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.

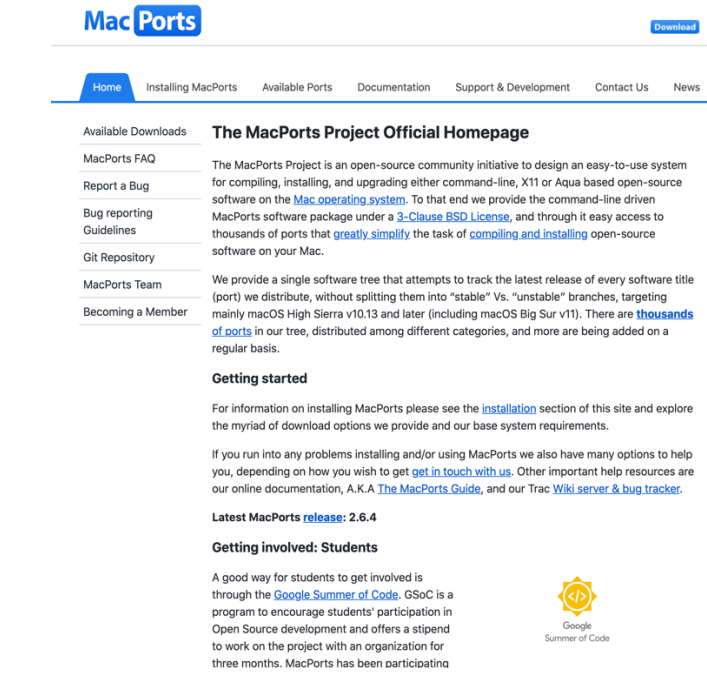
- Download a common LISP from the internet.
- Install the common LISP that you downloaded
- **DO NOT PAY FOR THIS**

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:



MacPorts

Download

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

- Available Downloads
- MacPorts FAQ
- Report a Bug
- Bug reporting Guidelines
- Git Repository
- MacPorts Team
- Becoming a Member

The MacPorts Project Official Homepage

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.

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.

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.

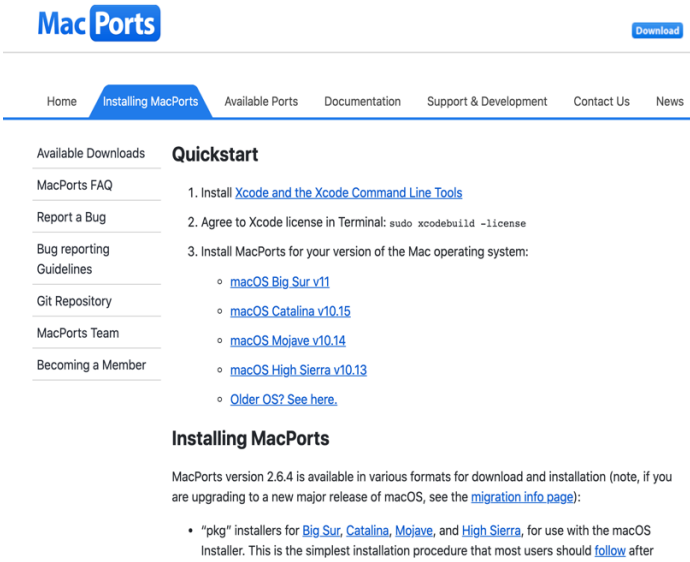
If you run into any problems installing and/or using MacPorts we also have many options to help you, depending on how you wish to get get in touch with us. Other important help resources are our online documentation, A.K.A The MacPorts Guide, and our Trac Wiki server & bug tracker.

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

Google Summer of Code



MacPorts

Download

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

- Available Downloads
- MacPorts FAQ
- Report a Bug
- Bug reporting Guidelines
- Git Repository
- MacPorts Team
- Becoming a Member

Quickstart

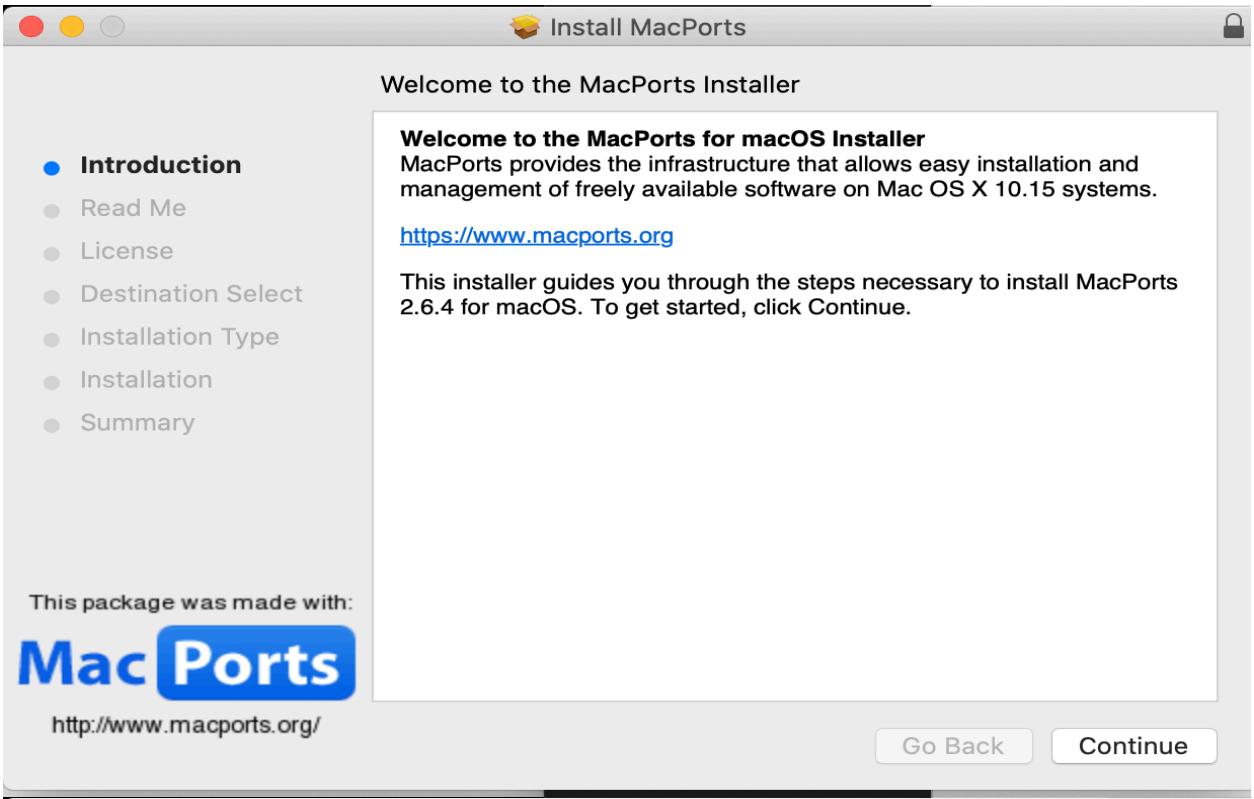
1. Install Xcode and the Xcode Command Line Tools
2. Agree to Xcode license in Terminal: `sudo xcodebuild -license`
3. Install MacPorts for your version of the Mac operating system:
   - macOS Big Sur v11
   - macOS Catalina v10.15
   - macOS Mojave v10.14
   - macOS High Sierra v10.13
   - Older OS? See here.

