CS 115
Lecture Notes
Winter 2018

Collin Roberts
January 9, 2018

Contents

1 Lecture 01 1
1.1 Administrivia ..................................................... 2
1.2 Introduction to CS 115 - Course Website and Slides 1-9 .... 2
1.3 Introduction to DrRacket - Slides 10-20 ........................... 2

2 Lecture 02 3
2.1 Administrivia ..................................................... 3
2.2 Mathematical Functions in DrRacket ............................. 3
2.3 The DrRacket Environment ........................................ 4
2.4 Defining New Functions in DrRacket ............................. 4

3 Lecture 03 5

1 Lecture 01

Outline

1. Administrivia
2. Introduction to CS 115 - Course Website and Slides 1-9
3. Introduction to DrRacket - Slides 10-20
1.1  Administrivia

1. Labs start **this week**.
2. I will lecture on the blackboard using the course slides as a resource. I will announce pre-reading for each future lecture.
3. Register your iClicker **exactly once**. The course staff will contact you at the end of January if your iClicker is not yet correctly registered.

1.2  Introduction to CS 115 - Course Website and Slides 1-9

Refer to the slides.

1.3  Introduction to DrRacket - Slides 10-20

Programming language design

- **Imperative:**
  - frequent changes to data
  - examples: machine language, Java, C++

- **Functional:**
  - examples: Excel formulas, LISP, ML, Haskell, Mathematica, XSLT, R (used in STAT 231)
  - more closely connected to mathematics
  - easier to design and reason about programs

We use **DrRacket**, a functional program, in CS 115. DrRacket is great for teaching, although I have yet to see a real computer system developed in it. DrRacket is **Turing Complete**, so in theory any computer system could be developed using it. DrRacket provides an easy entry point into coding.

You will work in the imperative language, Python, in CS 116. Real computer systems are written in Python.

**Themes of the course**

- design (the art of creation)
- abstraction (finding commonality, not worrying about details)
- refinement (revisiting and improving initial ideas)
- syntax (how to say it), expressiveness (how easy it is to say and understand), and semantics (the meaning of what’s being said)
communication (in general)

**Functions:** A mathematical function definition consists of

- the **name** of the function,
- its **parameters** (aka argument) (what the function **consumes**), and
- a mathematical **expression** using the parameters, to define what the function **produces**. The mathematical expression is **evaluated** by substitution.

**Functions in DrRacket:**

- As in the slides, we start with \( f(x) = x^2 \) and \( g(x, y) = x - y \).
- Include exactly one set of parentheses for each function call.
- Write the function first inside the open parenthesis, followed by the arguments.
- Observe that following this syntax removes any ambiguity about the order of operations.
- Using this setup, try some examples of your own in DrRacket.

## 2 Lecture 02

**Outline**

1. Administrivia
2. Mathematical Functions in DrRacket
3. The DrRacket Environment
4. Defining New Functions in DrRacket

### 2.1 Administrivia

### 2.2 Mathematical Functions in DrRacket

<table>
<thead>
<tr>
<th>Familiar Math Notation</th>
<th>DrRacket</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 3 - 2 )</td>
<td>((- 3 2))</td>
</tr>
<tr>
<td>( 3 - 2 + 4/5 )</td>
<td>(+ (- 3 2) (/ 4 5)))</td>
</tr>
<tr>
<td>((6 - 4)(3 + 2))</td>
<td>(* (- 6 4) (+ 3 2)))</td>
</tr>
</tbody>
</table>

- CQ2
- CQ3
2.3 The DrRacket Environment

Built-in Functions - Examples

- (quotient 75 7)
- (remainder 75 7)

Bad Syntax / Semantics - Examples

- (* (5) 3)
- (+ (* 2 4)
- (5 * 14)
- (* + 3 5 2)
- (/ 25 0)
- CQ4

2.4 Defining New Functions in DrRacket

- DrRacket enforces correct syntax at all times.
- If you enter an expression which is not syntactically correct, then you will get an error message.
- Every syntactically correct DrRacket function call has the form `(functionname exp1 . . . expk)`, for some function `functionname` and expressions `exp1 . . . expk`.

<table>
<thead>
<tr>
<th>Familiar Math Notation</th>
<th>DrRacket</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(x) = x^2 )</td>
<td>(define (f x) (* x x))</td>
</tr>
<tr>
<td>( g(x, y) = x - y )</td>
<td>(define (g x y) (- x y))</td>
</tr>
</tbody>
</table>

- `define` is a special form; it looks like a DrRacket function, but not all of its arguments are evaluated.
- It binds a name to an expression (which uses the parameters that follow the name).

Identifiers

- To give names to the function and parameters, we use identifiers.
- Syntax rule: an `identifier` starts with a letter, and can include letters, numbers, hyphens, underscores, and a few other punctuation marks.
• It cannot contain spaces or any of ( ) { } [] ‘ ’ “ ”.
• Syntax rule: **function definition** is of the form
  
  (define (id1 . . . idk) exp), where exp is an expression and each id is an identifier.

• CQ5

**DrRacket Definitions Window**

• can accumulate definitions and expressions
• Run button loads contents into Interactions window
• can save and restore Definitions window
• provides a Stepper to let one evaluate expressions step-by-step
• features include: error highlighting, subexpression highlighting, syntax checking, bracket matching

**DrRacket Constants**

<table>
<thead>
<tr>
<th>Familiar Math Notation</th>
<th>DrRacket</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k = 3 )</td>
<td>(define k 3)</td>
</tr>
<tr>
<td>( p = k^2 )</td>
<td>(define p (* k k))</td>
</tr>
</tbody>
</table>

**DrRacket Programs** A Racket program is a sequence of definitions and expressions.

• The definitions are of functions and constants.
• The expressions typically involve both user-defined and built-in functions and constants.

Programs may also make use of special forms (which look like functions, but don’t necessarily evaluate all their arguments).

• CQ6

### 3 Lecture 03

**Outline**

1. Administrivia
2. Slides 39-53