

## Computer Science I

Professor Tom Ellman  
Lecture 1

## Goals of this Course

- Teach key concepts of Computer Science.
  - That are relevant to all computational problems.
  - Regardless of the machine, operating system or programming language.
- Provide hands on experience writing programs in Scheme, a simple yet powerful programming language.

## Why Scheme?

- Scheme is a dialect of LISP, the second oldest programming language.
- Scheme has a simple syntax that allows us to focus on using the language, rather than the language itself.
- Scheme is used at many colleges and universities: MIT, Stanford, Princeton, Swarthmore ... etc.

## The course does not cover:

- Word processing, spread sheets, email, creating web pages or using web browsers.
- Specific brands of computers, operating systems or languages (any more than necessary).

## Enrolling in this Class

- The course is very popular!
- New people to be admitted if space is available.
- Check with professor at end of second meeting.

## Class Web Page

<http://www.cs.vassar.edu/~cs101/>

- Overview of the course.
- Schedule of topics and readings.
- Lab and homework assignments.
- Summary of grading policy.
- Professor Ellman's lecture notes.

## Contacting Professor Ellman

- Office: 118 OLB
- Phone: 437-5991
- Email: ellman@cs.vassar.edu

## Office Hours

- Tuesdays 2:00pm - 3:00pm.
- Thursdays 2:00pm - 3:00pm.
- Or send email to make an appointment.
- Or just stop by my office.

## Textbooks

- “Scheme and the Art of Programming”, Springer and Friedman, MIT Press, 1989.
- “The Limits of Computing”, Walker, Jones and Bartlett, 1994.

## Requirements and Grading

- Programming Assignments: 30%
- Labs 10%.
- Essay 10%.
- First Midterm: 15%
- Second Midterm: 15%
- Final Exam: 20%

## Labs

- One lab each week.
- Sign up for lab times next week.
- Coaches assist with lab work.
- Coaches grade lab work.

## Homework

- One programming assignment each week.
- Typically consisting of three or four exercises.
- Absolutely essential for learning the course material.
- Try not to fall behind, since it can be difficult to catch up.

## Essay

- Students will be required to read selections from “The Limits of Computation”.
- Students will be required to write a five-page paper based on these readings.
- The reading selections paper topic will be assigned later in the semester.
- The paper will be due at the end of the semester, on a date to be announced.

## Deadlines and Lateness

- Assignments are due at the start of class on the date specified.
- Late assignments will be accepted with 10% penalty, but only until the start of the next class.

## Examinations

- Open book and open notes.
- Cumulative, from start of the semester.
- Not scheduled yet.

## Academic Integrity

- You may discuss general ideas with classmates.
- You must do each programming assignment entirely by yourself.
- You may not discuss or share programs with other students.
- Vassar regulations require the professor to report suspected violations of academic integrity to the Dean of Studies.
- Read the “Originality and Attribution” pamphlet.

## Computational Process

- Computational problem.
- Algorithm for solving the problem.
- Implemented in a program and data.
- Running on a physical machine.

## Programming as an Intellectual Activity

- Problem formulation.
- Problem decomposition.
- Data representation.
- Procedure implementation.
- Testing.
- Debugging.
- Documentation.

## Managing Complexity

- Abstractions that hide information.
- Common patterns for combining programs and data into larger programs and data.
- Interfaces that facilitate mixing and matching modules.
- Specialized languages that provide mental leverage.

## What makes a program good?

- It gives the right answer. (Correctness)
- It runs fast. (Speed)
- It can be used over and over. (Generality)
- It was easy to write. (Simplicity)
- It is easy to read. (Clarity)

## Programming as a Craft

- A means of expression.
- Developed through practice.
- Analogous to writing ability.

## How to Start DrScheme:

- Walk up to the computer.
- Turn it on.
- Grab the mouse.
- Point at the DrScheme icon.
- Click.

## Advice:

- Ordinary humans are not born knowing how to use computers.
- Don't be embarrassed if you have to ask how to do something.

## Scheme does one thing, over and over:

- Read an expression typed in by the user.
- Evaluate the expression to obtain a value.
- Display the value of the expression.

“Read - Eval - Print Loop”

## Read - Eval - Print Example

Welcome to DrScheme

>

> 1

1

> 2

2

Scheme displays ">", called the "prompt".

User types the number "1" and hits the "Return" key.

Scheme evaluates the expression "1" and displays its value.

## Read - Eval - Print Example

Welcome to DrScheme

> (+ 8 7)

15

> (- 15 8)

7

User types a more complex expression and hits the "Return" key.

