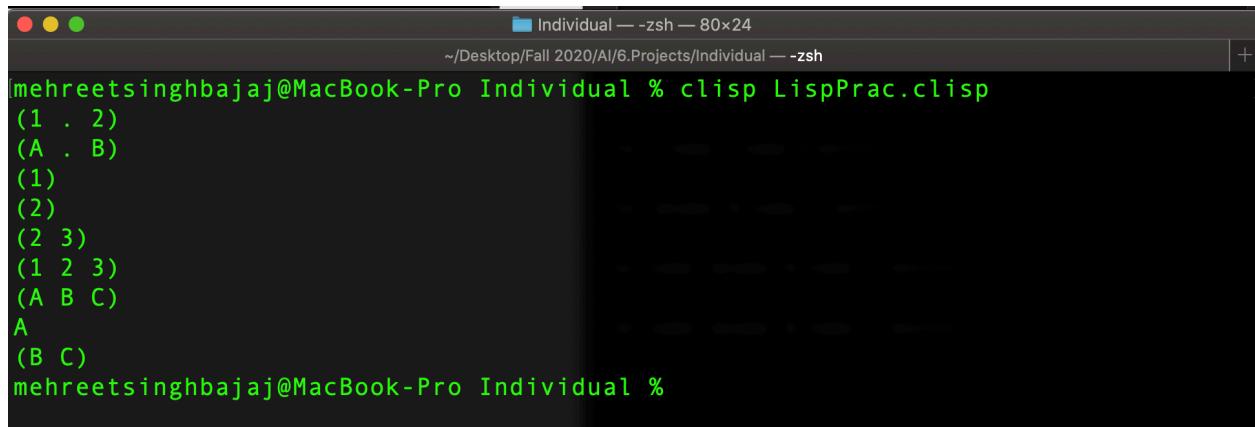
❖ EXERCISE – 1: DRIBBLE SESSION C (SQUARE AND CUBE FUNCTIONS)



LispPrac.clisp

Evaluation (30 days left)

```
1 (write (cons 1 2))
2 (terpri)
3 (write (cons 'a 'b))
4 (terpri)
5 (write (cons 1 nil ))
6 (terpri)
7 (write (cons 2 nil ))
8 (terpri)
9 (write (cons 2 (cons 3 nil)))
10 (terpri)
11 (write (cons 1 (cons 2 (cons 3 nil))))
12 (terpri)
13 (write (cons 'a(cons 'b (cons 'c nil))))
14 (terpri)
15 (write ( car (cons 'a(cons 'b (cons 'c nil)))))
16 (terpri)
17 (write ( cdr (cons 'a(cons 'b (cons 'c nil)))))
18 (terpri)
```



Individual — -zsh — 80x24

~/Desktop/Fall 2020/AI/6.Projects/Individual — -zsh

```
mehreetsinghbajaj@MacBook-Pro Individual % clisp LispPrac.clisp
(1 . 2)
(A . B)
(1)
(2)
(2 3)
(1 2 3)
(A B C)
A
(B C)
mehreetsinghbajaj@MacBook-Pro Individual %
```

❖ EXERCISE – 2: A. Factorial

The screenshot shows a Mac OS X desktop environment. At the top, there is a dock with icons for Finder, Mail, Safari, and others. Above the dock, the system menu bar displays the date and time. Below the dock, there is a vertical stack of windows.

The top window is titled "LispPrac.clisp" and shows the code for a factorial function:

```
(defun factorial (n)
  (if (= n 0)
      1
      (* n (factorial (- n 1)))))

(loop for i from 0 to 16
      do (format t "~D! = ~D~%" i (factorial i)))
```

The second window in the stack is titled "Individual — -zsh — 80x24" and shows the output of running the Lisp code:

```
mehreetsinghbajaj@MacBook-Pro Individual % clisp LispPrac.clisp
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
11! = 39916800
12! = 479001600
13! = 6227020800
14! = 87178291200
15! = 1307674368000
16! = 20922789888000
mehreetsinghbajaj@MacBook-Pro Individual %
```

