

---

## Lesson #1: Getting Acquainted with Racket/DrRacket

---

---

---

### What's It All About?

---

---

1. The first thing you will need to do in order that you can do some Racket programming in the DrRacket program development environment (PDE) is to download the software. The first part of this lesson points you in the right direction to do just that.
2. Then, you will want to get acquainted with the nature of Racket and the functionality of DrRacket. Three interactive sessions are presented to help you with that. The first interactive session presents a few numeric computations. The second interactive session solves a little numerical problem. The third interactive session makes use of a graphics library to illustrate the problem situation for the problem solved in the second interactive session.

---

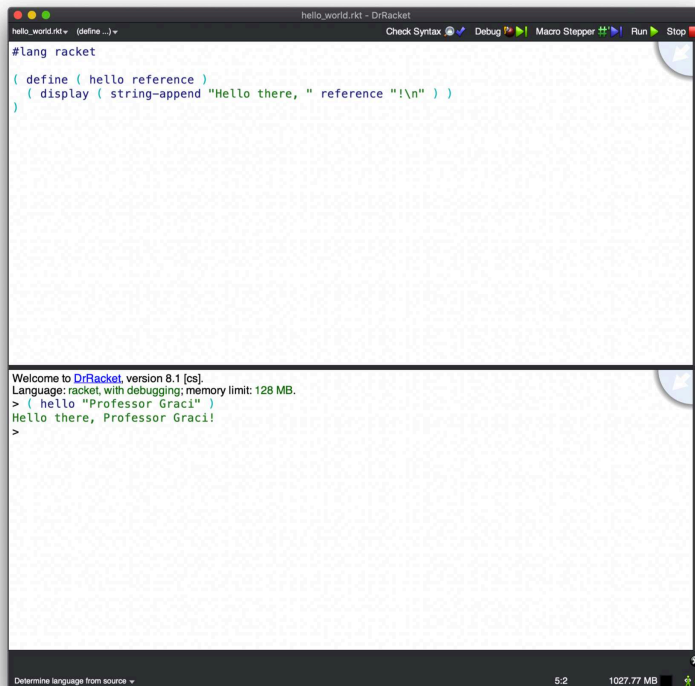
### The software

---

---

Find your way in a browser to [racket-lang.org](http://racket-lang.org) and then to the download button in the top righthand area of the page. Click it, and proceed to do what you need to do to download Racket/DrRacket to your machine. Your goal is to launch DrRacket, which should present itself in a window partitioned into a Definitions area and an Interactions area – along with surrounding bits of functionality.

Consider the following image of a DrRacket window.



```
hello_world.rkt - DrRacket
hello_world.rkt (define ...)
Check Syntax Debug Macro Stepper Run Stop

#lang racket
(define (hello reference)
  (display (string-append "Hello there, " reference "!\\n" )))
)

Welcome to DrRacket, version 8.1 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (hello "Professor Graci")
Hello there, Professor Graci!
>
```

Determine language from source 5:2 1027.77 MB

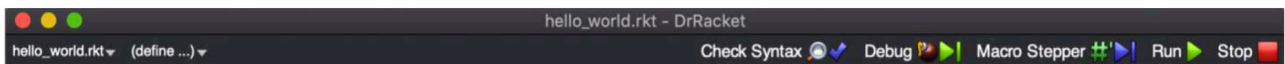
As you consider the image, imagine that:

1. I downloaded the software and then launched DrRacket.
2. I typed some code into the Definitions area, the top buffer in the arrangement that was displayed. Following the line that declares the language to be Racket (which is generally inserted automatically by the system), I typed in a variant of the traditional hello world program.

```
#lang racket

( define ( hello reference )
  ( display ( string-append "Hello there, " reference "!\n" ) )
)
```

3. I then clicked on the Run icon, the green triangle towards the right of the “control bar”, which normally appears at the top of the PDE window, but which appears just below in a disembodied manner below. This action caused the Interactions area, the bottom buffer in the arrangement that was displayed, to become aware of the function definition that I placed in the Definitions area.



4. In the Interactions area (the bottom buffer), I typed in a form asking that the hello world function definition be executed. Once I hit the return key, the interpreter presented the result of the execution.

```
Welcome to DrRacket, version 8.1 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( hello "Professor Graci" )
Hello there, Professor Graci!
>
```

Please download the software and mimic what I did in writing and running the hello world variant that is represented in the image of the DrRacket PDE.

