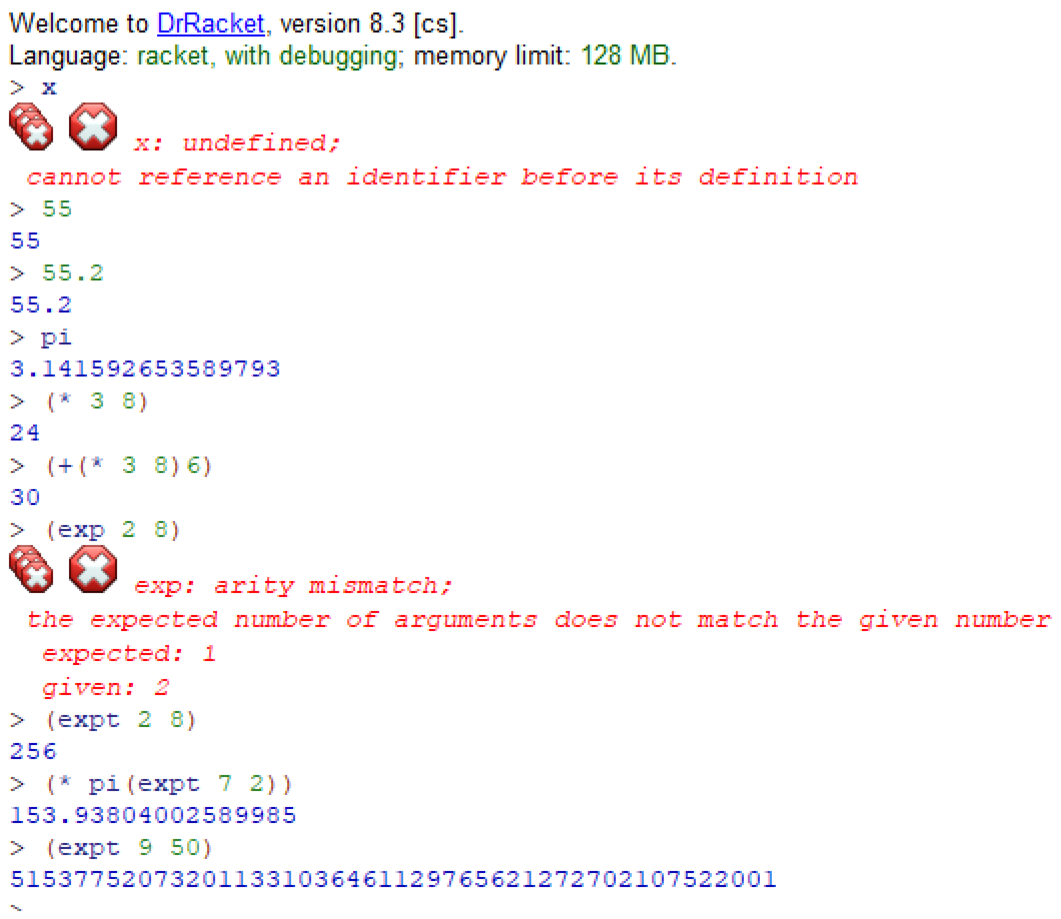
Racket Programming Assignment #1: First Interactions

Learning Abstract

This assignment features relatively simple interactions in the Racket programming language. In the first part of this assignment I learned a little bit about numeric computations in Lisp. By copying the example solutions document I was able to get my feet wet with the program and become familiar. The next two parts of the assignment featured a square tile which was blue except for a centered red dot. In the second part of the assignment I mimicked the solution of the problem of finding the area of the tile which was blue. In the third part I mimicked the computational rendering of the tile. The last two parts of the assignment featured an image consisting of 5 concentric squares. In the fourth part of this assignment I rendered the image. In the fifth part I computed a percentage based on the concentric squares image. Throughout the problem solving parts of this assignment the concept of binding values to variables was a predominant theme.

