CSCI-C311 Programming Languages

Introduction to Racket

Dr. Hang Dinh

1

What is Racket?

- Depending on how you look at it, **Racket** is
  - a *programming language*—a dialect of Lisp and a descendant of Scheme;
  - a *family* of programming languages—variants of Racket, and more; or
  - a set of *tools*—for using a family of programming languages.
- Racket's main tools
  - **racket**, the core compiler, interpreter, and run-time system;
  - **DrRacket**, the integrated programming environment; and
  - **raco**, a command-line tool for executing **Ra**cket **co**mmands that install packages, build libraries, and more.

2

Documents about Racket

- Getting Started
  - https://docs.racket-lang.org/getting-started/index.html
  - For how to install Racket and use DrRacket
- [Tutorial] Quick: An Introduction to Racket with Pictures
  - https://docs.racket-lang.org/quick/index.html
- The Racket Guide
  - https://docs.racket-lang.org/guide/index.html
- The Racket Reference
  - https://docs.racket-lang.org/reference/index.html

3

Reading Assignment for This Lecture

- Getting Started
- [Tutorial] Quick: An Introduction to Racket with Pictures
  - Parts 1, 2, 3, 4
- The Racket Guide
  - Sections 1.1, 1.2, 1.3

5

Using DrRacket and The Racket Language

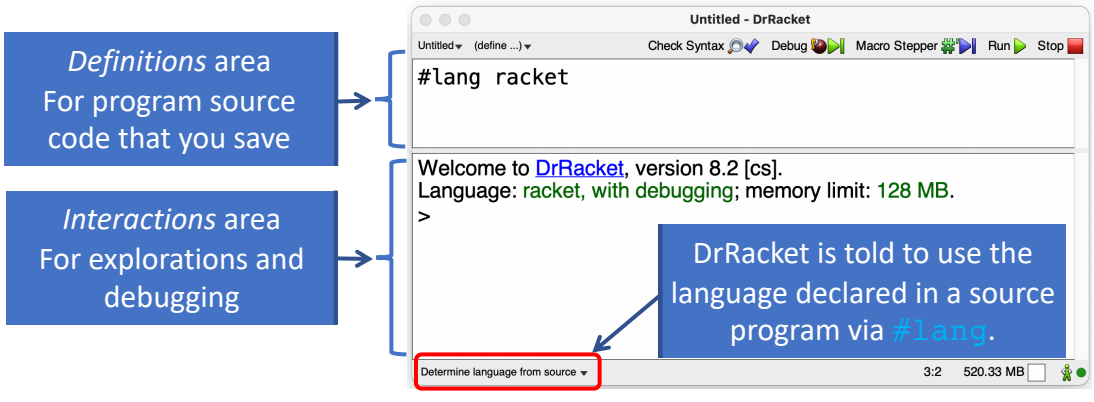
- When using DrRacket, need to choose the proper language
  - By typing a line starting with `#lang`
  - To use the normal variant of Racket language, start the program with the line:

    `#lang racket`
- If you've used DrRacket before,
  - DrRacket may remember the last language that you used, instead of inferring the language from the `#lang` line.
  - In that case, use the **Language|Choose Language...** menu item to tell DrRacket to use the language that is declared in a source program via `#lang`.

6

Quick: An Introduction to Racket with Pictures
1. Ready…

- Download Raket from https://download.racket-lang.org/
- Install, and then start DrRacket

