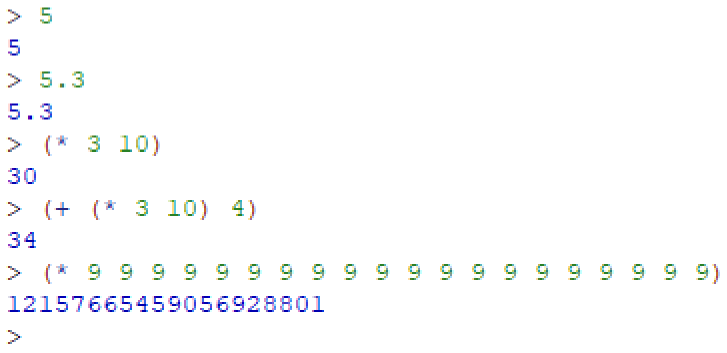
First Racket Programming Assignment Solution

During this assignment I was introduced to the Racket language and the DrRacket PDE. I performed various mathematical functions, graphically displayed shapes in the PDE, and calculated the area of multiple overlapping shapes.

Simple Numeric Processing

(a)

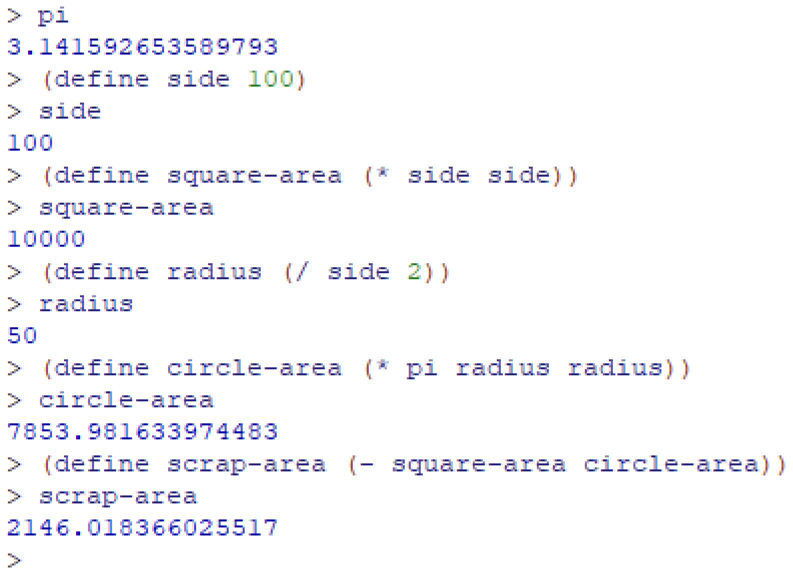


```
> 5
5
> 5.3
5.3
> (* 3 10)
30
> (+ (* 3 10) 4)
34
> (* 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9)
12157665459056928801
>
```

Solution to the Scrap Problem

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

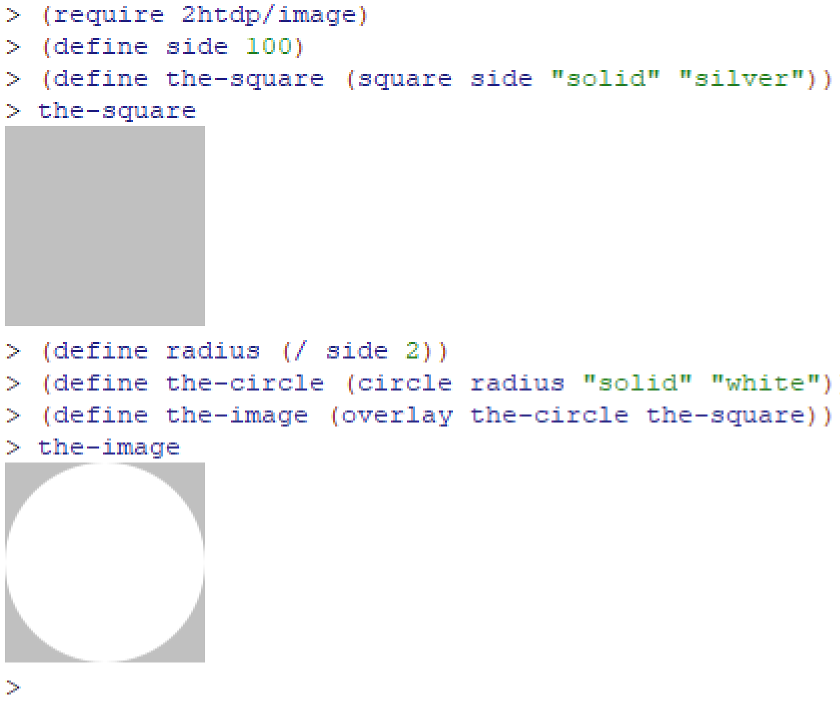
(b)



```
> pi
3.141592653589793
> (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
>
```

Illustration of Scrap Problem Situation

(c)



