CSSE 304 – Winter 2019-20
Programming Language Concepts

Pick up three handouts from the back table

http://www.rose-hulman.edu/class/csse/csse304/

https://plc.csse.rose-hulman.edu/

https://piazza.com/rose-hulman/winter2020/csse304/

https://www.rose-hulman.edu/class/csse/csse304/202020/announcements.html

Claude Anderson
Professor of Computer Science and Software Engineering
F-210 812-877-8331

Typical office hours MTRF 1:00-3:00 (plus other hours many days)

My weekly schedule is also linked from the course Schedule Page

anderson@rose-hulman.edu
csse304-staff@rose-hulman.edu

TAs: Alysa Crawford, Qiuyun Li, Valerie Liu, Fisher Shen, Quinn McKown, Ben Goldstein, Mitchell Schmidt, Jessica Myers, Duncan McKee

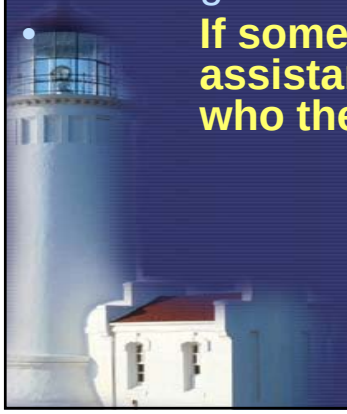
Joanna Garrett and Tommy McMichen– PercopoTutors for the Learning Center

CSSE 304 Day 1

- Pass the attendance sheet (daily)
  - **Today's sheet**: tell me what name you'd like to be called by me and by other students.
- Brief Announcements
- What's on the web?
- Scheme Intro (assumes that you viewed the five videos)
- Instructor/course intro: **will happen Day 3**
  - because we will dive into Scheme for the first two class days.
- **Scheme-a-thon**: Days 1-12.

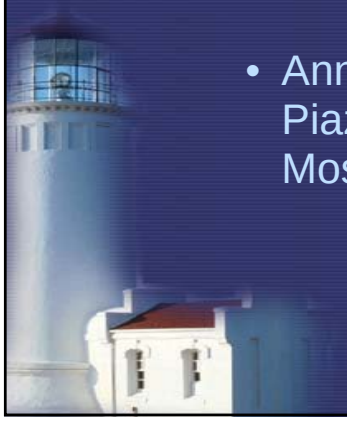
Announcements

- TA lab hours begin today.
- Student assistants can give you help on installing Scheme and/or editors, getting started with Scheme programming.
- Please give me feedback on how the assistants are doing.
- **If something else is going on in F-217, the assistants may be in F-225. If you don't know who the assistant is, be bold and ask.**



Daily Course Announcements

- Each Day's announcements will be in the **announcements.html** document, linked from the Resources column of the Schedule page.
- Announcements In-between class times:
  Piazza or email
  Mostly Piazza



Scheme-athon begins!

- It continues through Friday, Dec 20

If you are a member of NPS, plan to suspend your membership for this term!

From today's class notes

- This is from *The Scheme Programming Language* "Summary of Forms"

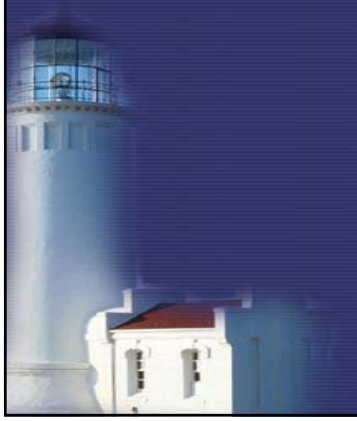
`(integer->char int)`	procedure	158
`(integer? obj)`	procedure	130
`(interaction-environment)`	procedure	117
`(lambda formals` exp_1 exp_2 `...)`	syntax	86
`(lcm int ...)`	procedure	147
`(length list)`	procedure	135
`(let ((var val) ...)` exp_1 exp_2 `...)`	syntax	87

- What's the difference between "procedure" and "syntax"?

From today's class notes

- Talk with the other student(s) at your table.
- What will Scheme print when we enter each of these?

```
(/ 3 5)
(cadr  '(a b c))
(cons 1 '(a b c))
(cons 1 2)
(eq? '(a b c) '(a b c))
(let ([a 5][b 6])
   (+ 3 a b))
```

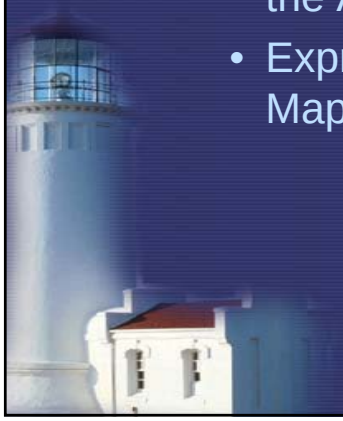


Overview of Scheme

- The next six slides give an overview of the Scheme language.
- Good stuff! Pretty self-explanatory.
- Duplicated on your class notes hand-out.
- Read it before tomorrow's class; if you have questions, ask them in tomorrow's class.
- Now we are going to jump right into more Scheme exploration.

