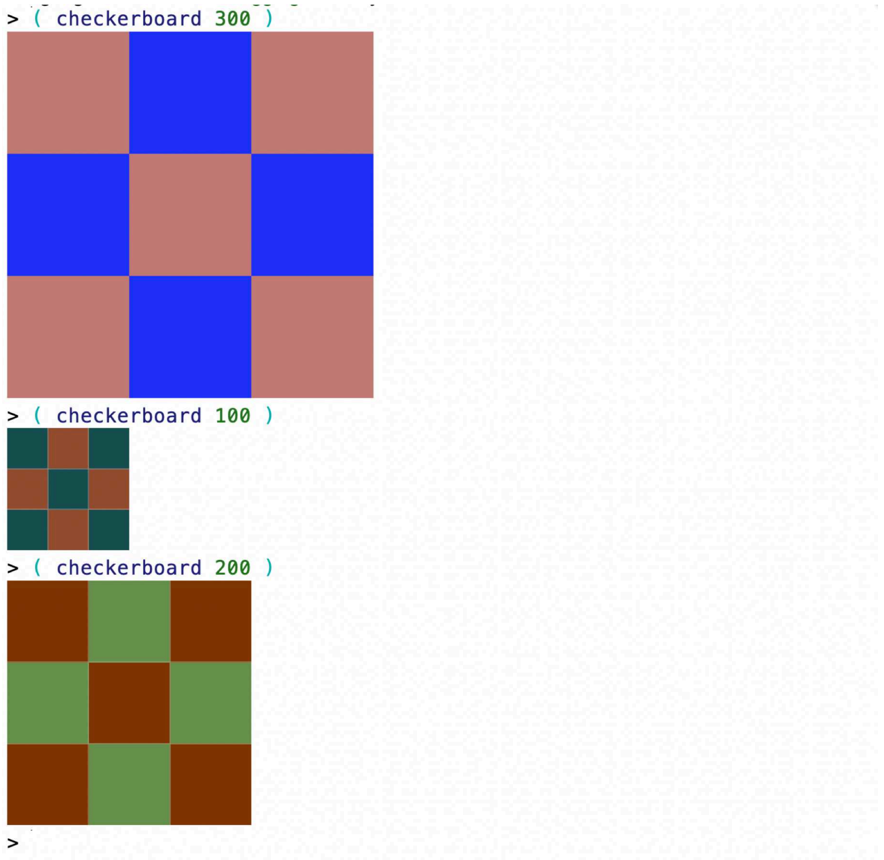
Lesson #2: Defining Functions

What's It All About?

1. By way of introducing you to function definition in Racket, we will write a program to paint randomly colored 3x3 checkerboards of various sizes.
2. In the process, you will see function definitions with parameters, and corresponding function applications with arguments. You will also see that variables introduced within a function are considered to be local to the function.
3. Also, in the process, you will be encouraged to adopt an approach to programming which can be characterized as "incremental development with bits of play thrown in".
4. And, you will experience the delight of painting with random colors!

The Problem

To ground the discussion of function definition in Racket, we will **develop** a program to paint a 3x3 randomly colored checkerboard, whose size is determined by a parameter to the main checkerboard painting function. A little demo:



```
> ( checkerboard 300 )
> ( checkerboard 100 )
> ( checkerboard 200 )
>
```

The Approach

The word “develop” was emphasized in the problem statement, since I intend to suggest an **incremental development process** which incorporates **bits of play**, as I present the program that I have in mind. Certainly this may seem like a bit of overkill for such a simple problem. But you should probably consider taking this idea of “incremental development accompanied by bits of play”, or “IDAP”, seriously in programming of solutions to more substantial problems, since it has the potential to serve you well in many respects. The approach may, for example, provide you with inspiration, insight, and information, to better motivate and guide you.

Prelude of Play

Simply perform some interactions in order to get a feel for how to generate random numbers, random colors, and squares.

