# > Lesson 1: Getting Acquainted with Racket/DrRacket

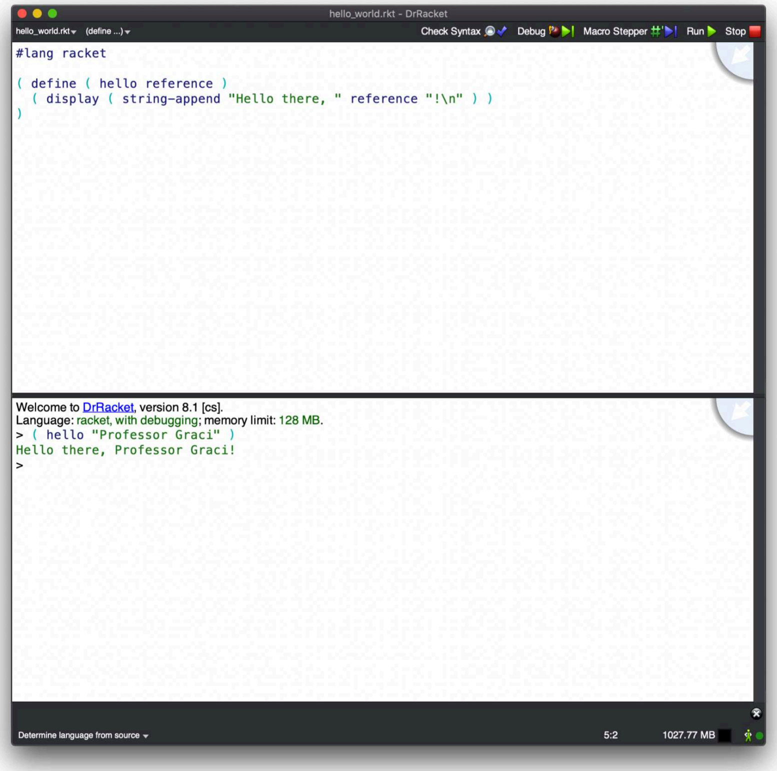
## > What's It All About?

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` 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.



```
#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!
>
```

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 is displayed. I first typed in a line that declared the language to be Racket. I then typed in a variant of the traditional hello world program.
3. I clicked on the "Run" icon, which caused the Interactions area, the bottom buffer in the arrangement, to become aware of the function definition that I placed in the Definitions area.
4. In the Interations area, I typed in a form which amounts to asking that the hello world function definition be executed, which in turn called on me to type in just a bit of information, and the interpreter then presented the result of the execution.

**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":

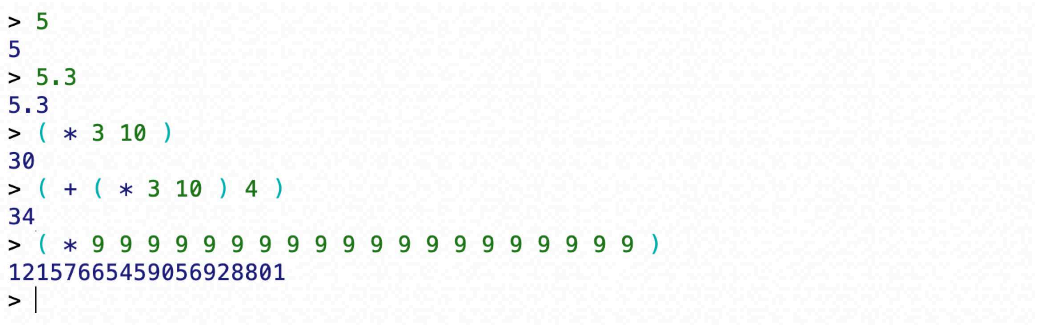
   (a) Quick: An Introduction to Racket with Pictures
   `https://docs.racket-lang.org/quick/`