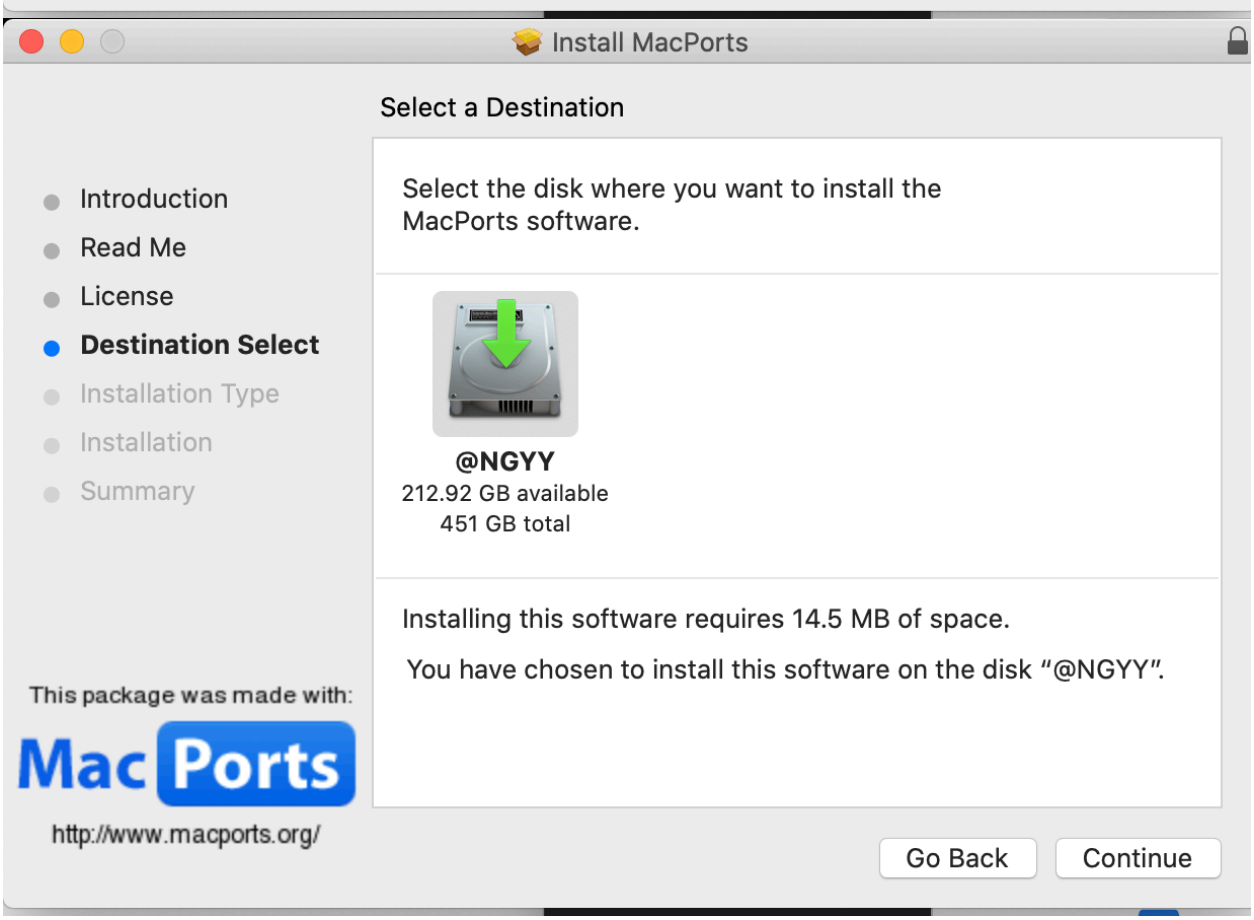
Installing MacPorts

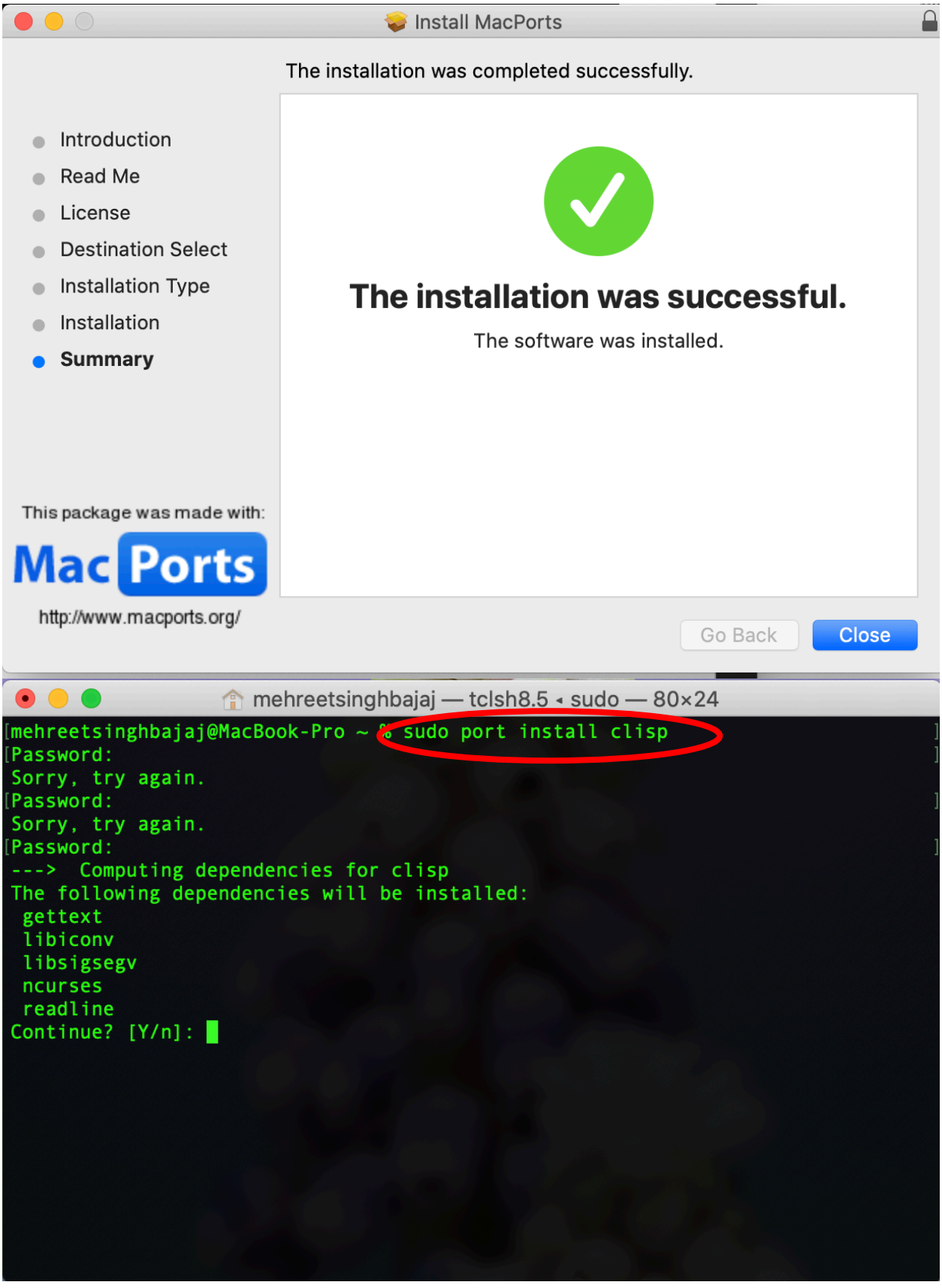
MacPorts version 2.6.4 is available in various formats for download and installation (note, if you are upgrading to a new major release of macOS, see the migration info page):

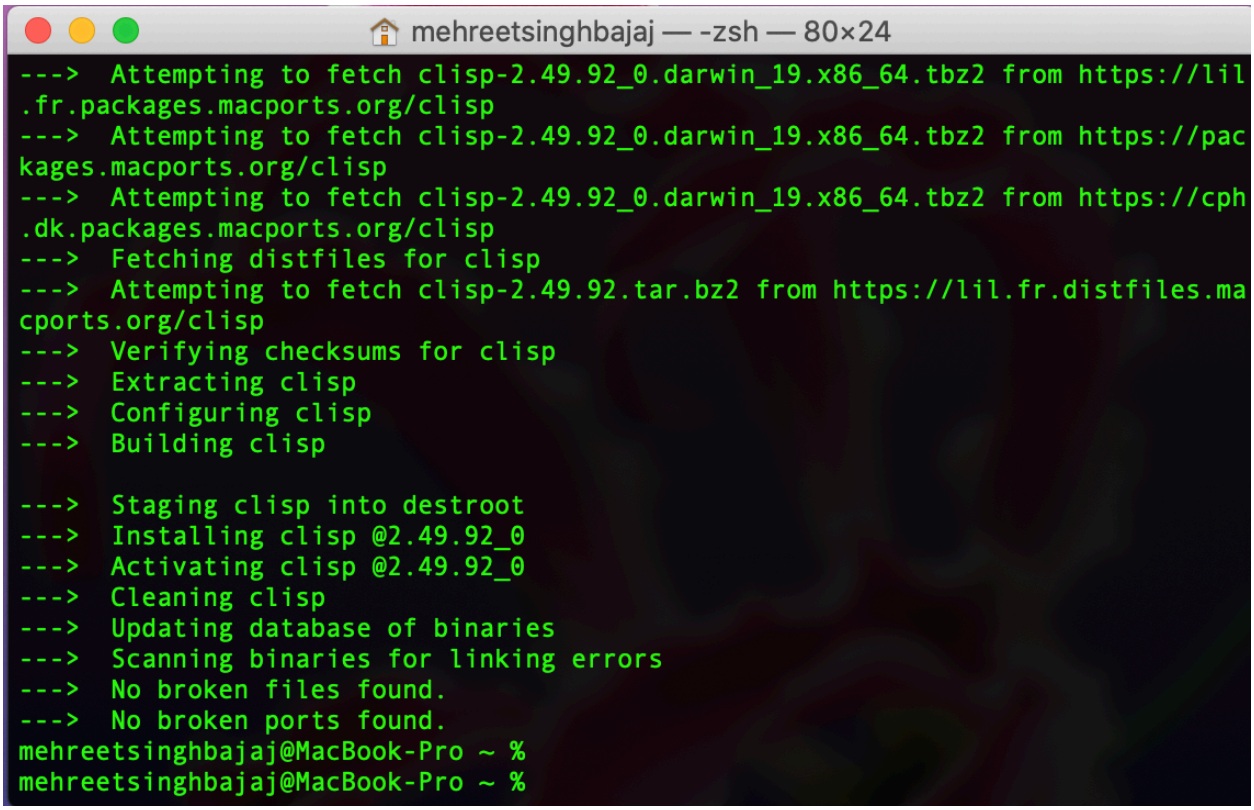
- "pkg" installers for Big Sur, Catalina, Mojave, and High Sierra, for use with the macOS Installer. This is the simplest installation procedure that most users should follow after meeting the requirements listed below. Installers for legacy platforms Sierra, El Capitan, Yosemite, Mavericks, Mountain 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.

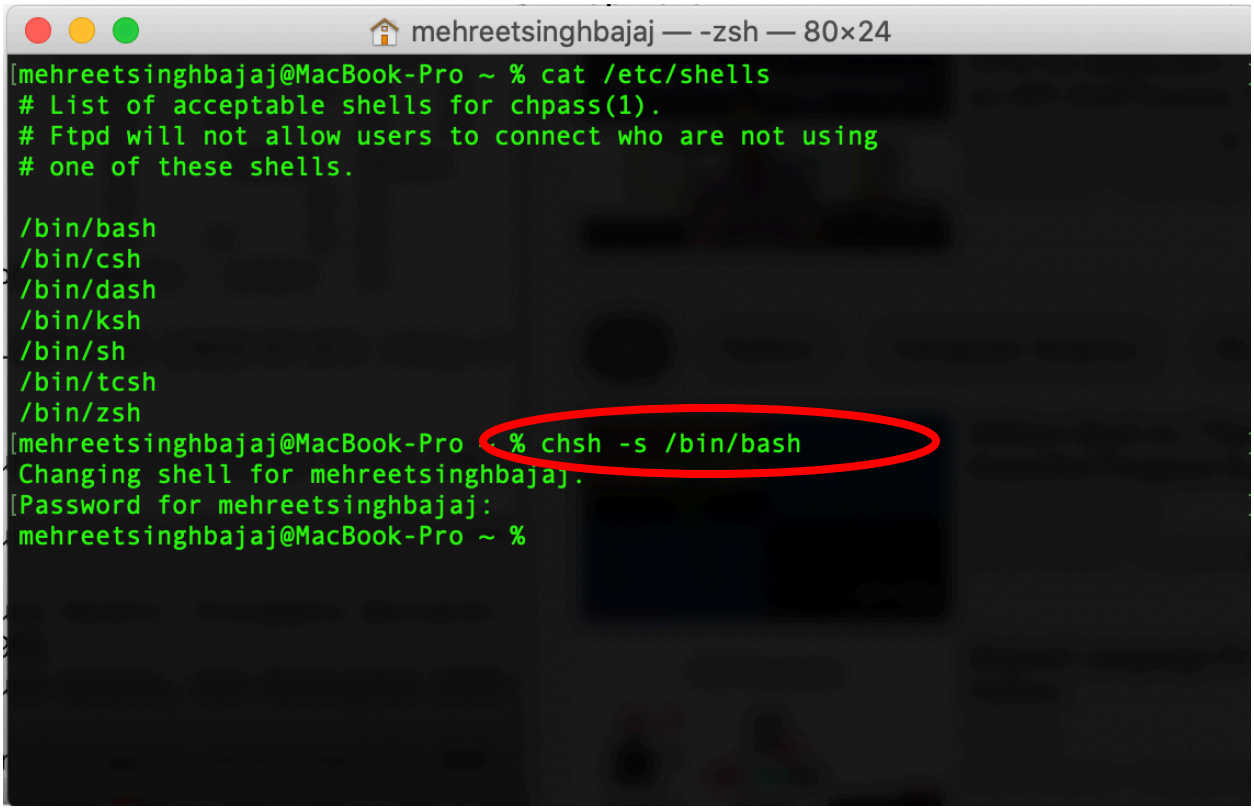

Install MacPorts
Welcome to the MacPorts Installer
Introduction
Read Me
License
Destination Select
Installation Type
Installation
Summary
Welcome to the MacPorts for macOS Installer
MacPorts provides the infrastructure that allows easy installation and management of freely available software on Mac OS 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:
MacPorts
http://www.macports.org/
Go Back
Continue


