Racket Programming Assignment #1 : First Interactions

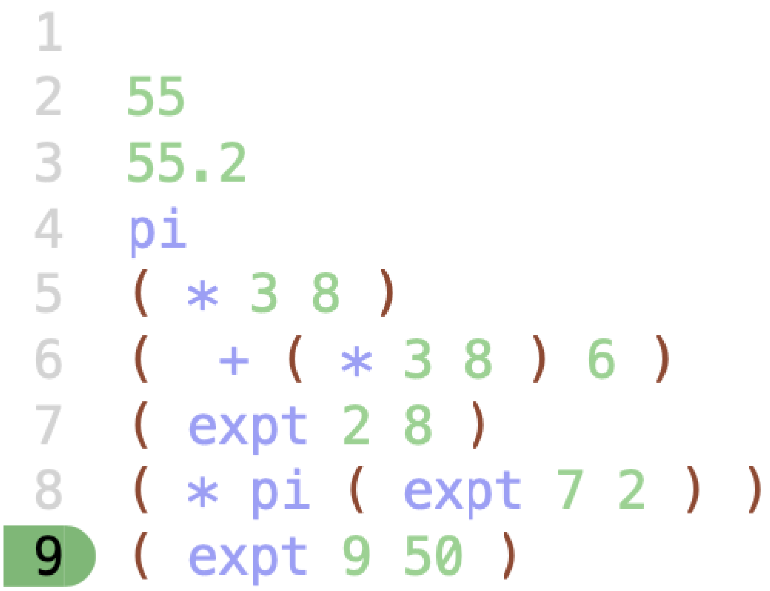
Learning Abstract:

This assignment allowed us to understand the professor's use of the programming language Racket during the first lecture of this semester. The first parts were pretty much a reiteration of the Professor's exercises, with the last two parts allowing me to apply the concepts from the previous activities.

Part four allowed us to go above just painting circles as we learned in part three. This time, we are expected to paint multiple circles on top of each other, with each new circle's radius decrementing from the last.

Part 5 saw us calculating the total area of the blue circles relative to the red circles overlaying on top of one another. This pushed our math skills and understanding of how to compute functions in the racket language.

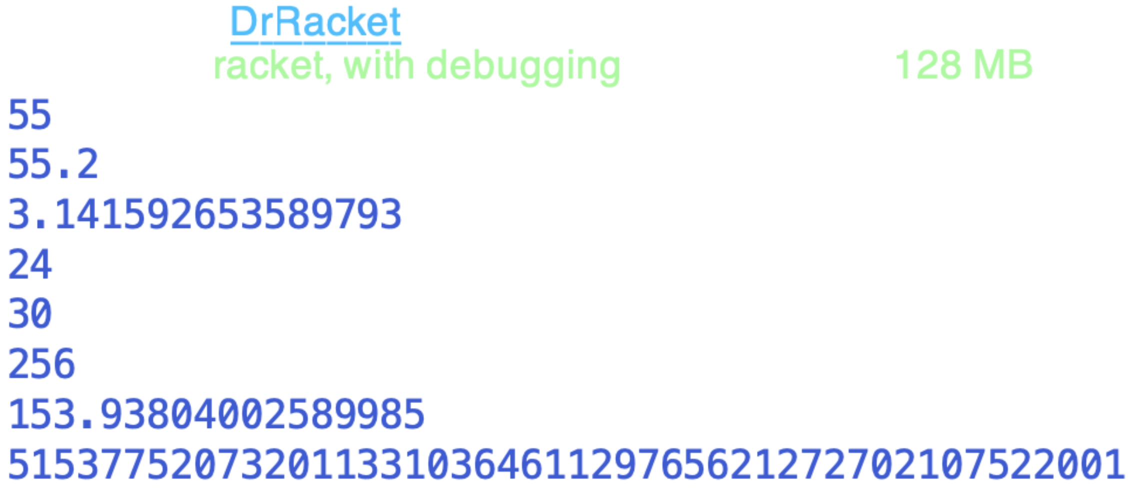
Interactions: Simple Numeric Processing



```

55
55.2
pi
( * 3 8 )
(  + ( * 3 8 ) 6 )
( expt 2 8 )
( * pi ( expt 7 2 ) )
( expt 9 50 )
```