   (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?*

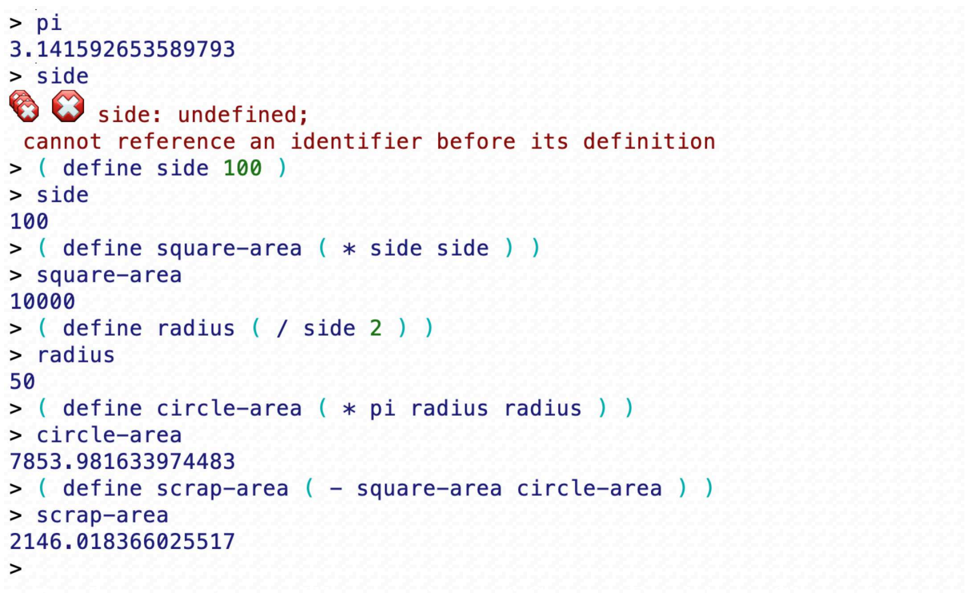


```
> 5
5
> 5.3
5.3
> ( * 3 10 )
30
> ( + ( * 3 10 ) 4 )
34
> ( * 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 )
12157665459056928801
> |
```

## > Interactions: Solution to the Scrap Problem

**The Scrap Problem**: A circular disk of maximal size is cut from a square piece of tin of side 100 units. What is the area of the scrap?

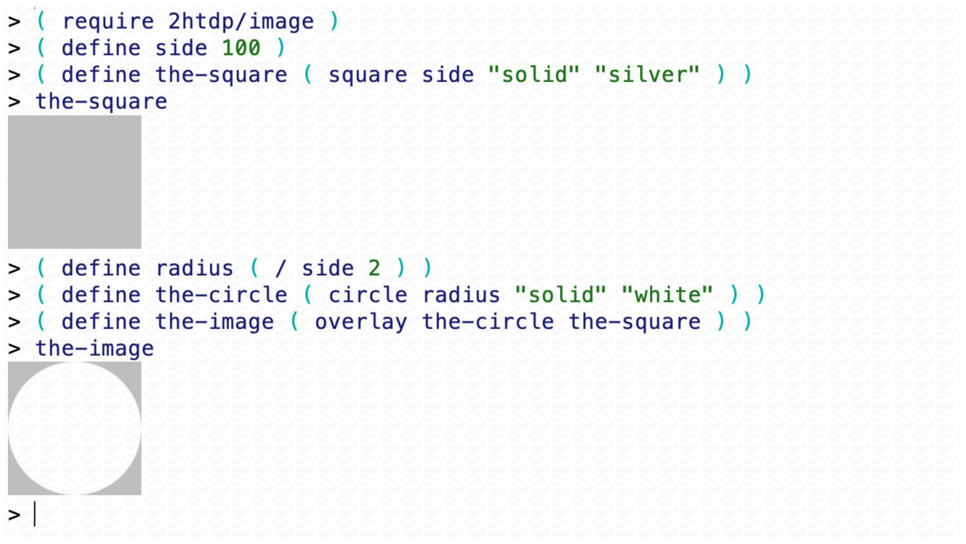
The following Racket interaction presents a solution to the Scrap problem. *What can you say about Racket as a result of the following interaction? What questions does the interaction most immediately bring to mind?*



```
> pi
3.141592653589793
> side
side: undefined;
 cannot reference an identifier before its definition
> ( define side 100 )
> side
100
> ( define square-area ( * side side ) )
> square-area
10000
> ( define radius ( / side 2 ) )
> radius
50
> ( define circle-area ( * pi radius radius ) )
> circle-area
7853.981633974483
> ( define scrap-area ( - square-area circle-area ) )
> scrap-area
2146.018366025517
>
```

## > Interactions: Illustration of Scrap 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 Scrap 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?*



```
> ( require 2htdp/image )
> ( define side 100 )
> ( define the-square ( square side "solid" "silver" ) )
> the-square
```

```
> ( define radius ( / side 2 ) )
> ( define the-circle ( circle radius "solid" "white" ) )
> ( define the-image ( overlay the-circle the-square ) )
> the-image
```

```
> 
```