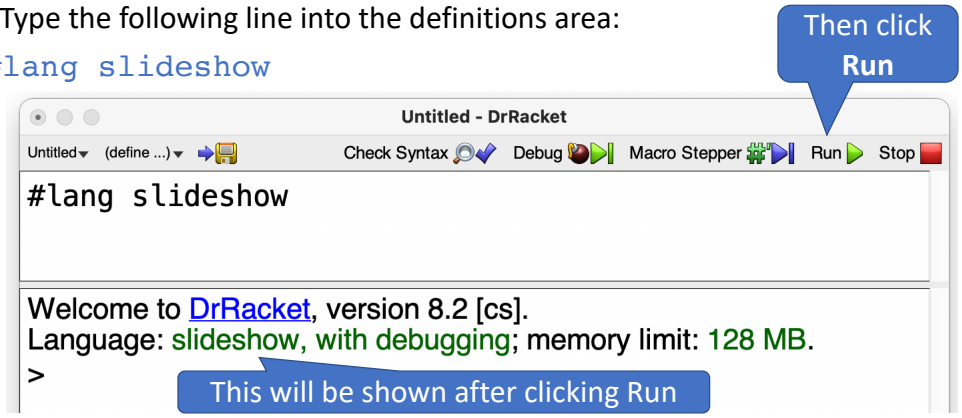


7

Quick: An Introduction to Racket with Pictures
2. Set…

- Load some picture functions (to draw pictures)
  - Part of a library for creating slide presentations
  - Type the following line into the definitions area:

```
#lang slideshow
```



8

Quick: An Introduction to Racket with Pictures
3. Go!

- Type an expression after the **>** in the interactions area and hit **Enter**

```
> 31
31
> "programing languages"
"programing languages"
>
```

An expression can be just a value, such as a number or a string.

- An expression can also be a function call

```
> (circle 15)
○
> (rectangle 15 25)
□
```

To call a function, put an open parenthesis before the function name, then function arguments, and then a close parenthesis

9

Quick: An Introduction to Racket with Pictures
3. Go!

- Function `hc-append` combines pictures horizontally where pictures are centered vertically
- Compose function calls in Racket:

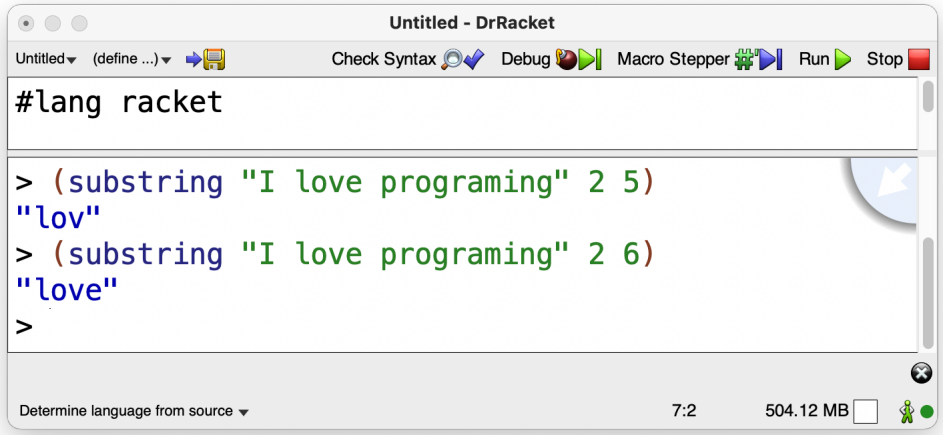
```
> (hc-append (circle 15) (rectangle 15 25))
> (hc-append 10 (circle 15) (rectangle 15 25))
```

- To learn more about `hc-append` function
  - Select the name `hc-append` and press the **F1** key in DrRacket