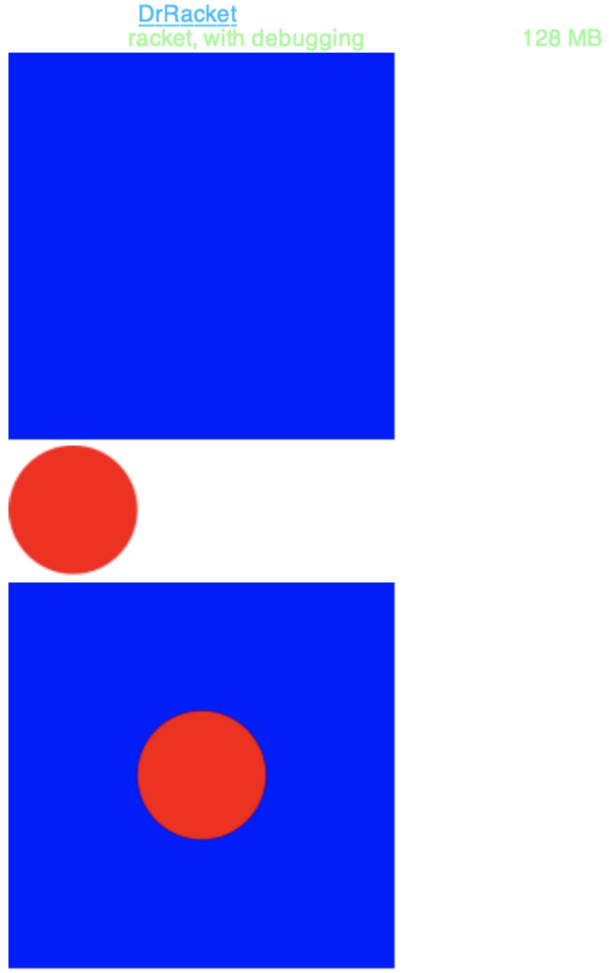
Interaction: Solution to the blue and red tile area problem