Install MacPorts
Select a Destination
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:
MacPorts
http://www.macports.org/
Go Back
Continue

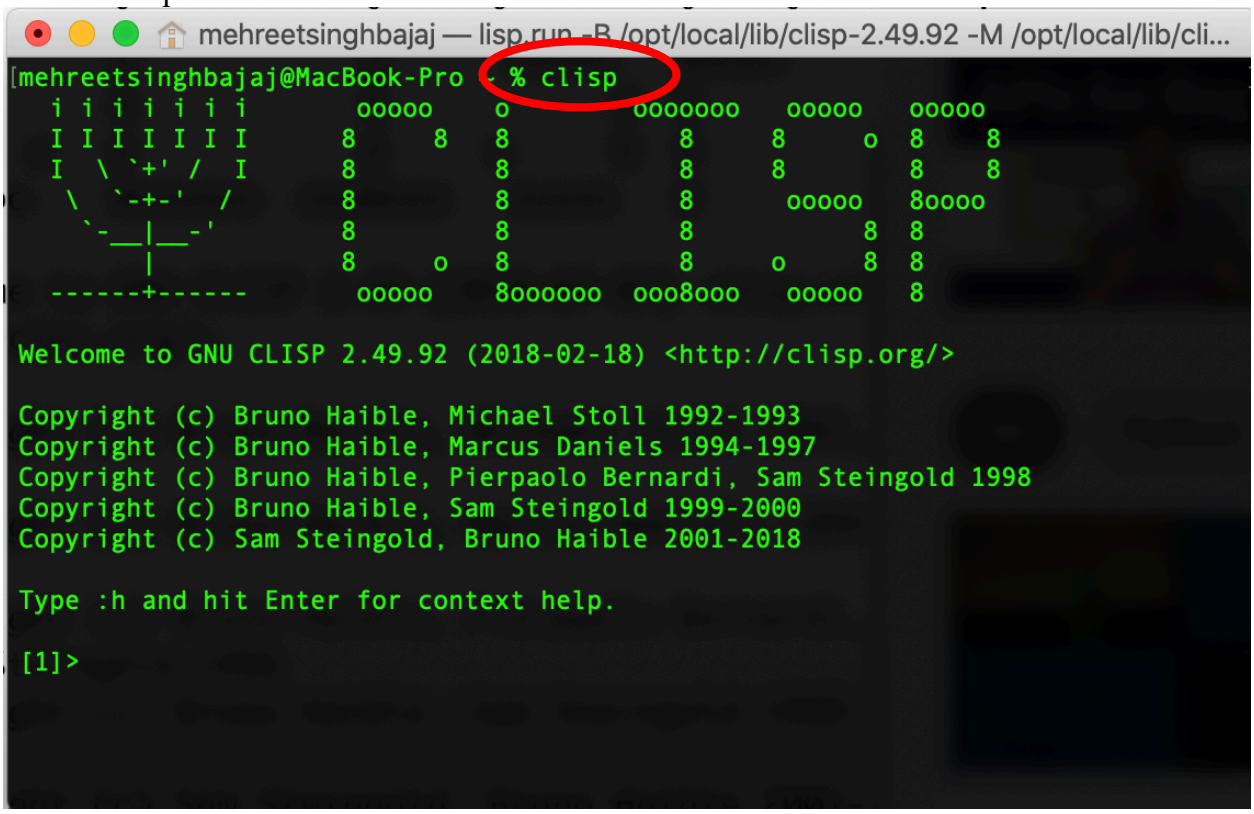

Install MacPorts
The installation was completed successfully.
Introduction
Read Me
License
Destination Select
Installation Type
Installation
Summary
The installation was successful.
The software was installed.
This package was made with:
MacPorts
http://www.macports.org/
Go Back
Close
mehreetsinghbajaj — tclsh8.5 ◂ sudo — 80×24
[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]:



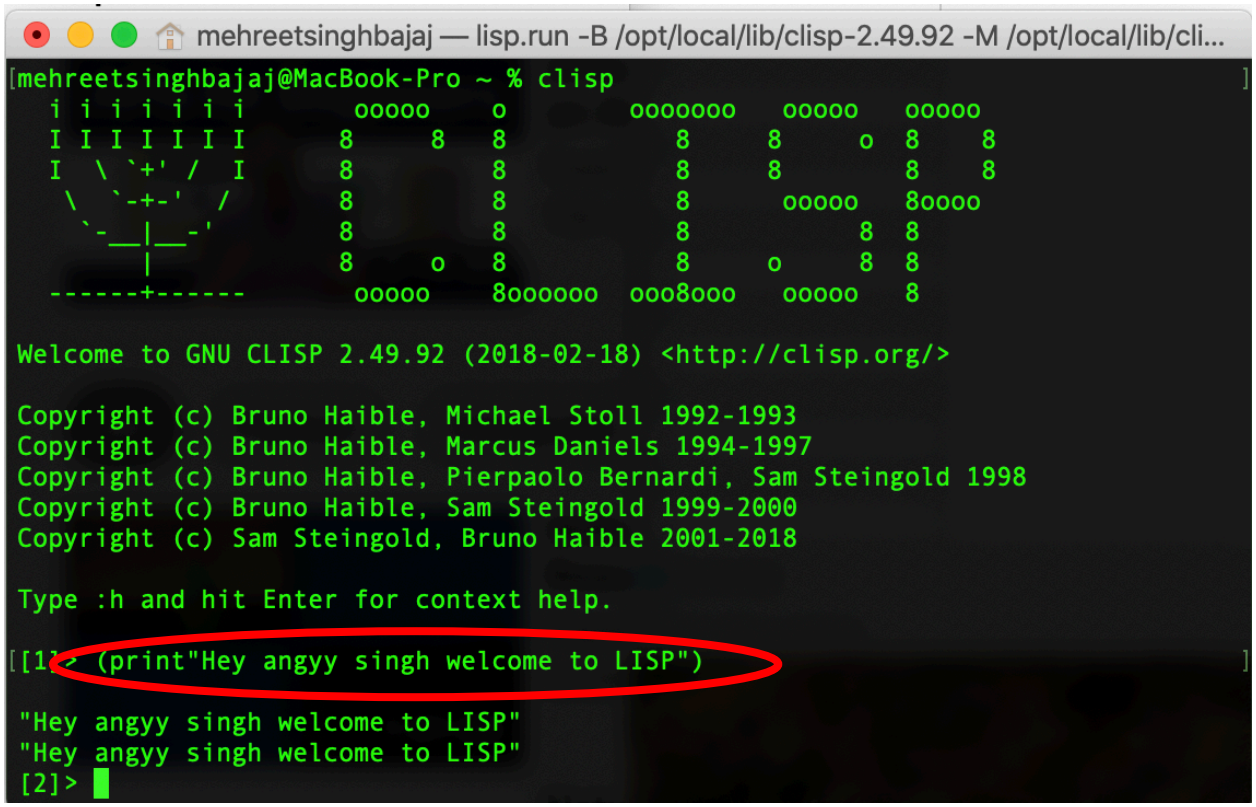
Let's start working in our bash shell



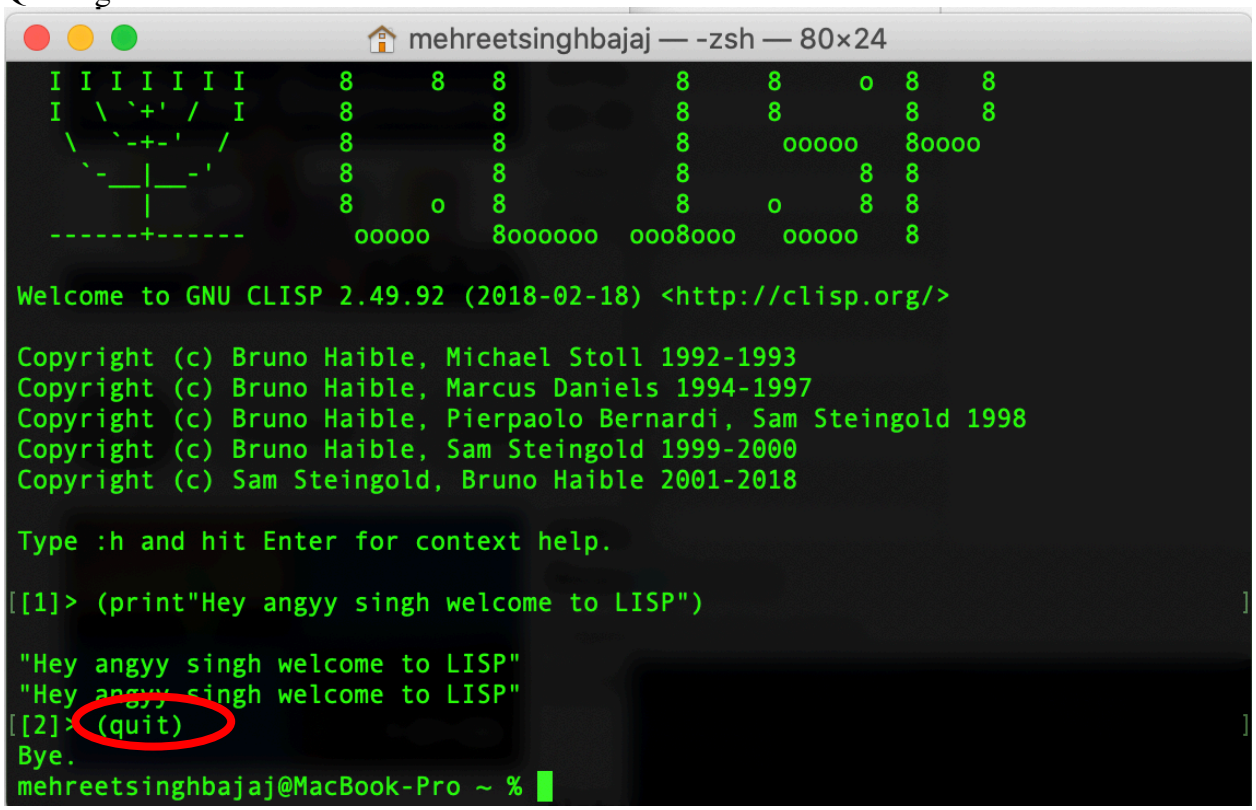
Start the clisp interface in the terminal