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

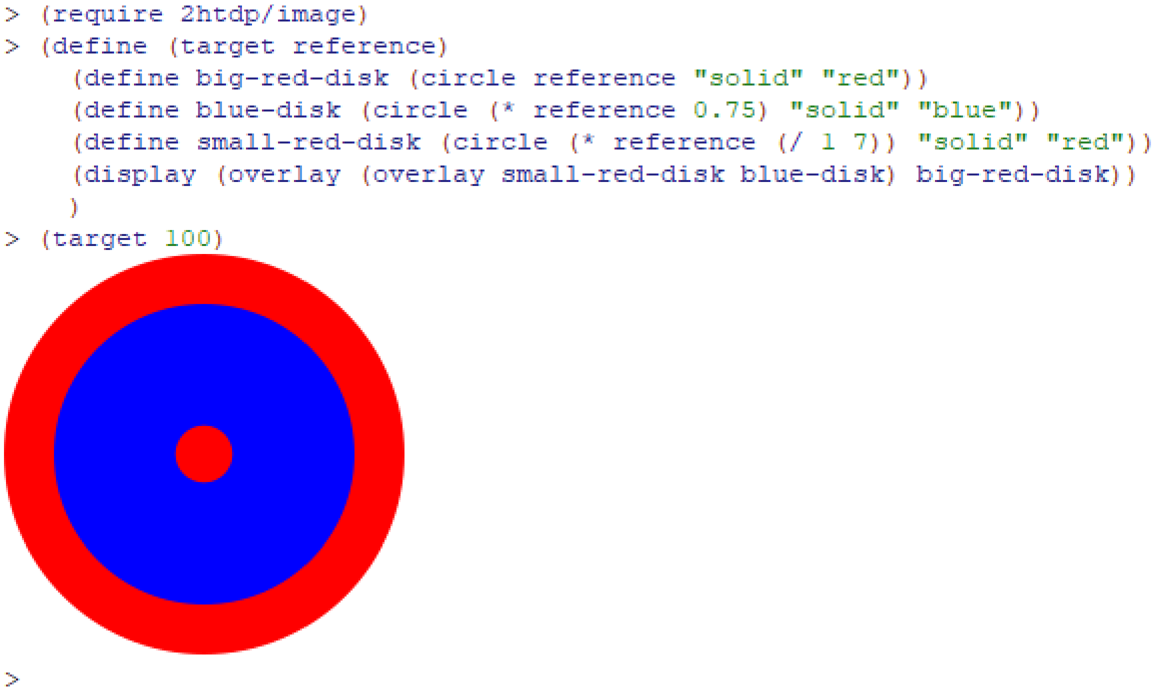
```
> (define radius (/ side 2))
> (define the-circle (circle radius "solid" "white"))
> (define the-image (overlay the-circle the-square))
> the-image
```

```
>
```

Illustration of the Target Problem Situation

The Target: A “target” consists of a red disc of some diameter, containing a blue disc of diameter ¾ that of the bigger disc, which, in turn, contains another red disc, this one of diameter 1/7 that of the biggest disc.

(d)



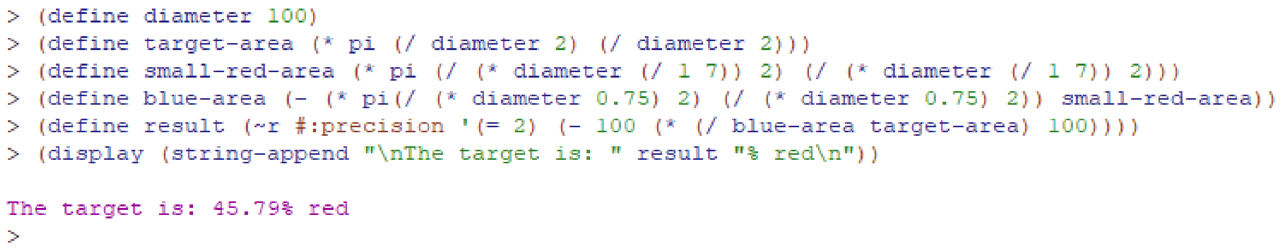
```
> (require 2htdp/image)
> (define (target reference)
    (define big-red-disk (circle reference "solid" "red"))
    (define blue-disk (circle (* reference 0.75) "solid" "blue"))
    (define small-red-disk (circle (* reference (/ 1 7)) "solid" "red"))
    (display (overlay (overlay small-red-disk blue-disk) big-red-disk))
    )
> (target 100)
```

```
>
```

Solution to Target Problem

Target Problem: What percentage of the target is red?

(e)



```
> (define diameter 100)
> (define target-area (* pi (/ diameter 2) (/ diameter 2)))
> (define small-red-area (* pi (/ (* diameter (/ 1 7)) 2) (/ (* diameter (/ 1 7)) 2)))
> (define blue-area (- (* pi(/ (* diameter 0.75) 2) (/ (* diameter 0.75) 2)) small-red-area))
> (define result (~r #:precision '(= 2) (- 100 (* (/ blue-area target-area) 100))))
> (display (string-append "\nThe target is: " result "% red\n"))

The target is: 45.79% red
>
```