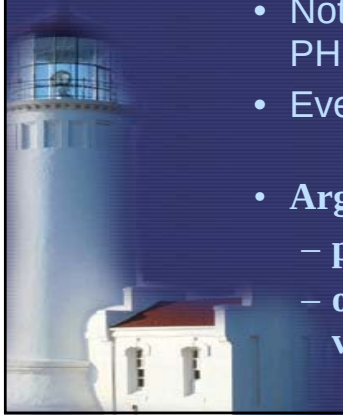
Overview of Scheme 1

- Invented in 1975 by Guy Steele and Gerald Sussman at MIT.
- Syntax similar to LISP, semantics more like the Algol family (Pascal, Ada, C, Java …)
- Expression-oriented and interactive (like Maple, Python, MatLab).



Overview of Scheme 2

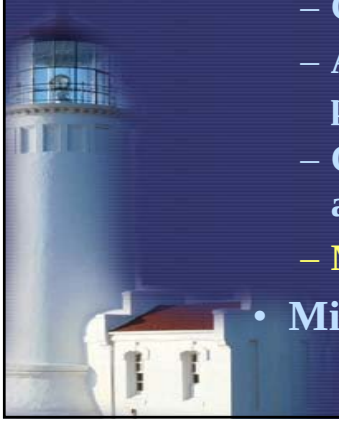
- Data and programs have the same syntax http://xkcd.com/859/
- (Linked) **lists** are a fundamental, built-in data type
- Not statically typed (similar to Python, Maple, PHP, JavaScript; unlike C and Java)
- Everything is in prefix form:
  `(+ a b)` instead of `a + b`
- **Argument passing is similar to Java**
  - **primitives are passed to procedures by value**
  - **others (lists, vectors*, etc.): references passed by value.**



*** vector** is Scheme's array type. Similar to Java arrays

Overview of Scheme 3

- Symbols can be data
  - **Somewhat similar to Maple**
- **Procedures are (first-class) data**
  - **Can pass them as arguments to other procedures**
  - **A procedure can create and return another procedure**
  - **Can store a procedure in a data structure (such as a list or array of procedures)**
  - **More on this soon!**
- **Minimal procedure overloading.**



Overview of Scheme 4

- Java's ***new*** operator has no Scheme equivalent
  - **Instead there are specific procedures for creating objects of each type:**
    - ***cons*** **creates a pair (*pair* is Scheme's main data type)**
    - ***vector*** **creates a vector (like a C or Java array)**
    - ***list*** **creates a (proper) list of its arguments**
    - ***string*** **creates a string from zero or more characters**
    - ***lambda*** **(which is not a procedure*) creates a procedure**
    - ***define-syntax*** **(also not a procedure) creates new syntax that extends the Scheme language itself**

*In the Summary of Forms at the end of TSPL, **"not a procedure"** is denoted by **"syntax"**

http://***scheme.com/tspl4/summary.html***

Scheme data types and constants

- Numbers
  - `6  -12  142839178509230947676264 56`
  - `5/17  (+ 1/3 1/6)  (max 5 7 3)`
  - `7.05  3.5e7  (+ 2e3 3e2)`
  - TSPL section 6.3 lists the available operations on numbers.
- Boolean (note that `if` returns a value)
  - `#t  #f  (if (< a b) a (+ b 1))`
- String
  - `"This is a String"`
    `(string-length "Hello")`

Data types and constants 2

- Character
  - `#\A  #\newline  (char->integer #\A)`
- Symbol `(quote hello)  'hello`
- Vector (like an array in other languages)
  - `#(1 3 2)  (make-vector 5 7)`
    `(vector-ref v 4)`
- Empty List `( )`
- Pair `(3 . 5)`
- List of three elements: `(3 5 7)`
- Improper list `(3 5 7 . 8)`
- List of lists: `((2 4) (5 6 7) (8) ())`

Draw pictures

Live demo

Scheme source file-name extensions

– .ss , .scm **I'll use .ss**

- I'll demonstrate SWL and the Emacs editor.
- In-class coding will be in the Live-in-class folder, linked from Resources column of Day 1 of the schedule page.
- We will continue this demo/discussion tomorrow.
- **You can follow along on your computer, or you can just watch, think, and ask questions.**