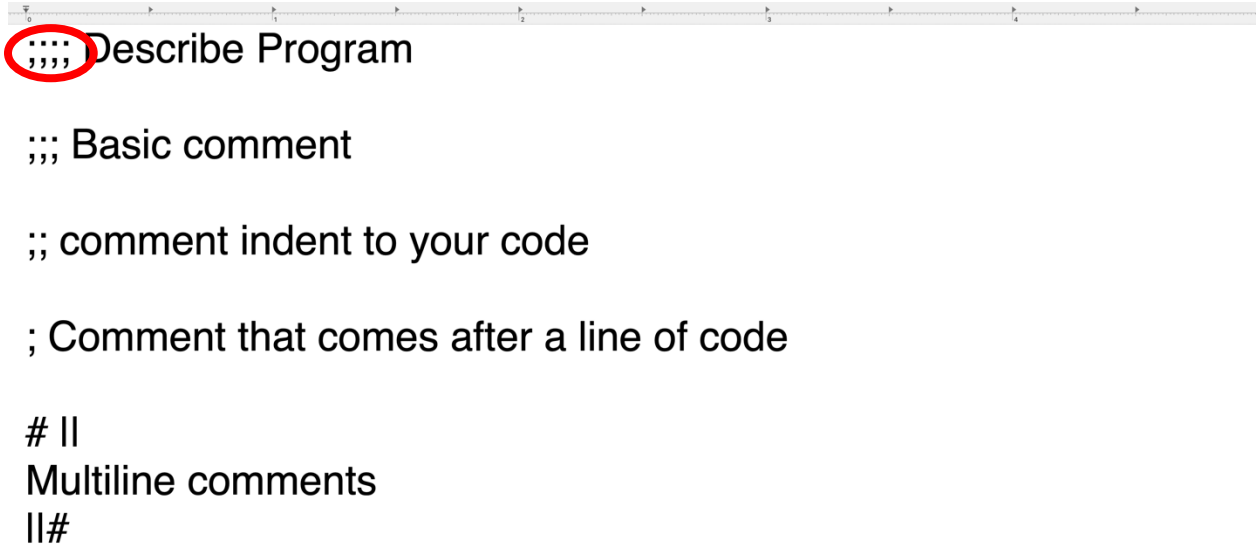
1. Print function



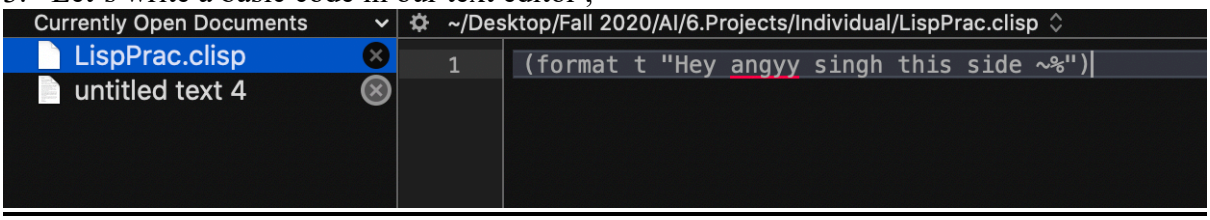
2. Quitting from the interface



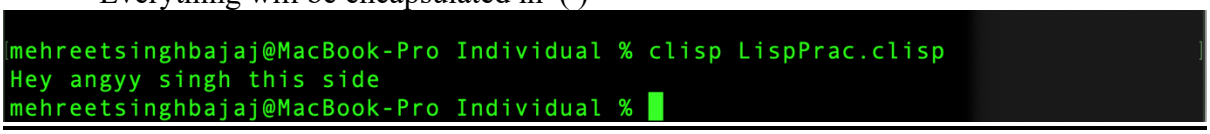
So , lets start writing our programs in some text editor



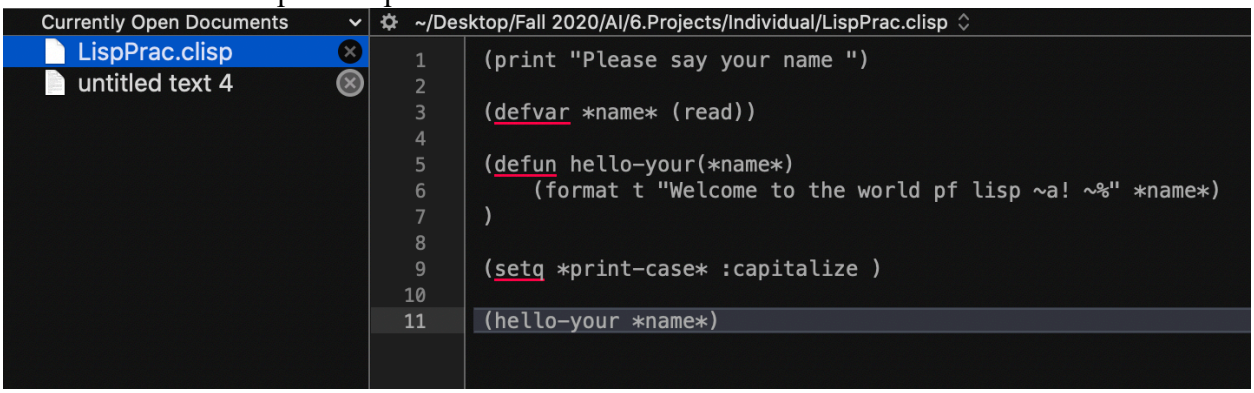
3. Let's write a basic code in our text editor ,

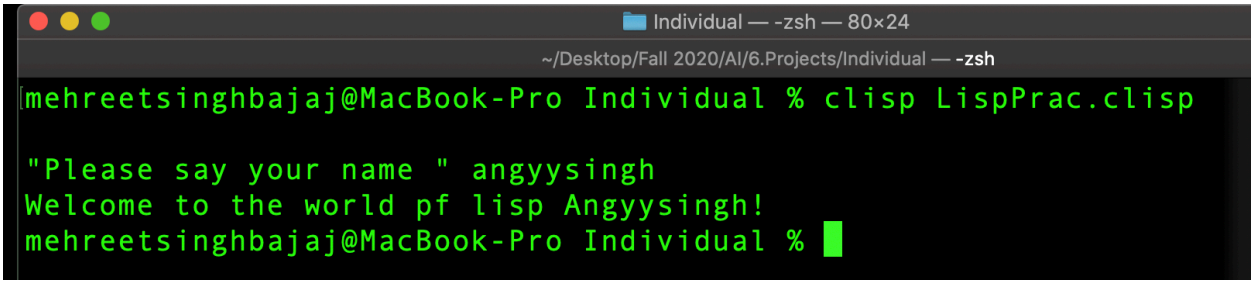


- Naming it LispPrac.clisp → extension is .clisp as we will run this on our terminal
- Everything will be encapsulated in ( )



4. Read and write input / output

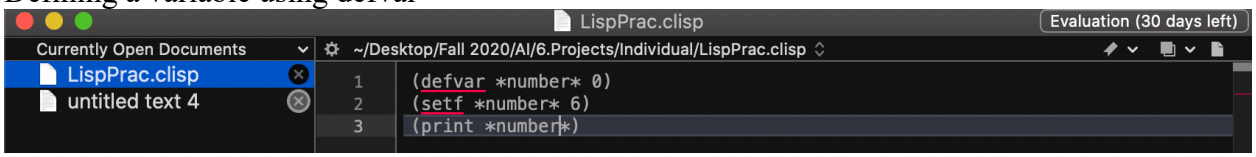




Defining a variable using defvar
Definig a function using defun
format -t helps printing info on terminal