DrRacket
racket, with debugging 128 MB

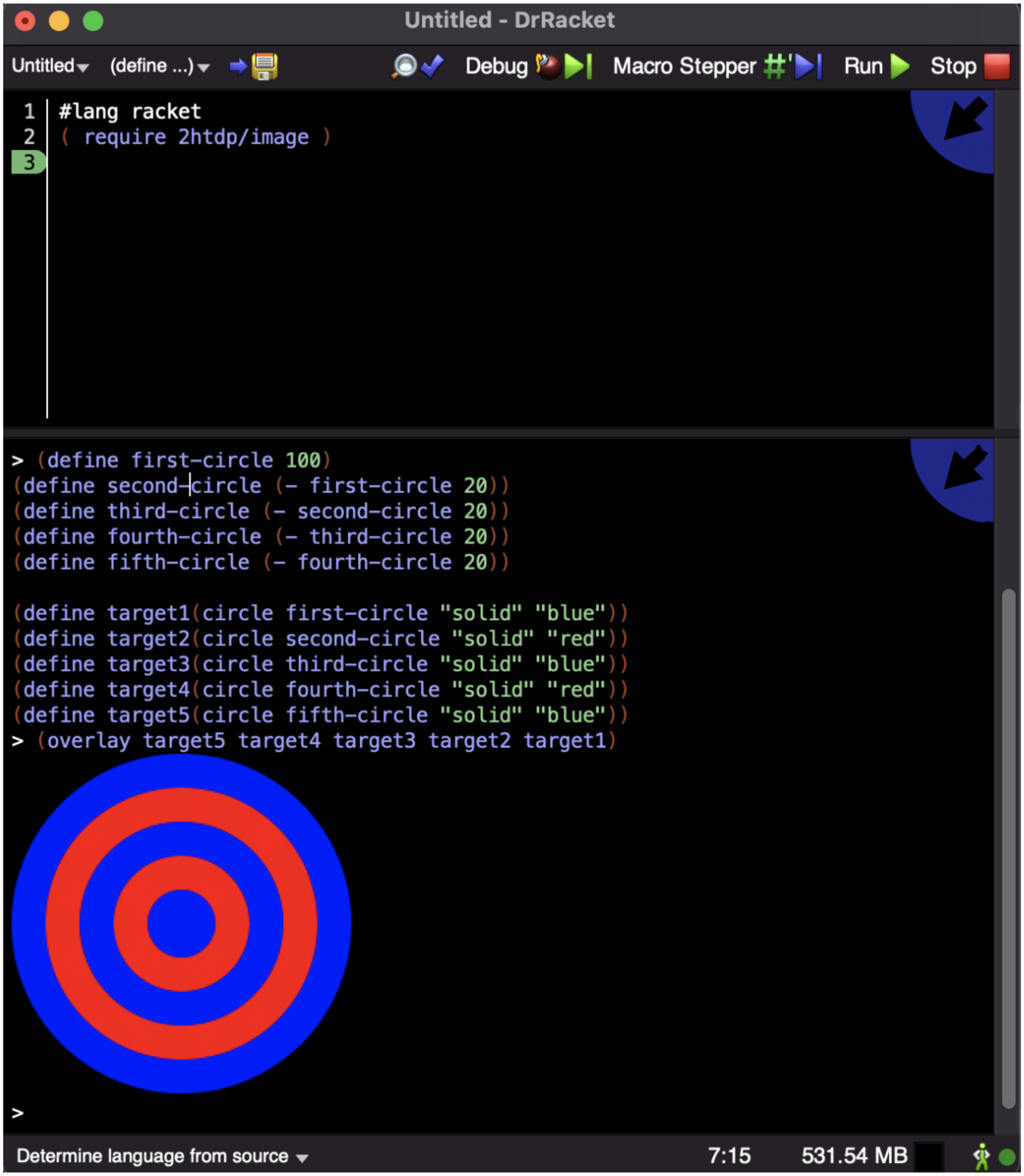
```
55
55.2
3.141592653589793
24
30
256
153.93804002589985
515377520732011331036461129765621272702107522001
```