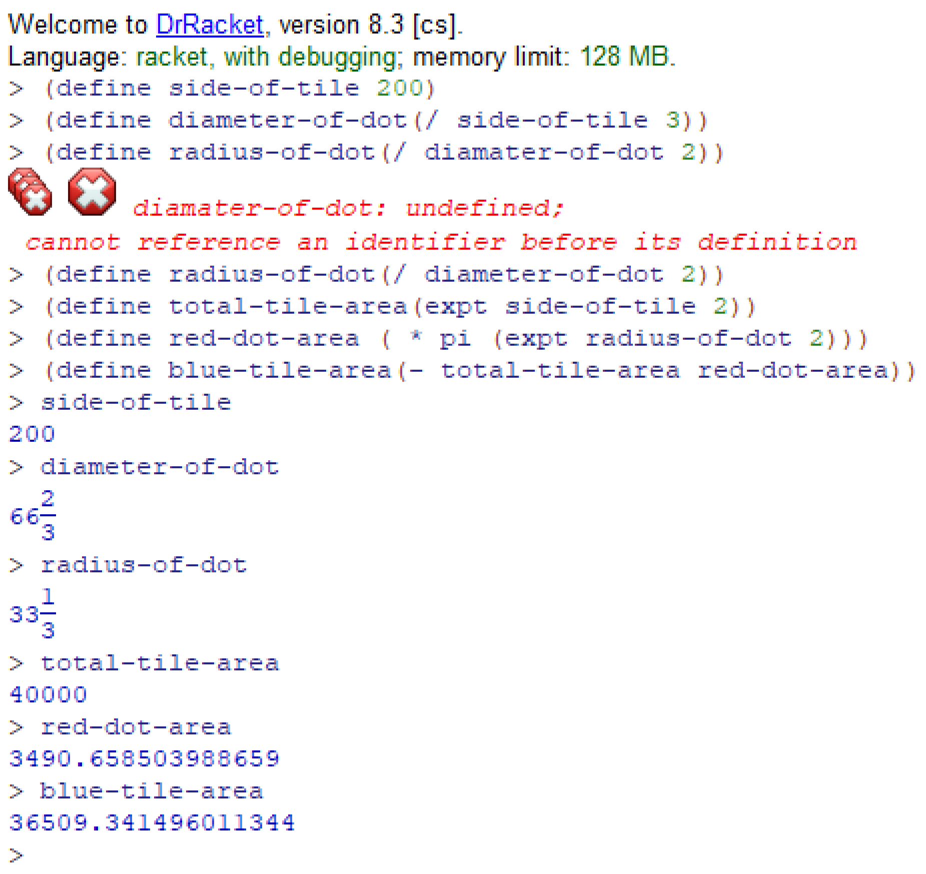
Interaction: Simple Numeric Processing



```
Welcome to DrRacket, version 8.3 [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
> (exp 2 8)
exp: arity mismatch;
 the expected number of arguments does not match the given number
  expected: 1
  given: 2
> (expt 2 8)
256
> (* pi(expt 7 2))
153.93804002589985
> (expt 9 50)
515377520732011331036461129765621272702107522001
>
```

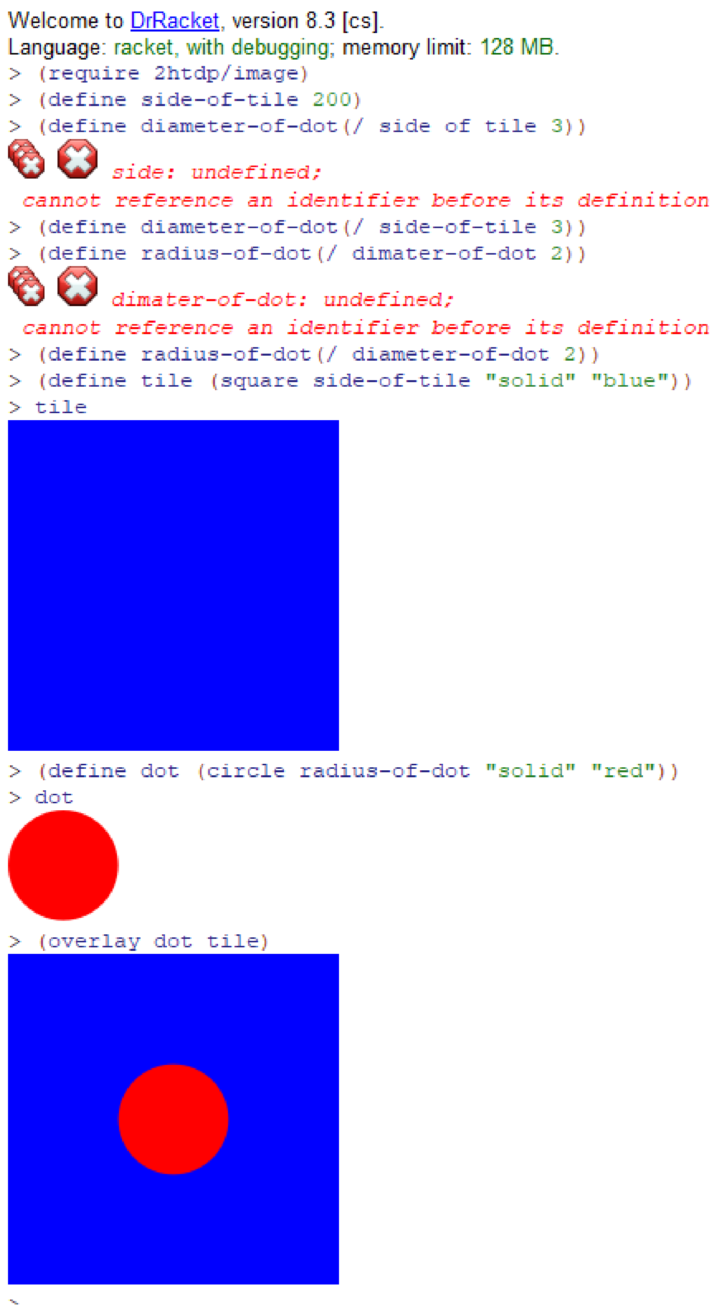
Interaction: Solution to the blue and red tile area problem

