



CSE341: Programming Languages

Lecture 1 Course Mechanics ML Variable Bindings

Brett Wortzman
Summer 2019

Slides originally created by Dan Grossman

1

Welcome!

We have ~~40~~ 8.5 weeks to learn *the fundamental concepts* of programming languages

With hard work, patience, and an open mind, this course makes you a much better programmer

- Even in languages we won't use
- Learn the core ideas around which every language is built, despite countless surface-level differences and variations
- *Poor* course summary: "Learn ML, Racket, and Ruby"

Today's class:

- Course mechanics
- [A rain-check on motivation]
- Dive into ML: Homework 1 due next week

Summer 2019

CSE341: Programming Languages

2

Concise to-do list

In the next 24-48 hours:

1. Read course web page:
<http://courses.cs.washington.edu/courses/cse341/19su/>
2. Read all course policies (4 short documents on web page)
3. Adjust class email-list settings as necessary
4. Complete Homework 0 (survey worth 0 points)
5. Get set up using Emacs [optional; recommended] and ML
 - Installation/configuration/use instructions on web page
 - Essential; non-intellectual
 - No reason to delay!

Summer 2019

CSE341: Programming Languages

3

Who: Course Staff

Instructor - Brett Wortzman:

Teaching faculty
First time teaching 341
Wrote undergraduate thesis on similar topics

Two *great* TAs:

Brendan Murphy
Svetoslav Kolev

Get to know us!

Summer 2019

CSE341: Programming Languages

4

Staying in touch

- Course email list: cse341a_su19@u.washington.edu
 - Students and staff already subscribed
 - You must get announcements sent there
 - Fairly low traffic
- Message Board (Piazza)
 - For appropriate discussions; TAs will monitor
 - Highly encouraged, but won't use for important announcements
- Anonymous feedback link on webpage
 - For good and bad: If you don't tell us, we don't know

Summer 2019

CSE341: Programming Languages

5

Lecture: Brett

- Slides, code, and reading notes / videos posted
 - May be revised after class
 - *Take notes*: materials may not describe everything
 - Slides in particular are *visual aids* for me to use
- Ask questions, focus on key ideas
- Engage actively
 - Arrive *punctually* (beginning matters most!) and well-rested
 - Just like you will for the midterm!
 - *Write* down ideas and code as we go
 - If attending and paying attention is a poor use of your time, one of us is doing something wrong

Summer 2019

CSE341: Programming Languages

6

Section

- Required: will usually cover new material
- Sometimes more language or environment details
- Sometimes main ideas needed for homework
- *Will* meet this week: need to get ahead

Material often also covered in reading notes / videos

Summer 2019

CSE341: Programming Languages

7

7

Reading Notes and Videos

- Posted for each "course unit"
 - Go over most (all?) of the material (and some extra stuff?)
- So why come to class?
 - Materials let us make class-time much more useful and interactive
 - Answer questions without being rushed because *occasionally* "didn't get to X; read/watch about it"
 - Can point to optional topics/videos
 - Can try different things in class, not just recite things
- Don't need other textbooks – Dan has roughly made one himself

Summer 2019

CSE341: Programming Languages

8

8

Office hours

- Regular hours and locations TBD
 - Will be posted on course website
 - Changes as necessary announced on email list and posted on website
- Use them
 - *Please visit me*
 - Ideally *not just* for homework questions (but that's OK too)

Summer 2019

CSE341: Programming Languages

9

9

Homework

- Seven* total
- To be done individually
- Doing the homework involves:
 1. Understanding the concepts being addressed
 2. Writing code demonstrating understanding of the concepts
 3. Testing your code to ensure you understand and have correct programs
 4. "Playing around" with variations, incorrect answers, etc. Generally, only (2) is graded, but focusing on only (2) makes homework harder
- Challenge problems: Low points/difficulty ratio