---

## About Racket and DrRacket

---

1. Racket is a variant of Scheme, which was created at MIT during the 1970s by Guy Steele and Gerald Sussman. Scheme is a language influenced by Lisp and Algol. From Lisp, conceived by John McCarthy at MIT in the

late 1950s, it takes the the syntax of S-expressions (and all that goes with that), the treatment of functions as first class entities, and garbage collection. From Algol, designed by committee in Europe around 1960, it takes static scoping and block structure. When the time is right, we will talk about all of these ideas.

2. Most people programming in Racket do so within the DrRacket program development environment (PDE), which was presented in brief in the preceding section.
3. You will soon note that I prefer to configure DrRacket so that the two windows are aligned horizontally, rather than vertically. There are lots of things that you can do to change the “look and feel” of DrRacket. Just look in the menus if you want to find the PDE functionality for those things.
4. It is often comforting to know where to look for relevant tutorials and documentation when getting acquainted with a language. These are generally acknowledged to be the go to materials for those aspiring “Racketeers”:
  - (a) Quick: An Introduction to Racket with Pictures  
<https://docs.racket-lang.org/quick/>
  - (b) The Racket Guide  
<https://docs.racket-lang.org/guide/>
  - (c) The Racket Reference  
<https://docs.racket-lang.org/reference/>

---

## Interactions: Simple Numeric Processing

---

The following Racket interaction is simply intended to introduce numbers and numeric operators in Racket. *What can you say about Racket as a result of the interaction? What questions does the interaction most immediately bring to mind?*

Welcome to [DrRacket](#), version 8.1 [cs].  
Language: racket, with debugging; memory limit: 128 MB.

> x



x: undefined;  
cannot reference an identifier before its definition

> 55

55

> 55.2

55.2

> pi

3.141592653589793

> ( \* 3 8 )

24

> ( + ( \* 3 8 ) 6 )

30

> ( expt 2 8 )

256

> ( \* pi ( expt 7 2 ) )