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

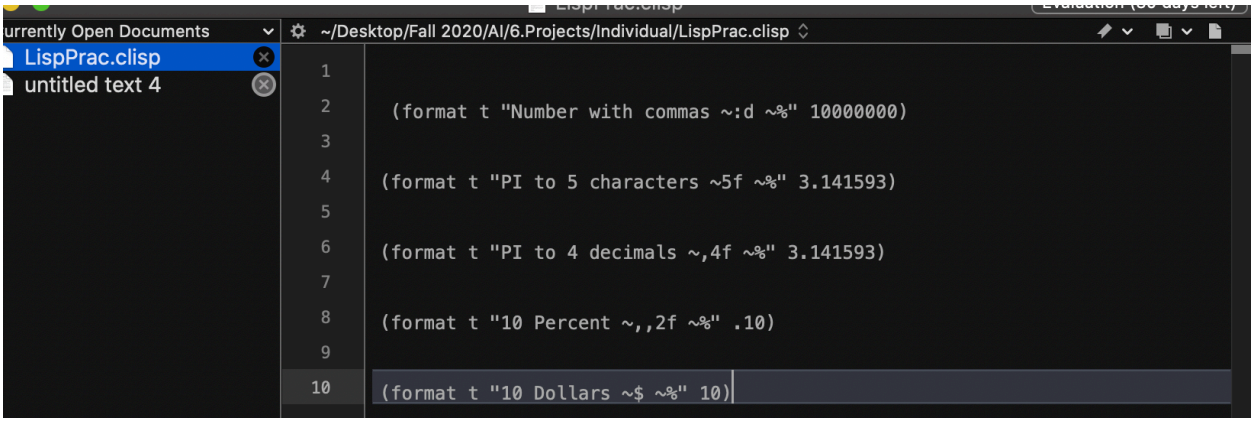
5. Defining a variable using defvar



Setf allows us to change the value of variable

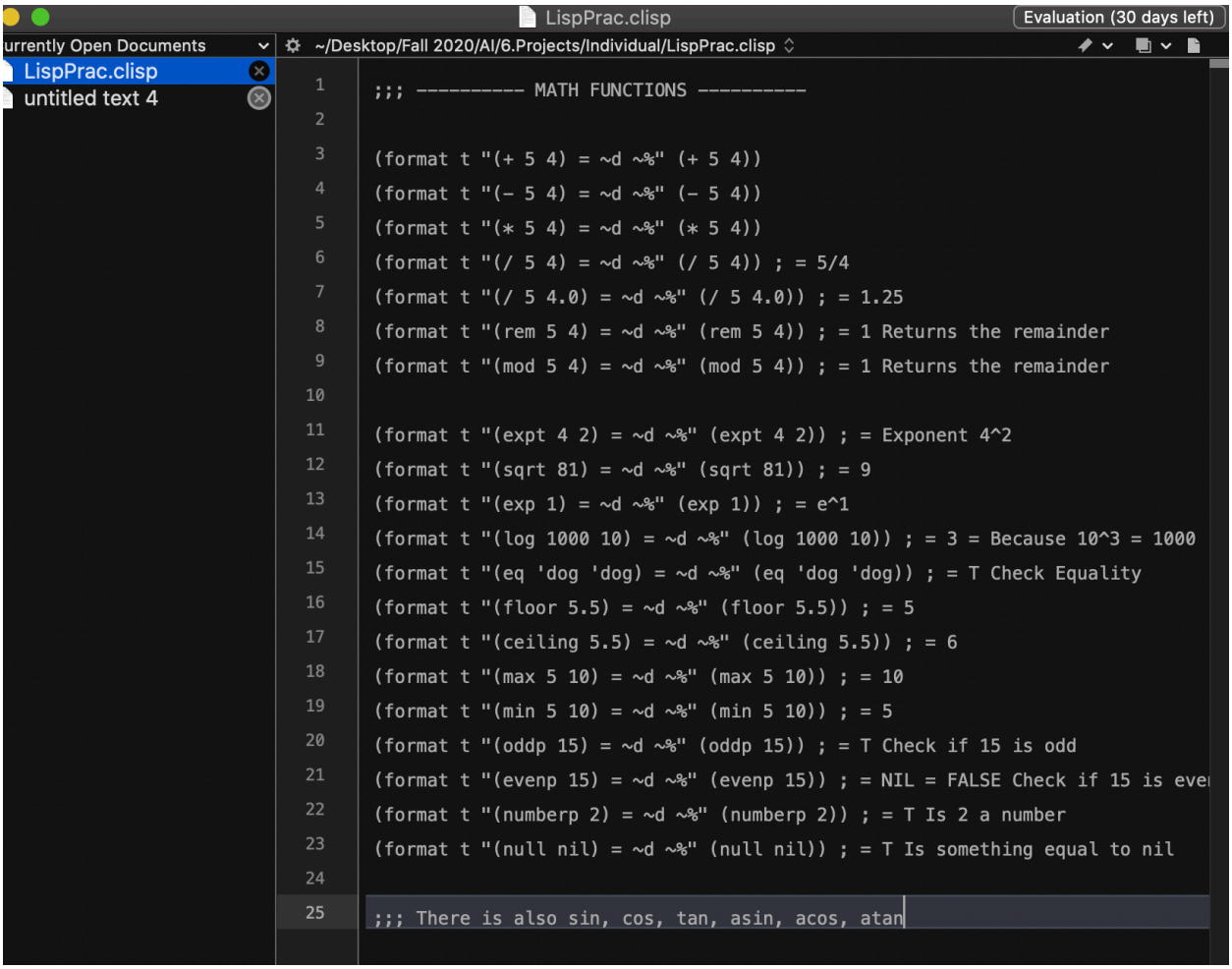
```
mehreetsinghbajaj@MacBook-Pro Individual % clisp LispPrac.clisp

6
mehreetsinghbajaj@MacBook-Pro Individual %
```

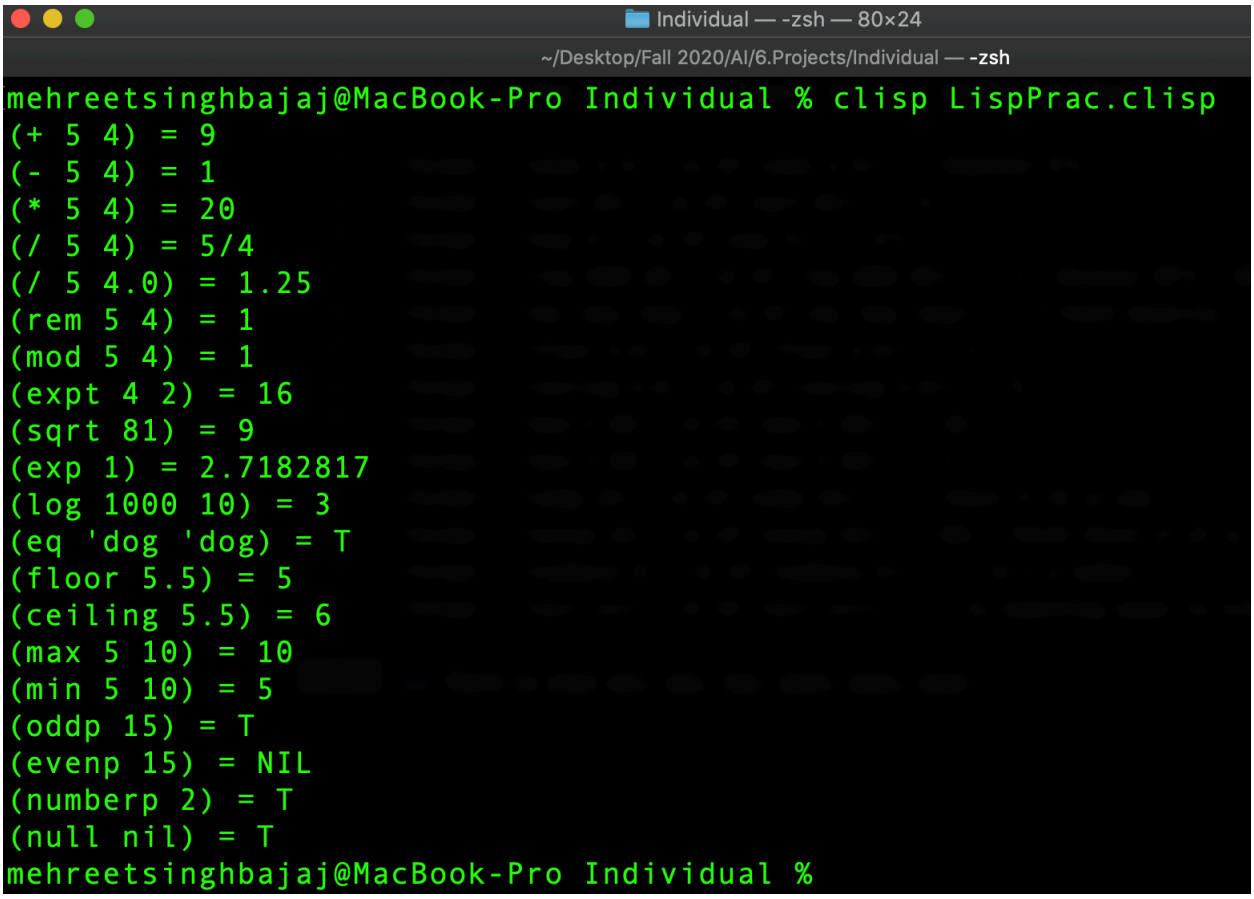
6. Playing with factor method



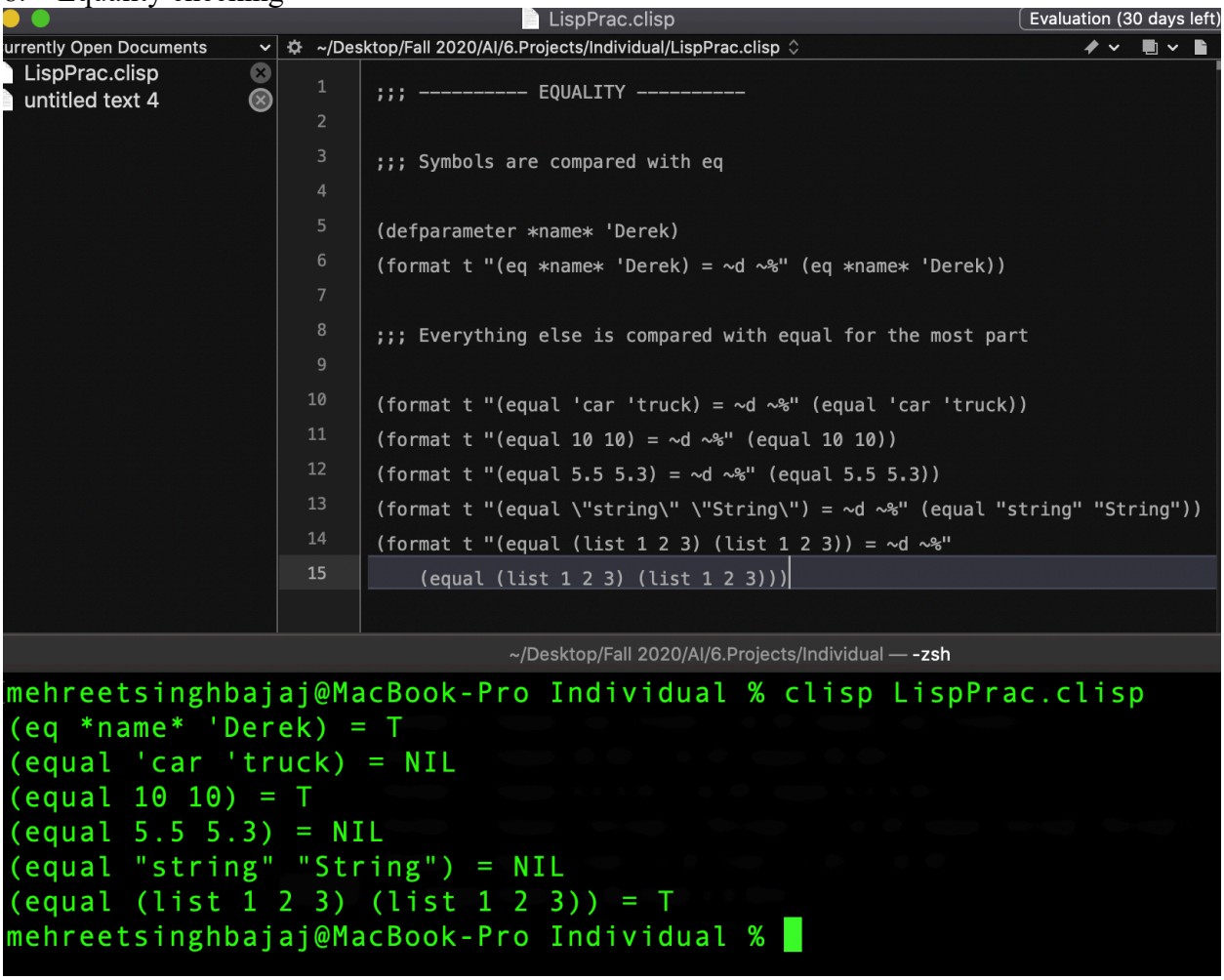

```
mehreetsinghbajaj@MacBook-Pro Individual % clisp LispPrac.clisp
Number with commas 10,000,000
PI to 5 characters 3.142
PI to 4 decimals 3.1416
10 Percent 10.0
10 Dollars 10.00
mehreetsinghbajaj@MacBook-Pro Individual %
```