Summer 2019

CSE341: Programming Languages

10

10

Note Dan's writing style

- Homeworks tend to be worded very precisely and concisely
 - I write like a computer scientist (a good thing!)
 - Technical issues deserve precise technical writing
 - Conciseness values your time as a reader
 - You should try to be precise too
- *Skimming or not understanding why a word or phrase was chosen can make the homework harder*
- By all means ask if a problem is confusing
 - Being confused is normal and understandable
 - And I may have made a mistake

Summer 2019

CSE341: Programming Languages

11

11

Academic Integrity

- Read the course policy carefully
 - Clearly explains how you can and cannot get/provide help on homework and projects
- Always explain any unconventional action
- I have promoted and enforced academic integrity since I was a freshman
 - Great trust with little sympathy for violations
 - Honest work is the most important feature of a university
- This course especially: Do not web-search for homework solutions! We will check!

Summer 2019

CSE341: Programming Languages

12

12

Exams

- Midterm: Friday July 26, in class (tentative)
- Final: Friday, August 23, in class
- Same concepts, but different format from homework
 - More conceptual (but write code too)
 - Will post old exams
 - Closed book/notes
 - Note sheet policy TBA

Summer 2019

CSE341: Programming Languages

13

13

Coursera (more info in document)

- Dan has taught this material to thousands of people around the world
 - A lot of work and extremely rewarding
- You are not allowed to participate in that class!
- This should have little impact on you
 - Two courses are separate
 - 341 is a great class and staff is committed to this offering being the best ever
- But this is a neat thing you are likely curious about...

Summer 2019

CSE341: Programming Languages

14

14

More Coursera

- Why did we do a MOOC?
 - Have more impact (like a textbook) for Dan's favorite stuff!
 - Experiment with where higher-ed might be going
- So why pay tuition?
 - Personal attention from humans
 - Homeworks/exams with open-ended questions
 - Class will adjust as needed
 - We can be sure you actually learned
 - Course is part of a coherent curriculum
 - Beyond the classroom: job fairs, advisors, social, ...

Summer 2019

CSE341: Programming Languages

15

15

Has Coursera help/hurt 341?

- Biggest risks
 - Becomes easier to cheat – don't! (And we've changed things)
 - We become too resistant to change – hope not!
- Benefits too
 - More resources: videos, grading scripts, ...
 - Way fewer typos
 - Taking the "VIP version" of a more well-known course
 - Change the world to be more 341-friendly

Summer 2019

CSE341: Programming Languages

16

16

Questions?

Anything I forgot about course mechanics before we discuss, you know, programming languages?

Summer 2019

CSE341: Programming Languages

17

17

What this course is about

- Many essential concepts relevant in any programming language
 - And how these pieces fit together
- Use ML, Racket, and Ruby languages:
 - They let many of the concepts "shine"
 - Using multiple languages shows how the same concept can "look different" or actually be slightly different
 - In many ways simpler than Java
- Big focus on *functional programming*
 - Not using *mutation* (assignment statements) (!)
 - Using *first-class functions* (can't explain that yet)
 - But many other topics too

Summer 2019

CSE341: Programming Languages

18

18

Why learn this?

This is the "normal" place for course motivation

- Why learn this material?

But we don't have enough shared vocabulary/experience yet

- So 3-4 week delay on motivation for functional programming
- I promise full motivation: delay is worth it
- (Will motivate immutable data at end of "Unit 1")

Summer 2019

CSE341: Programming Languages

19

19

Our claim

Learning to think about software in this "PL" way will make you a better programmer even if/when you go back to old ways

It will also give you the mental tools and experience you need for a lifetime of confidently picking up new languages and ideas

[Somewhat in the style of *The Karate Kid* movies (1984, 2010)]



Summer 2019

CSE341: Programming Languages

20

20

A strange environment

- Next 4-5 weeks will use
 - ML language
 - Emacs editor
 - Read-eval-print-loop (REPL) for evaluating programs
- Need to get things installed and configured
 - Either in the Allen School labs or your own machine
 - We've written thorough instructions (questions welcome)
- Only then can you focus on the content of Homework 1
- Working in strange environments is a CSE life skill

Summer 2019

CSE341: Programming Languages

21

21

Mindset

- "Let go" of all programming languages you already know
- For now, treat ML as a "totally new thing"
 - Time later to compare/contrast to what you know
 - For now, "oh that seems kind of like this thing in [Java]" will confuse you, slow you down, and you will learn less
- Start from a blank file...