The blue and red tile area problem: A tile of side 200 is blue, except for a centered red disk of radius one-third the side of the tile. What is the area of the tile which is blue?



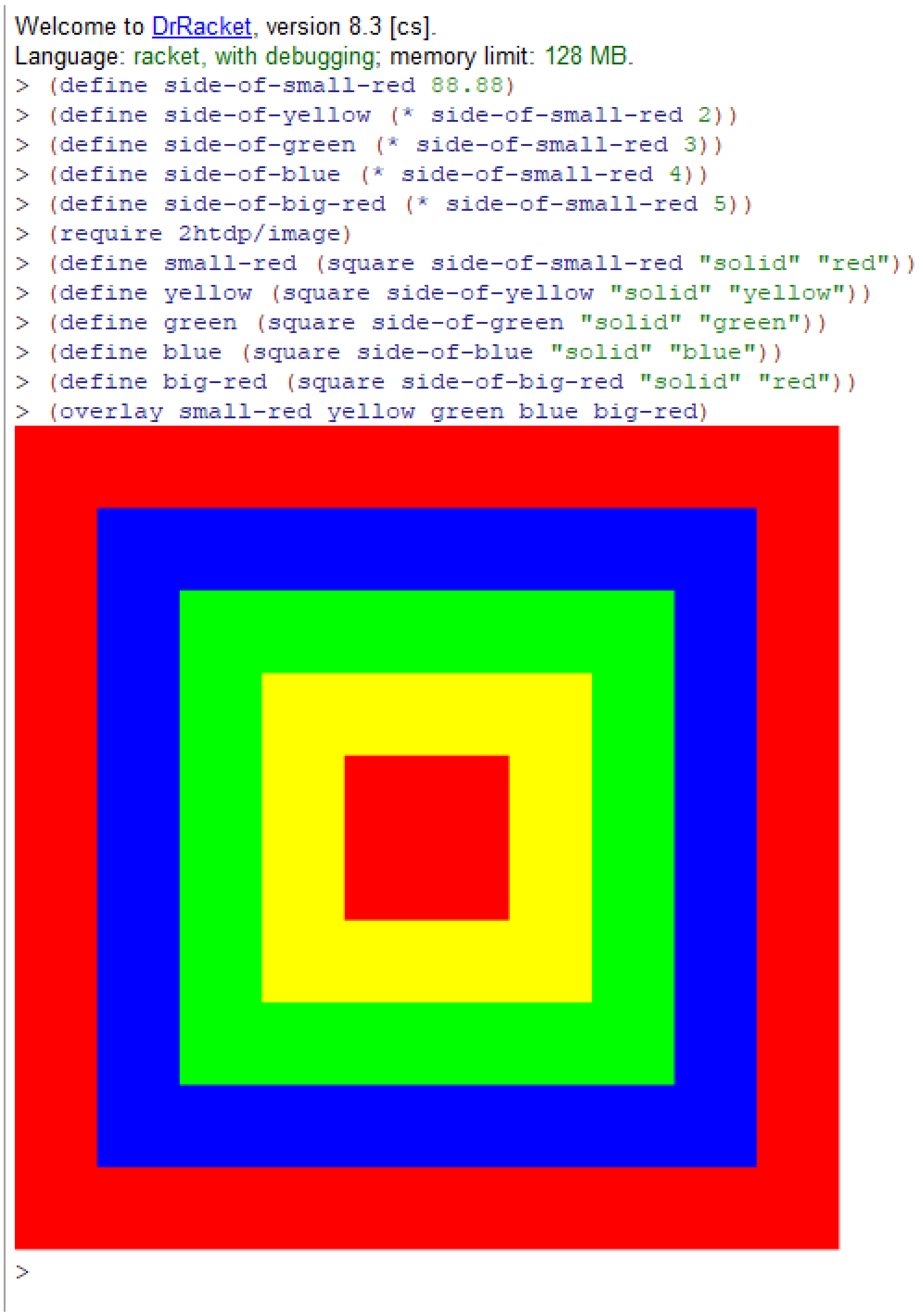
```
Welcome to DrRacket, version 8.3 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (define side-of-tile 200)
> (define diameter-of-dot(/ side-of-tile 3))
> (define radius-of-dot(/ diamater-of-dot 2))
      diamater-of-dot: undefined;
 cannot reference an identifier before its definition
> (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 2/3
> radius-of-dot
33 1/3
> total-tile-area
40000
> red-dot-area
3490.658503988659
> blue-tile-area
36509.341496011344
>
```