7. Mathematical Functions

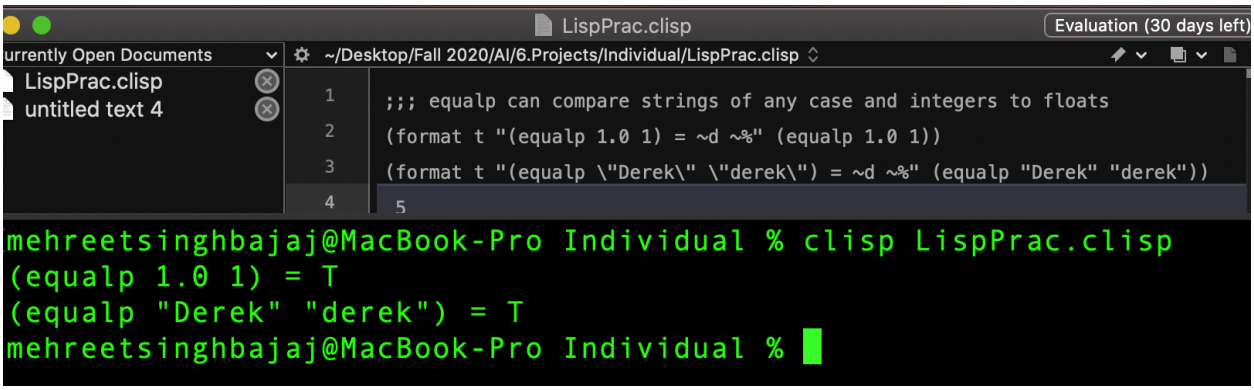



Individual — -zsh — 80×24
~/Desktop/Fall 2020/AI/6.Projects/Individual — -zsh
mehreetsinghbajaj@MacBook-Pro Individual % clisp LispPrac.clisp
(+ 5 4) = 9
(- 5 4) = 1
(* 5 4) = 20
(/ 5 4) = 5/4
(/ 5 4.0) = 1.25
(rem 5 4) = 1
(mod 5 4) = 1
(expt 4 2) = 16
(sqrt 81) = 9
(exp 1) = 2.7182817
(log 1000 10) = 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



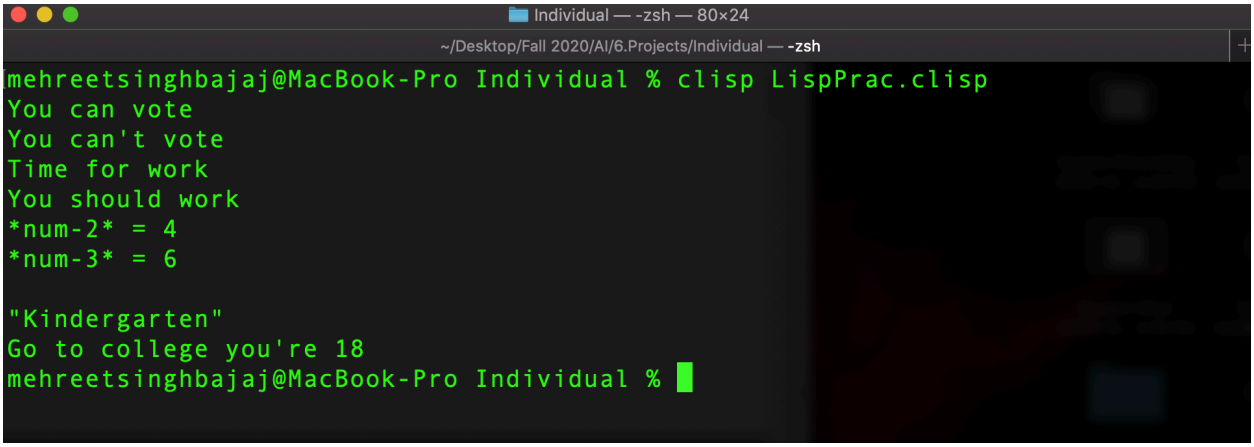
9. Comparing Strings



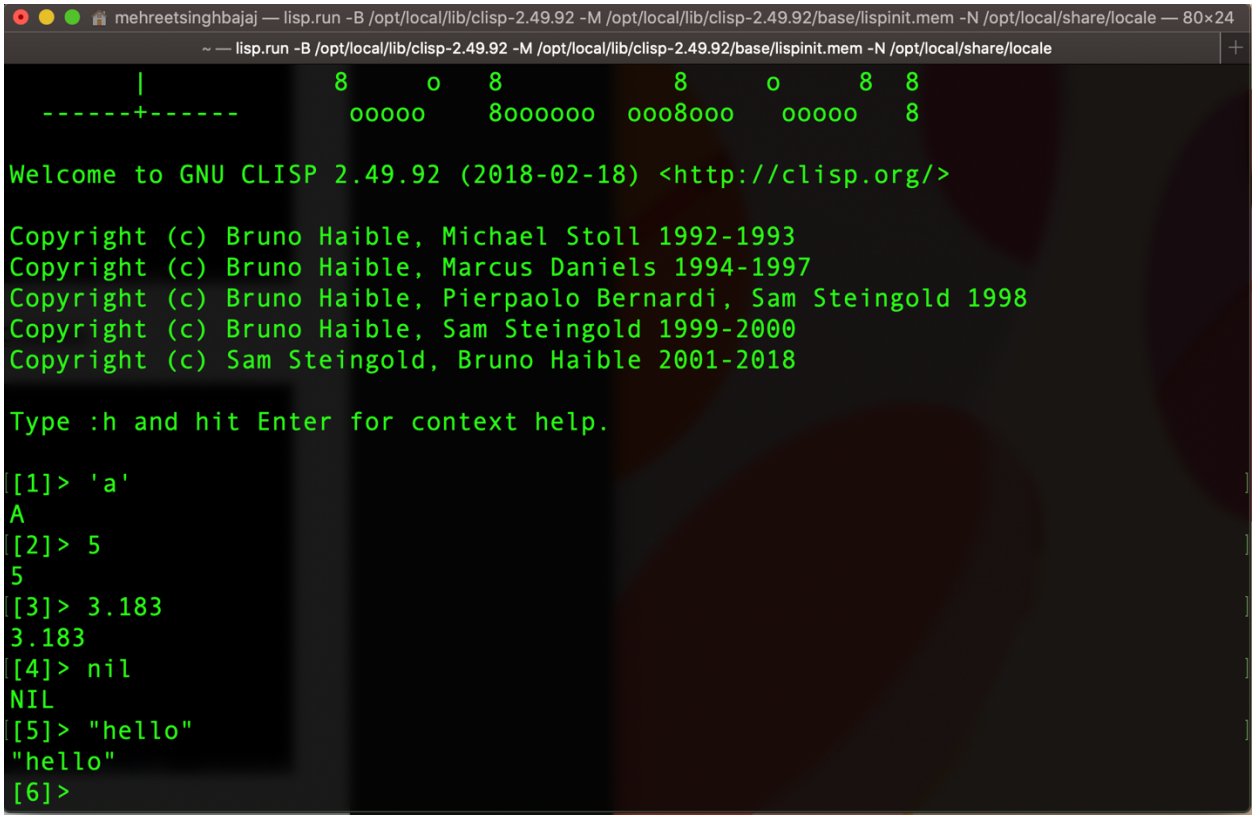
10. .Conditionals, not equal , logical operators, multiple statements using if


```
;; ---------- CONDITIONALS ----------

(defparameter *age* 18) ; Create variable age

;;; Relational Operators > < >= <= =

;;; Check if age is greater than or equal to 18

(if (= *age* 18)
(format t "You can vote~%")
(format t "You can't vote~%"))

;;; How to check for not equal

(if (not (= *age* 18))
(format t "You can vote~%")
(format t "You can't vote~%"))

;;; Logical Operators : and, or, not

(if (and (>= *age* 18) (<= *age* 67) )
(format t "Time for work~%")
(format t "Work if you want~%"))

(if (or (<= *age* 14) (>= *age* 67) )
(format t "You shouldn't work~%")
(format t "You should work~%"))

(defparameter *num* 2)
(defparameter *num-2* 2)
(defparameter *num-3* 2)

;;; You can execute multiple statements in an if with progn

(if (= *num* 2)
    (progn
        (setf *num-2* (* *num-2* 2))
        (setf *num-3* (* *num-3* 3))
    )
    (format t "Not equal to 2~%"))

(format t "*num-2* = ~d ~%" *num-2*)
(format t "*num-3* = ~d ~%" *num-3*)
```

```
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  )
```