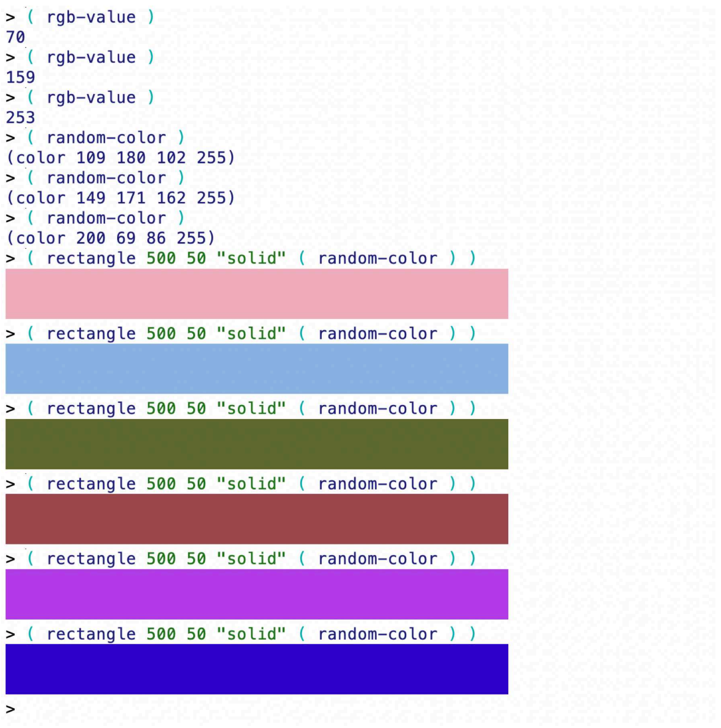
Step 1 - Generate random colors

The first step will be to write a little Racket function to generate a random color. In support of this function, we will write an auxiliary function to generate a random rgb-value.

Step 1 - Code

```
( define ( random-color ) ( color ( rgb-value ) ( rgb-value ) ( rgb-value ) ) )
( define ( rgb-value ) ( random 256 ) )
```

Step 1 - Demo



```
> ( rgb-value )
70
> ( rgb-value )
159
> ( rgb-value )
253
> ( random-color )
(color 109 180 102 255)
> ( random-color )
(color 149 171 162 255)
> ( random-color )
(color 200 69 86 255)
> ( rectangle 500 50 "solid" ( random-color ) )
> ( rectangle 500 50 "solid" ( random-color ) )
> ( rectangle 500 50 "solid" ( random-color ) )
> ( rectangle 500 50 "solid" ( random-color ) )
> ( rectangle 500 50 "solid" ( random-color ) )
> ( rectangle 500 50 "solid" ( random-color ) )
>
```

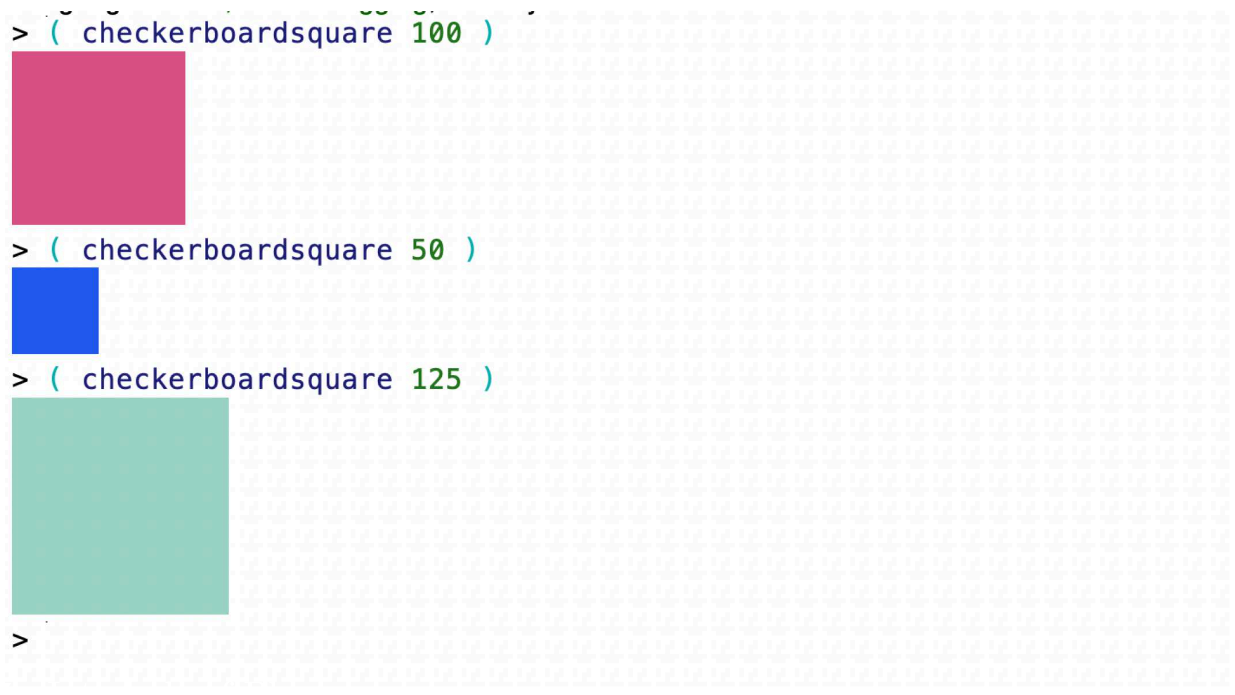
Step 2 - Generating a randomly colored checkerboard square of given size

The second step will be to generate a randomly colored checkerboard square of given a side length, making good use of the `random-color` function.