Interaction: Painting the blue and red tile



```
Welcome to DrRacket, version 8.3 [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))
  side: undefined;
 cannot reference an identifier before its definition
> (define diameter-of-dot(/ side-of-tile 3))
> (define radius-of-dot(/ dimater-of-dot 2))
  dimater-of-dot: undefined;
 cannot reference an identifier before its definition
> (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)
>
```

Interaction: Painting the concentric squares image

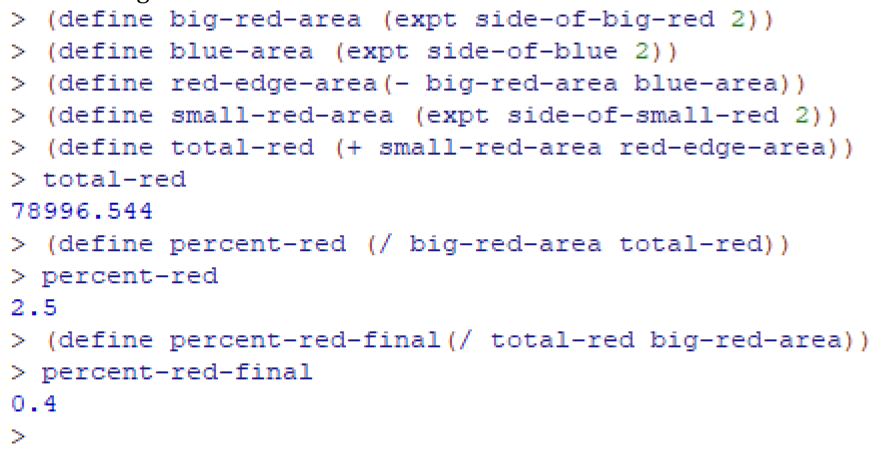


```
Welcome to DrRacket, version 8.3 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (define side-of-small-red 88.88)
> (define side-of-yellow (* side-of-small-red 2))
> (define side-of-green (* side-of-small-red 3))
> (define side-of-blue (* side-of-small-red 4))
> (define side-of-big-red (* side-of-small-red 5))
> (require 2htdp/image)
> (define small-red (square side-of-small-red "solid" "red"))
> (define yellow (square side-of-yellow "solid" "yellow"))
> (define green (square side-of-green "solid" "green"))
> (define blue (square side-of-blue "solid" "blue"))
> (define big-red (square side-of-big-red "solid" "red"))
> (overlay small-red yellow green blue big-red)
```

```
>
```

~~Interaction: Computing the percent of the concentric squares image which is red~~

→ the work goes here ←



```
> (define big-red-area (expt side-of-big-red 2))
> (define blue-area (expt side-of-blue 2))
> (define red-edge-area(- big-red-area blue-area))
> (define small-red-area (expt side-of-small-red 2))
> (define total-red (+ small-red-area red-edge-area))
> total-red
78996.544
> (define percent-red (/ big-red-area total-red))
> percent-red
2.5
> (define percent-red-final(/ total-red big-red-area))
> percent-red-final
0.4
>
```

Total percent red is 40%