Summer 2019

CSE341: Programming Languages

22

22

A very simple ML program

[The same program we just wrote in Emacs; here for convenience if reviewing the slides]

```
(* My first ML program *)
val x = 34;
val y = 17;
val z = (x + y) + (y + 2);
val q = z + 1;
val abs_of_z = if z < 0 then 0 - z else z;
val abs_of_z_simpler = abs z
```

Summer 2019

CSE341: Programming Languages

23

23

A variable binding

```
val z = (x + y) + (y + 2); (* comment *)
```

More generally:

```
val x = e;
```

- *Syntax:*
 - Keyword `val` and punctuation `=` and `;`
 - Variable `x`
 - Expression `e`
 - Many forms of these, most containing *subexpressions*

Summer 2019

CSE341: Programming Languages

24

24

The semantics

- **Syntax** is just how you write something
- **Semantics** is what that something means
 - **Type-checking** (before program runs)
 - **Evaluation** (as program runs)
- For variable bindings:
 - Type-check expression and extend **static environment**
 - Evaluate expression and extend **dynamic environment**

So what is the precise syntax, type-checking rules, and evaluation rules for various expressions? Good question!

Summer 2019

CSE341: Programming Languages

25

25

ML, carefully, so far

- A program is a sequence of *bindings*
- *Type-check* each binding in order using the *static environment* produced by the previous bindings
- *Evaluate* each binding in order using the *dynamic environment* produced by the previous bindings
 - Dynamic environment holds *values*, the results of evaluating expressions
- So far, the only kind of binding is a *variable binding*
 - More soon

Summer 2019

CSE341: Programming Languages

26

26

Expressions

- We have seen many kinds of expressions:


```
34 true false x e1+e2 e1<e2
if e1 then e2 else e3
```
- Can get arbitrarily large since any subexpression can contain subsubexpressions, etc.
- Every kind of expression has
 1. Syntax
 2. Type-checking rules
 - Produces a type or fails (with a bad error message ☹)
 - Types so far: `int bool unit`
 3. Evaluation rules (used only on things that type-check)
 - Produces a value (or exception or infinite-loop)

Summer 2019

CSE341: Programming Languages

27

27

Variables

- **Syntax:**
 - sequence of letters, digits, `_`, not starting with digit
- **Type-checking:**
 - Look up type in current static environment
 - If not there fail
- **Evaluation:**
 - Look up value in current dynamic environment

Summer 2019

CSE341: Programming Languages

28

28

Addition

- **Syntax:**
 - `e1 + e2` where `e1` and `e2` are expressions
- **Type-checking:**
 - If `e1` and `e2` have type `int`,
 - then `e1 + e2` has type `int`
- **Evaluation:**
 - If `e1` evaluates to `v1` and `e2` evaluates to `v2`,
 - then `e1 + e2` evaluates to sum of `v1` and `v2`

Summer 2019

CSE341: Programming Languages

29

29

Values

- All values are expressions
- Not all expressions are values
- A value "evaluates to itself" in "zero steps"
- Examples:
 - `34, 17, 42` have type `int`
 - `true, false` have type `bool`
 - `()` has type `unit`

Summer 2019

CSE341: Programming Languages

30

30

Slightly tougher ones

What are the syntax, typing rules, and evaluation rules for conditional expressions?

What are the syntax, typing rules, and evaluation rules for less-than expressions?

Summer 2019

CSE341: Programming Languages

31

31

The foundation we need

We have many more types, expression forms, and binding forms to learn before we can write "anything interesting"

Syntax, typing rules, evaluation rules will guide us the whole way!

For Homework 1: functions, pairs, conditionals, lists, options, and local bindings

- Earlier problems require less

Will not add (or need):

- Mutation (a.k.a. assignment): use new bindings instead
- Statements: everything is an expression
- Loops: use recursion instead

Summer 2019

CSE341: Programming Languages

32

32