Step 2 - Code

```
( define ( checkerboardsquare side )
  ( square side "solid" ( random-color ) )
)
```

Step 2 - Demo



```
> ( checkerboardsquare 100 )
```

```
> ( checkerboardsquare 50 )
```

```
> ( checkerboardsquare 125 )
```

```
>
```

Interlude of Play

Simply perform some interactions in order to create a couple of checkerboard square rows of length 3, using the `beside` function, and a checkerboard, by combining the rows, using the `above` function.

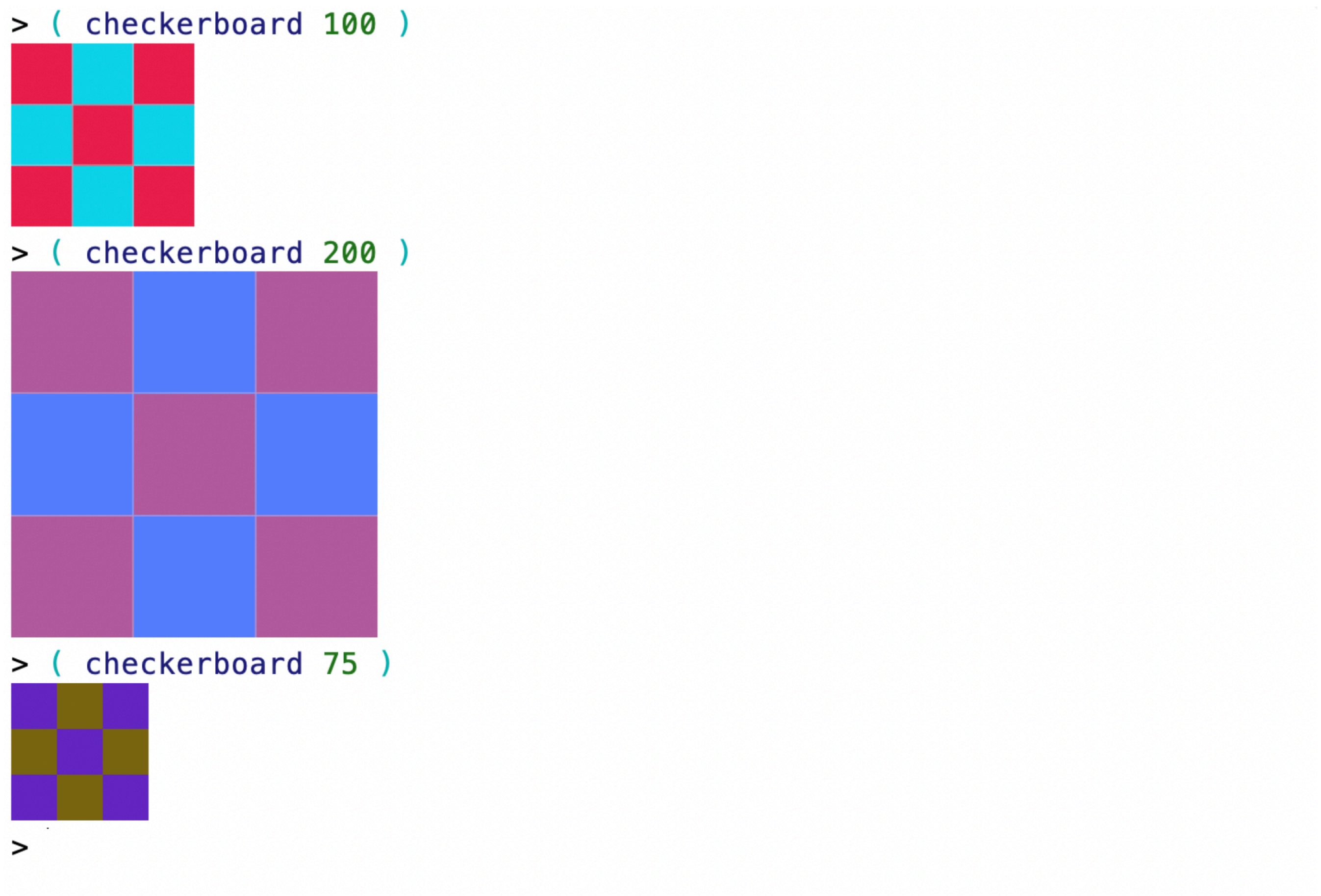
Step 3 - Generating the checkerboard

The third and last step will be to write the function to generate a desired checkerboard.

Step 3 - Code

```
( define ( checkerboard side-of-board )
  ( define side-of-cell ( / side-of-board 3 ) )
  ( define square1 ( checkerboardsquare side-of-cell ) )
  ( define square2 ( checkerboardsquare side-of-cell ) )
  ( define row-121 ( beside square1 square2 square1 ) )
  ( define row-212 ( beside square2 square1 square2 ) )
  ( define the-checkerboard ( above row-121 row-212 row-121 ) )
  the-checkerboard
)
```

Step 3 - Demo



```
> ( checkerboard 100 )
> ( checkerboard 200 )
> ( checkerboard 75 )
>
```