10

Another Built-in Function

- Function `substring` in `#lang racket`



11

Interacting with Racket

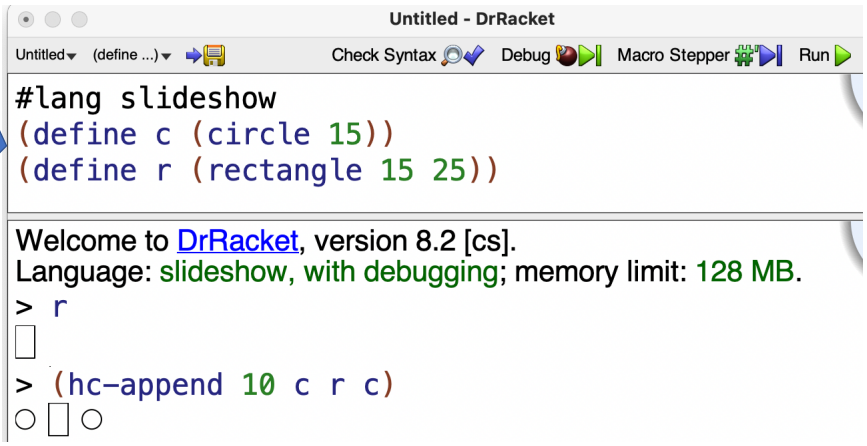
- DrRacket's interactions area and the racket command-line program (when started with no options) both act as a kind of calculator.
  - You type a Racket expression, hit the Return key, and the answer is printed.
  - This kind of calculator is called a ***read-eval-print loop*** or **REPL**.
- Racket uses parentheses to wrap larger expressions
  - almost any kind of expression, other than simple constants.

12

Quick: An Introduction to Racket with Pictures
4. Definitions

- Use the `define` form for definitions

Although you can evaluate the `define` forms in the REPL, definitions are normally a part of a program that you want to keep and use later.

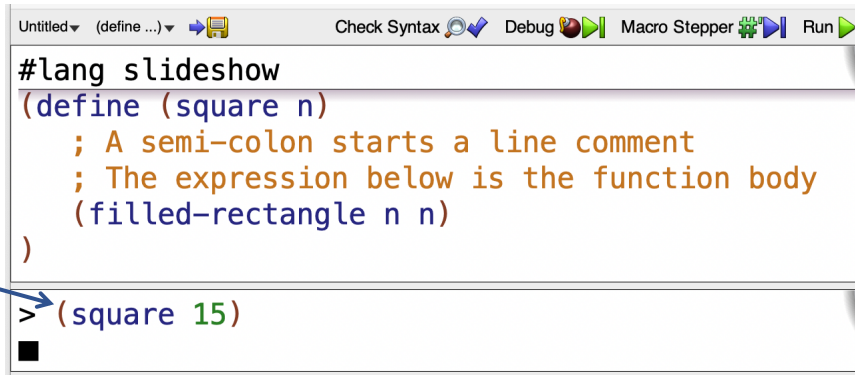


13

Quick: An Introduction to Racket with Pictures
4. Definitions

- Use the `define` form to define a function
  - but with an open parenthesis before the function name, and names for the function arguments before the matching close parenthesis

You can put this expression in the definition area. When a program is run, expression results from the definition area are shown in the interaction area.



Using Command-line `racket`

- When using command-line `racket` instead of DrRacket,
  - Save the text in the definitions area as an **.rkt** file using your favorite editor
  - Start the `racket` program
  - Evaluate the `enter!` form to load the code, and switch the evaluation context to the inside of the module, just like the **Run** button of DrRacket.
    - If **pictures.rkt** is saved in the same folder as `racket` you can evaluate:

```
>(enter! "pictures.rkt")
>(square 20)
```

  - Can also run a program from a command line

```
racket pictures.rkt
```

pictures.rkt

```
#lang slideshow
(define c (circle 15))
(define r (rectangle 15 25))
(define (square n)
   ; A semi-colon starts a line comment
   ; The expression below is the function body
   (filled-rectangle n n)
)
(square 15)
```

Creating Executables

- To package the program as an executable, you have a few options:
  - In DrRacket, you can select the **Racket|Create Executable...** menu item.
  - From a command-line prompt, run `raco exe ‹src-filename›`, where ‹*src-filename*› contains the program.
- With Unix or Mac OS, you can turn the program file into an executable script by inserting this line at the beginning of the file:

```
#! /usr/bin/env racket
```

  - Also change the file permissions to executable using the command line

```
chmod +x ‹filename›
```

  - The script works as long as racket is in the user's executable search path.

16