❖ EXERCISE – 2: B. Fibonacci

The screenshot shows a terminal window with two panes. The top pane is a code editor for Lisp, displaying a script named `LispPrac.clisp`. The code defines a function `fibo(n)` to generate the first `n` numbers of the Fibonacci series. The bottom pane is a terminal window titled `Individual — -zsh — 80x24`, showing the output of running the Lisp code to print the first 20 numbers of the series.

```
LispPrac.clisp
~/Desktop/Fall 2020/AI/6.Projects/Individual/LispPrac.clisp
Evaluation (30 days left)

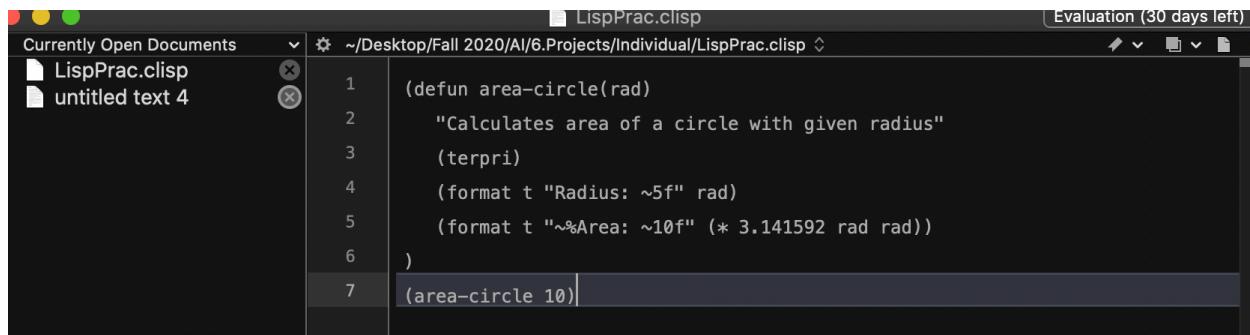
Currently Open Documents
LispPrac.clisp
untitled text 4

1 ;Fibonacci series - 0 1 1 2 3 5 ...
2 (setq first -1)
3 (setq second 1)
4 (defun fibo(n)
5   ( loop for x from 1 to n
6         do( setq third (+ first second))( print third)
7             do( setq first second)
8                 do( setq second third)
9                   )
10      )
11 ( fibo 20)

Individual — -zsh — 80x24
~/Desktop/Fall 2020/AI/6.Projects/Individual — -zsh
[mehreetsinghbajaj@MacBook-Pro Individual % clisp LispPrac.clisp

0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
mehreetsinghbajaj@MacBook-Pro Individual % ]
```

❖ EXERCISE – 2: C. Area of circle



LispPrac.clisp

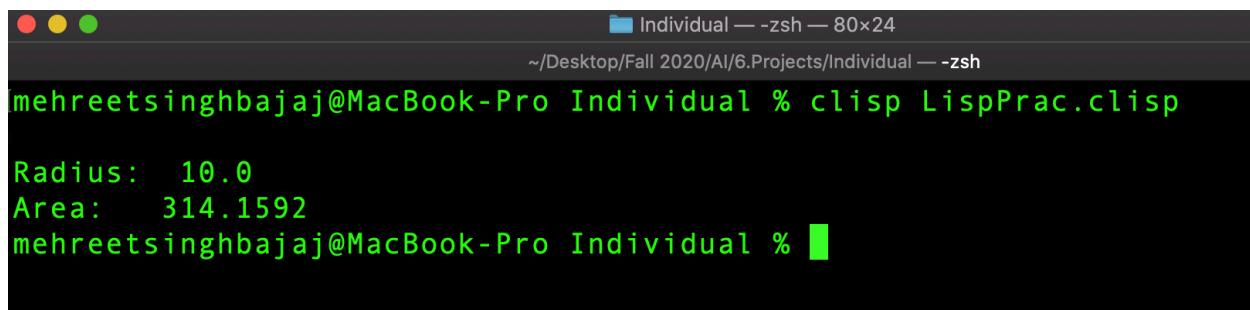
Currently Open Documents

LispPrac.clisp

untitled text 4

```
(defun area-circle(rad)
  "Calculates area of a circle with given radius"
  (terpri)
  (format t "Radius: ~5f" rad)
  (format t "~%Area: ~10f" (* 3.141592 rad rad)))
(area-circle 10)
```

Evaluation (30 days left)



```
Individual — -zsh — 80x24
~/Desktop/Fall 2020/AI/6.Projects/Individual — -zsh
mehreetsinghbajaj@MacBook-Pro Individual % clisp LispPrac.clisp

Radius: 10.0
Area: 314.1592
mehreetsinghbajaj@MacBook-Pro Individual %
```

❖ EXERCISE – 2: D. User Input Output

The screenshot shows a software interface for editing and running Lisp code. On the left, there's a sidebar titled "Currently Open Documents" with two entries: "LispPrac.clisp" and "untitled text 4". The main workspace is titled "LispPrac.clisp" and contains the following Lisp code:

```
1 (defun DoubleNumber()
2   (terpri)
3   (princ "Enter Number : ")
4   (setq n1 (read))
5   (setq doubled (* 2.0 n1 ))
6   (princ "The Number : ")
7   (write n1)
8   (terpri)
9   (princ "The number doubled : ")
10  (write doubled)
11  )
12 (DoubleNumber)
```

Below the editor, a terminal window is open with the title "Individual — -zsh — 80x24" and the command "mehreetsinghbajaj@MacBook-Pro Individual % clisp LispPrac.clisp". The terminal output shows the program's interaction with the user:

```
Enter Number : 12
The Number : 12
The number doubled : 24.0
mehreetsinghbajaj@MacBook-Pro Individual %
```