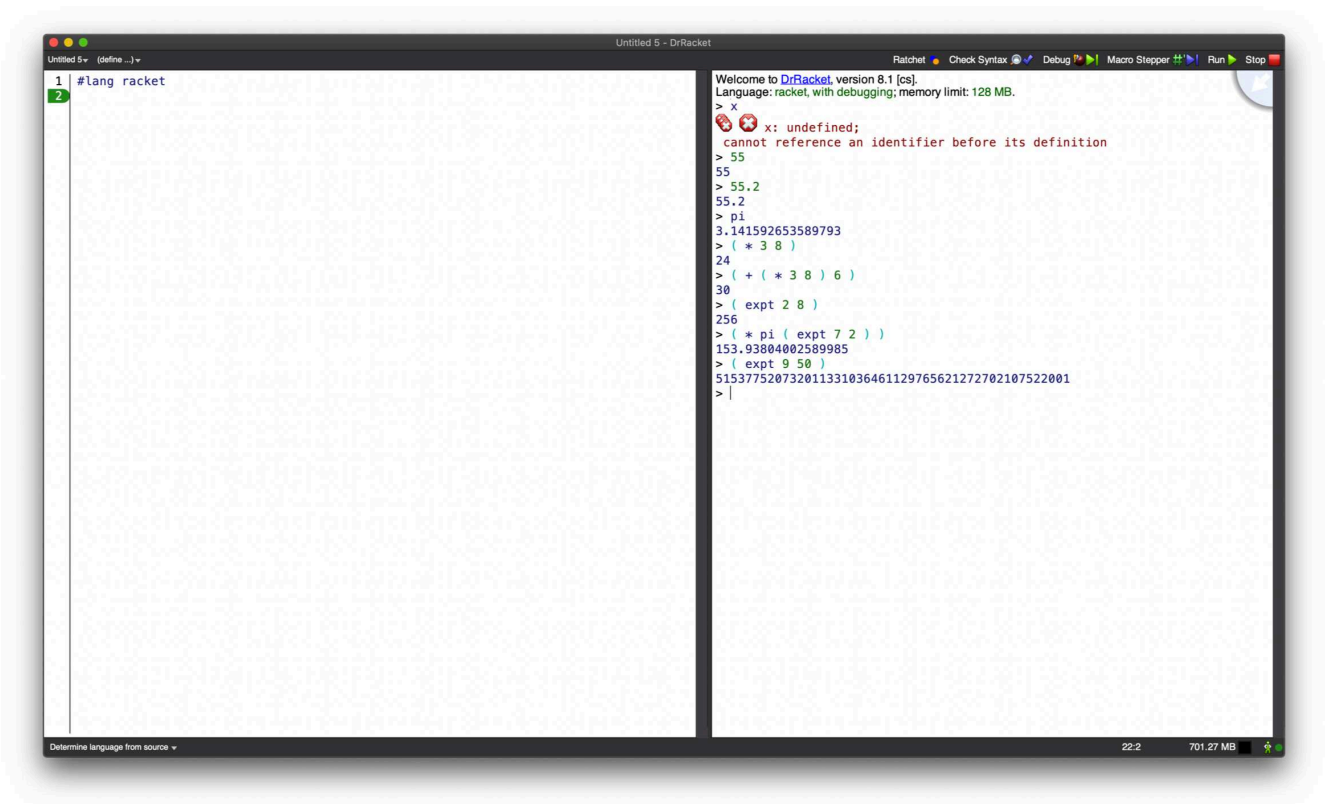
153.93804002589985

> ( expt 9 50 )

515377520732011331036461129765621272702107522001

>

As previously mentioned, I generally like to arrange my panes in a side by side configuration, with the Definitions pane on the left and the Interactions pane on the right. Take a look, and you will see that I lifted the previously presented interaction from the right pane of the following window.



---

## Problem: Compute the blue area of a blue and red tile

---

A tile of side 200 is blue, except for a centered red disk of diameter one-third the side of the tile. What is the area of the tile which is blue?

We will actually do two things with respect to this problem:

1. We will perform some interactions in Racket to solve the problem.
2. We will perform some Racket interactions in the context of a little graphics library to paint a picture of the problem situation.

---

## Interactions area: Solution to the area problem

---

The following Racket interaction presents a solution to the area problem. *What can you say about Racket as a result of the following interaction? What questions does the interaction most immediately bring to mind?*

```
Welcome to DrRacket, version 8.1 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( define side-of-tile 200 )
> ( define diameter-of-dot ( / side-of-tile 3 ) )
> ( define radius-of-dot ( / diameter-of-dot 2 ) )
> ( define total-tile-area ( expt side-of-tile 2 ) )
> ( define red-dot-area ( * pi ( expt radius-of-dot 2 ) ) )
> ( define blue-tile-area ( - total-tile-area red-dot-area ) )
> side-of-tile
200
> diameter-of-dot
66 $\frac{2}{3}$ 
> radius-of-dot
33 $\frac{1}{3}$ 
> total-tile-area
40000
> red-dot-area
3490.658503988659
> blue-tile-area
36509.341496011344
>
```

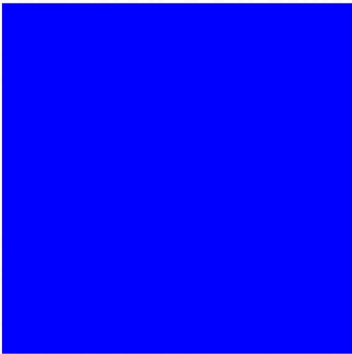
---

## Interactions: Illustration of the area problem situation

---

A “library” called `2htdp/image` is available for drawing and painting images. By way of introduction to this library, some simple computations are performed to produce pictures consistent with the blue tile area problem situation. *What can you say about Racket as a result of the following interaction? What questions does the interaction most immediately bring to mind?*

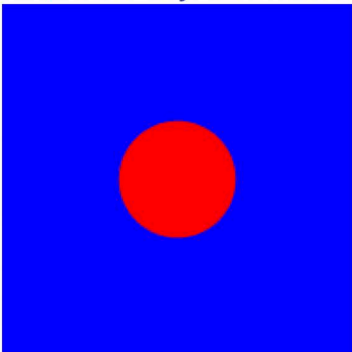
```
Welcome to DrRacket, version 8.1 [cs].  
Language: racket, with debugging; memory limit: 128 MB.  
> ( require 2htdp/image )  
> ( define side-of-tile 200 )  
> ( define diameter-of-dot ( / side-of-tile 3 ) )  
> ( define radius-of-dot ( / diameter-of-dot 2 ) )  
> ( define tile ( square side-of-tile "solid" "blue" ) )  
> tile
```



```
> ( define dot ( circle radius-of-dot "solid" "red" ) )  
> dot
```



```
> ( overlay dot tile )
```



```
>
```