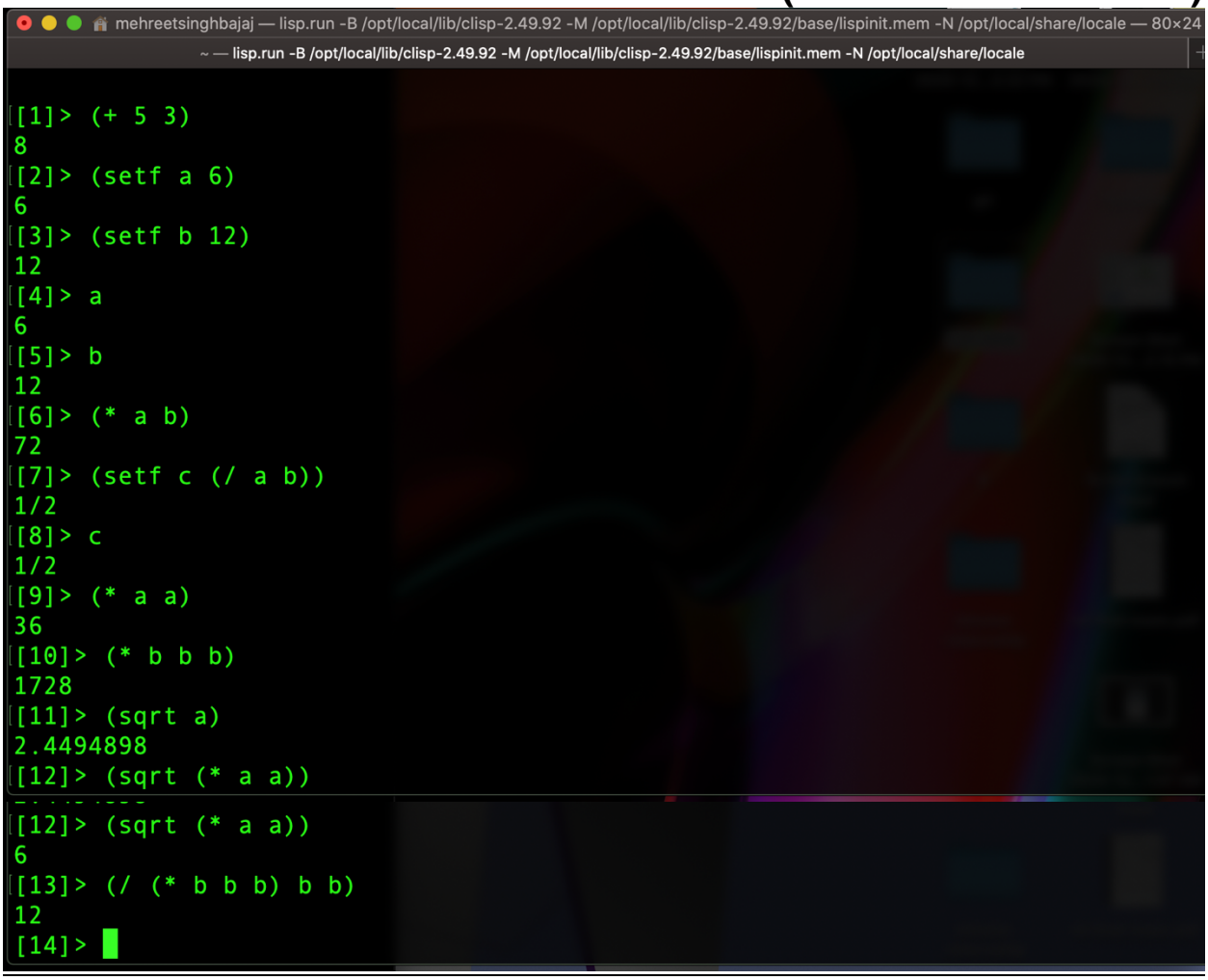


❖ EXERCISE – 1: DRIBBLE SESSION A



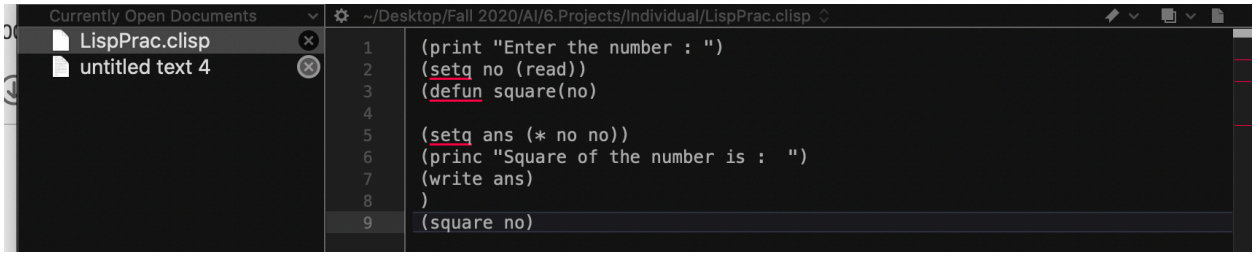
- Write the normal char and they will be printed in capital
- Input the numerical and it will echo
- Input the information you want to print in between “ ” and it will print it like that

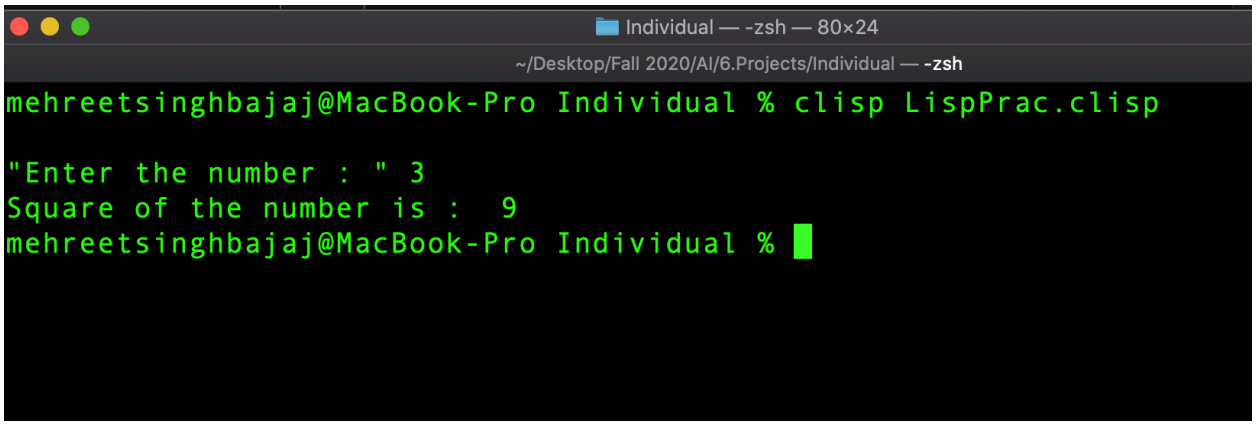
❖ EXERCISE – 1: DRIBBLE SESSION B (BUILT IN FUNCTION)

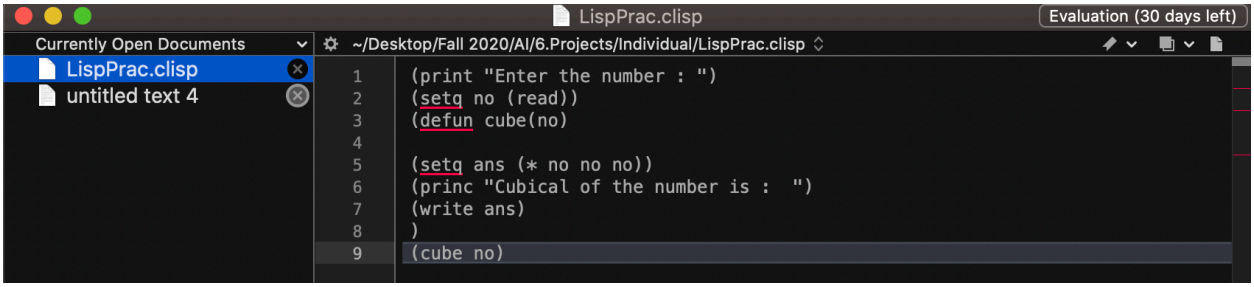


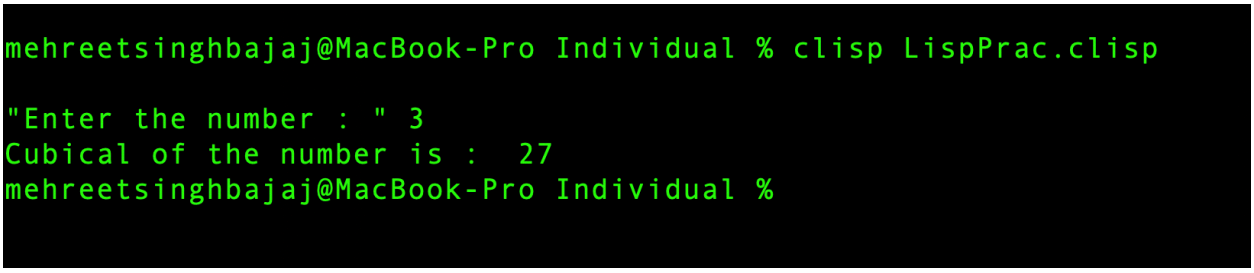
- **(operator values)**
- **Setf- sets the values of function (setf variable value)**
- **Nested operator – (sqrt (operator var var))**

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









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

LispPrac.clisp — Evaluation (30 days left)