Scheme evaluates the expression and displays its value.

## Kinds of Expressions in Scheme

- Primitive expressions:
  - Constants, such as numbers: "1", "2", "3", ... etc.
  - Variables, such as "name", "age", "sex", ..., etc.
- Composite expressions: Expressions that are composed of other, smaller expressions.
  - Example: "(+ 8 7)" which means 8 + 7.
  - Example: "(- 15 8)" which means 15 - 8.

## Examples of evaluating composite expressions:

Welcome to DrScheme

> (+ 8 7)

15

> (- 15 8)

7

## Examples of evaluating composite expressions:

Welcome to DrScheme

> (\* 8 7)

56

> (/ 56 8)

7

## Translating "Infix" Expressions into "Prefix" Expressions

8 + 7

↓

+ 8 7

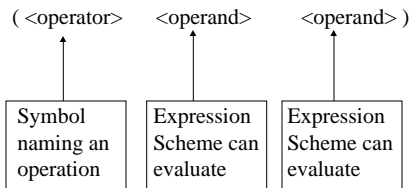
↓

( + 8 7 )

Move "+" in front of "8" and "7".

Add parentheses.

## Syntax of Composite Expressions

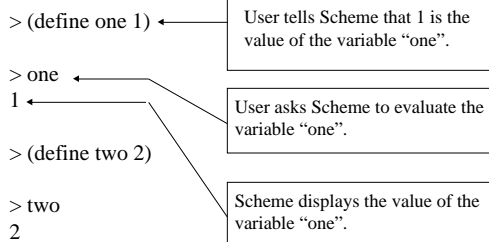


## Definition of a Variable

- A special kind of composite expression.
- Processed in a special way by Scheme.

## Examples of Defining Variables

Welcome to DrScheme



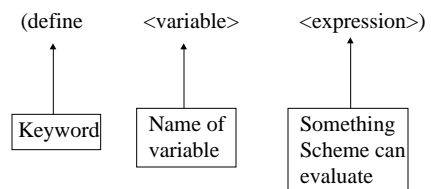
Variable names are arbitrary. The following is silly, but legal :

```
Welcome to DrScheme
> (define five 6)
> five
6
> (define six 5)
> six
5
```

Several variables may have the same value:

```
Welcome to DrScheme
> (define seven 7)
> seven
7
> (define sept 7)
> sept
7
```

## Syntax of Variable Definitions



Once a variable is defined, it may be used in any expression:

```
Welcome to DrScheme  
> (define year 2000)  
> (+ year 4)  
2004
```

Any expression may be used in the definition of a variable:

```
Welcome to DrScheme  
> (define graduation (+ year 4))  
> graduation  
2004
```

Expressions may be used inside larger expressions:

```
Welcome to DrScheme  
> (* (+ 8 7) 2)  
30  
> (* (* 2 4) (* 5 3))  
120  
> (+ (+ (+ 3 3) 3) 3)  
12
```

Steps in evaluating complex expressions:

```
(* (+ 8 7) 2)  
↓  
(* 15 2)  
↓  
30
```

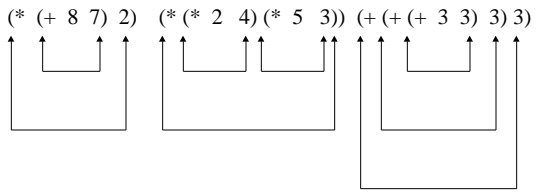
Steps in evaluating complex expressions:

```
(* (* 2 4) (* 5 3))  
↓  
(* 8 (* 5 3))  
↓  
(* 8 15)  
↓  
120
```

Steps in evaluating complex expressions:

```
(+ (+ (+ 3 3) 3) 3)  
↓  
(+ (+ 6 3) 3)  
↓  
(+ 9 3)  
↓  
12
```

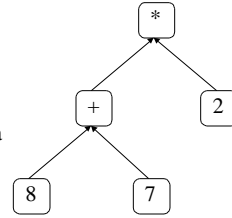
### Parentheses must be balanced:



### Interpreting Expressions as Trees

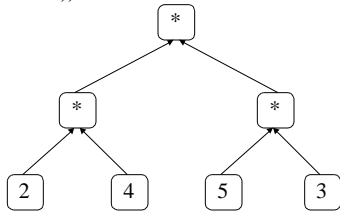
$(* (+ 8 7) 2)$

Arrows indicate direction of data flow.



### Interpreting Expressions as Trees

$(* (* 2 4) (* 5 3))$



### Interpreting Expressions as Trees

$(+ (+ (+ 3 3) 3) 3)$