Interaction: Painting the blue and red tile



e

Interaction: Painting the blue and red concentric disks image



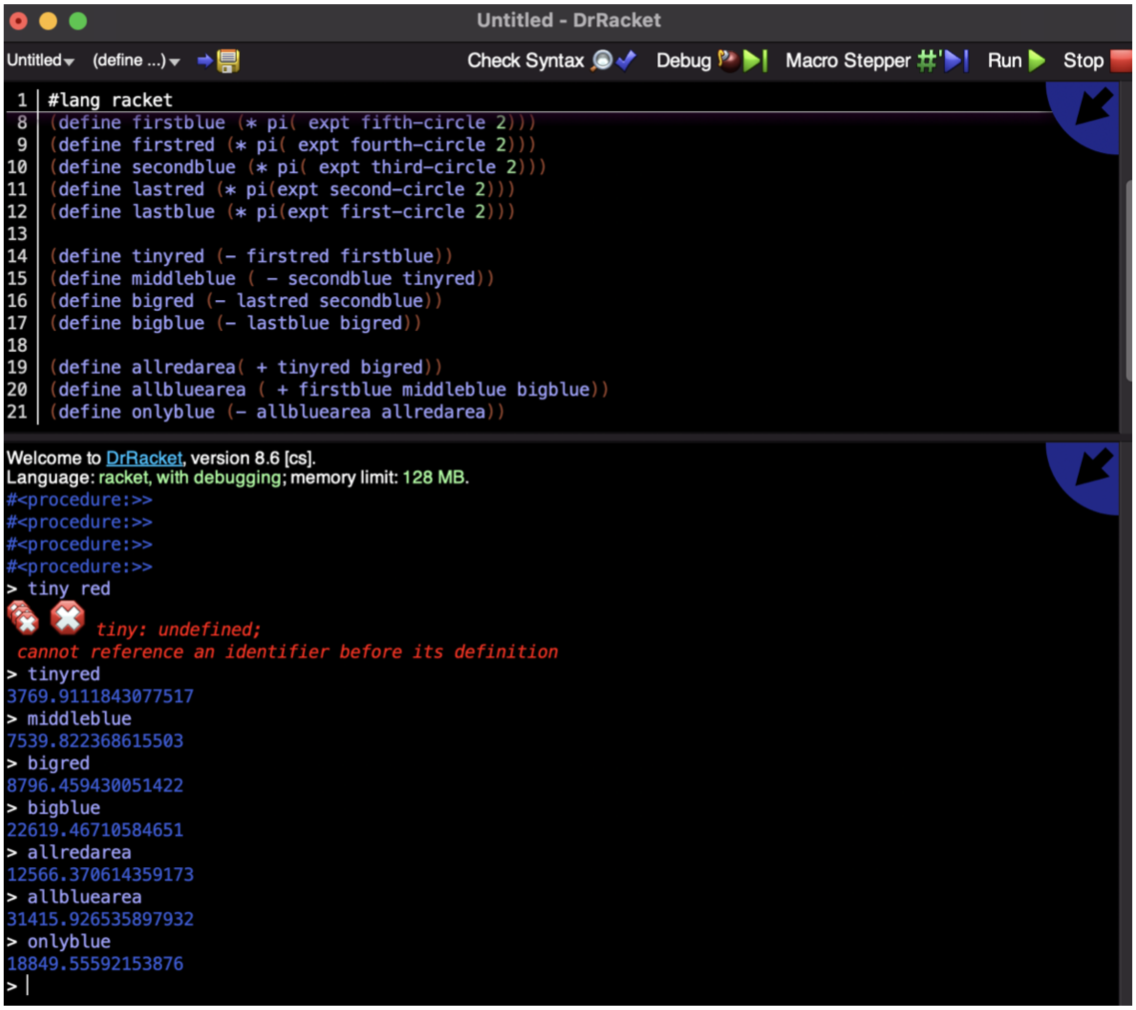
```
#lang racket
( require 2htdp/image )
```

```
> (define first-circle 100)
(define second-circle (- first-circle 20))
(define third-circle (- second-circle 20))
(define fourth-circle (- third-circle 20))
(define fifth-circle (- fourth-circle 20))

(define target1(circle first-circle "solid" "blue"))
(define target2(circle second-circle "solid" "red"))
(define target3(circle third-circle "solid" "blue"))
(define target4(circle fourth-circle "solid" "red"))
(define target5(circle fifth-circle "solid" "blue"))
> (overlay target5 target4 target3 target2 target1)
```

```
>
```

Interaction: Computing the area of the concentric disks image which is blue



```
#lang racket
(define firstblue (* pi( expt fifth-circle 2)))
(define firstred (* pi( expt fourth-circle 2)))
(define secondblue (* pi( expt third-circle 2)))
(define lastred (* pi(expt second-circle 2)))
(define lastblue (* pi(expt first-circle 2)))

(define tinyred (- firstred firstblue))
(define middleblue ( - secondblue tinyred))
(define bigred (- lastred secondblue))
(define bigblue (- lastblue bigred))

(define allredarea( + tinyred bigred))
(define allbluearea ( + firstblue middleblue bigblue))
(define onlyblue (- allbluearea allredarea))
```

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
#<procedure:>>
#<procedure:>>
#<procedure:>>
#<procedure:>>
> tiny red
      tiny: undefined;
 cannot reference an identifier before its definition
> tinyred
3769.9111843077517
> middleblue
7539.822368615503
> bigred
8796.459430051422
> bigblue
22619.46710584651
> allredarea
12566.370614359173
> allbluearea
31415.926535897932
> onlyblue
18849.55592153876
> |
```