~/Desktop/Fall 2020/AI/6.Projects/Individual/LispPrac.clisp

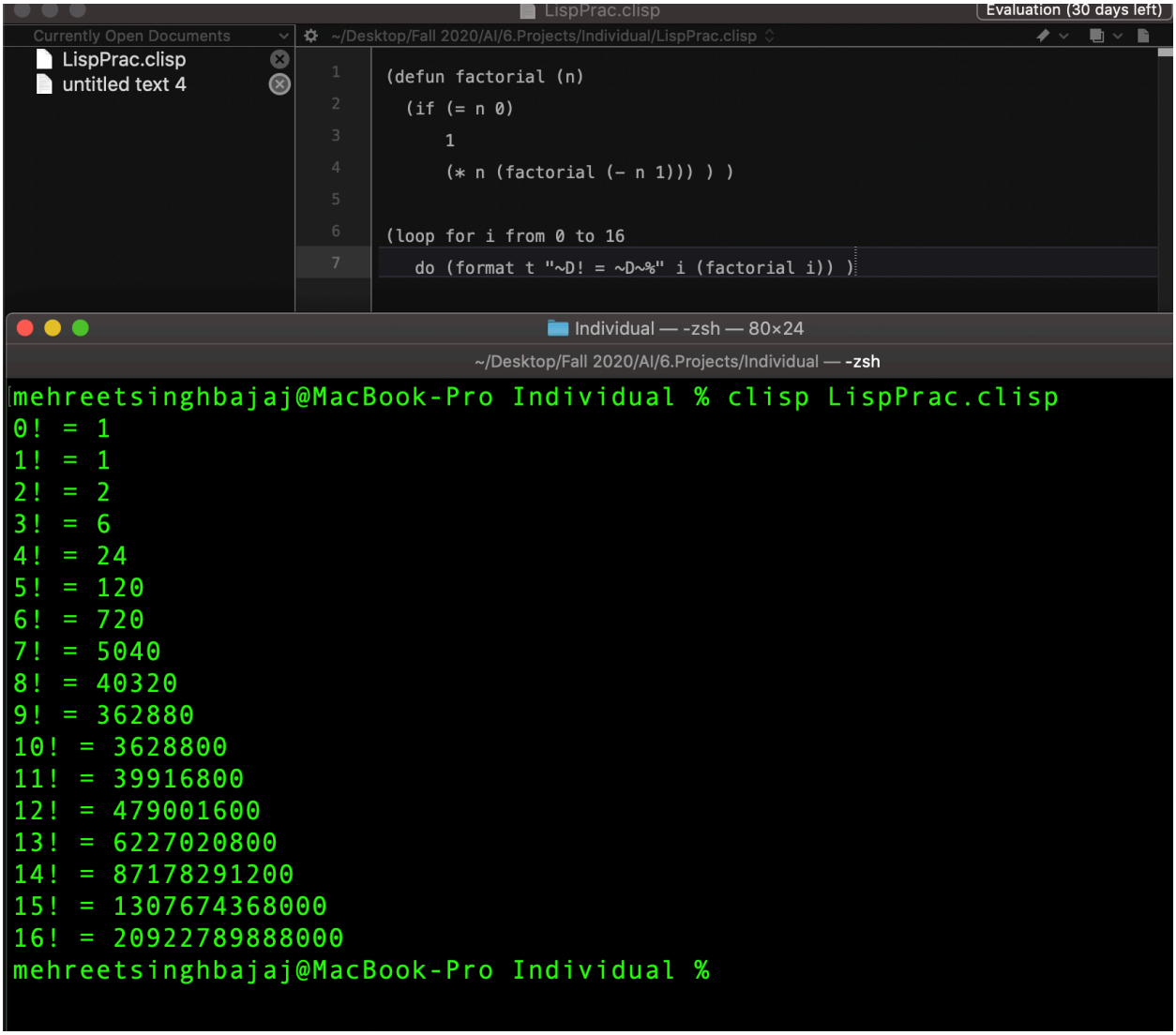
```
1  (write (cons 1 2))
2  (terpri)
3  (write (cons 'a 'b))
4  (terpri)
5  (write (cons 1 nil ))
6  (terpri)
7  (write (cons 2 nil ))
8  (terpri)
9  (write (cons 2 (cons 3 nil)))
10 (terpri)
11 (write (cons 1 (cons 2 (cons 3 nil))))
12 (terpri)
13 (write (cons 'a(cons 'b (cons 'c nil))))
14 (terpri)
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 — 80×24

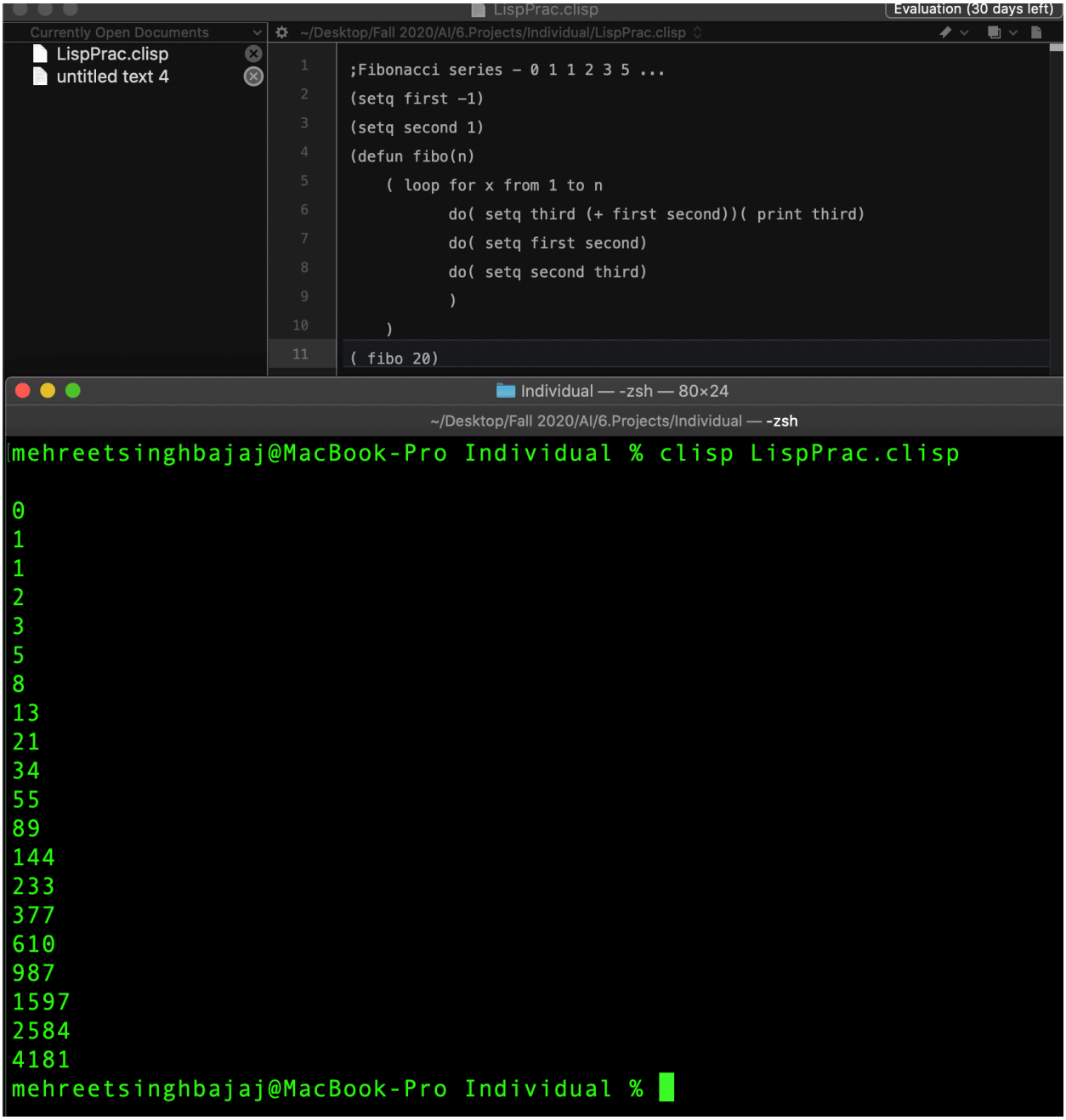
~/Desktop/Fall 2020/AI/6.Projects/Individual — -zsh

```
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 %
```

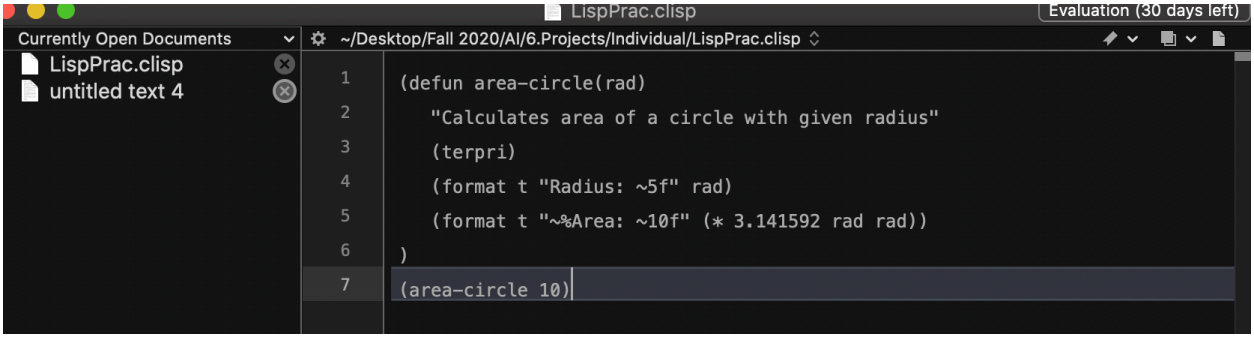
❖ EXERCISE – 2: A. Factorial

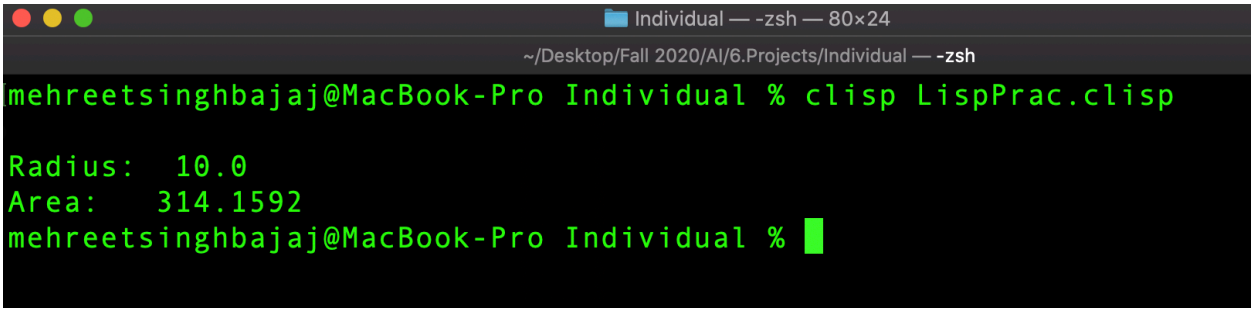


❖ EXERCISE – 2: B. Fibonacci



❖ EXERCISE – 2: C. Area of circle





❖ EXERCISE – 2: D. User Input Output