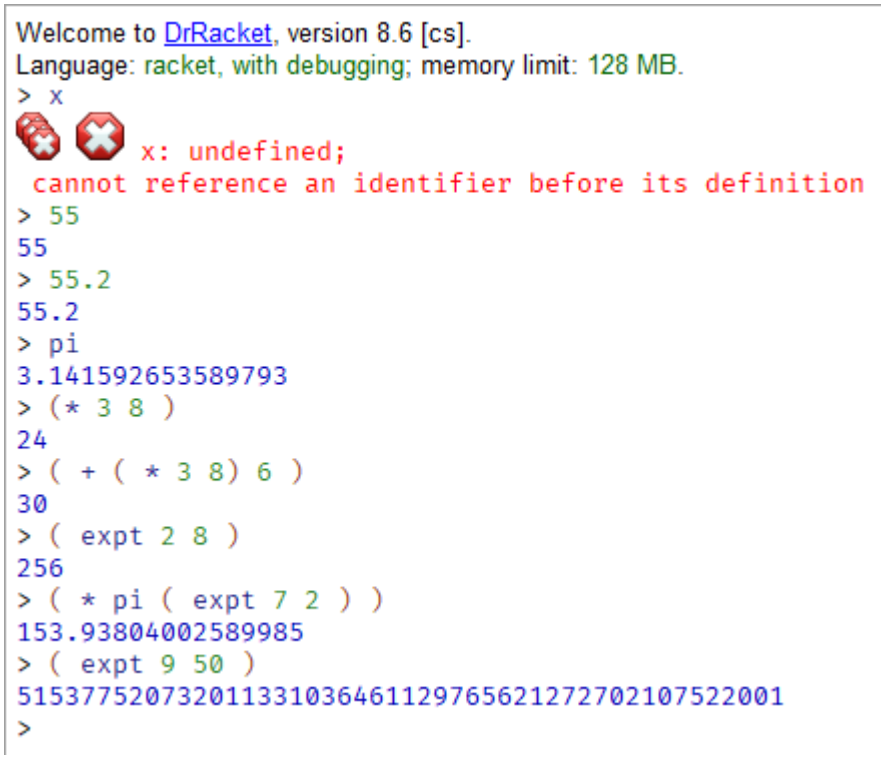
Racket Programming Assignment 1: First Interactions

Abstract

This assignment serves the purpose of getting used to Racket, as I've never used a functional programming language before. Exploring the limits of the environment and just messing around with different functions allows me the time to get comfortable wth the language before diving deep into something more complex and higher stakes.

Interaction 1: Simple Numeric Processing



```
Welcome to DrRacket, version 8.6 [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
> ( expt 2 8 )
256
> ( * pi ( expt 7 2 ) )
153.93804002589985
> ( expt 9 50 )
515377520732011331036461129765621272702107522001
>
```

These interactions showcase simple arithmetic processing. I find it interesting that the addition and multiplication operations shown use the exact same syntax as if you are running a function, such as `expt`.

Interaction 2: The blue and red tile area problem

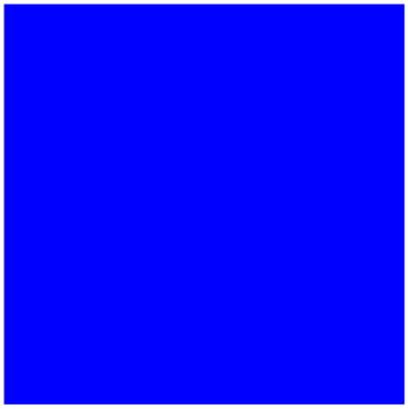
Problem: A tile of side 200 is blue, except for a centered red disc with a radius one-third the side of the tile. what is the area of the tile which is blue?

```
Welcome to DrRacket, version 8.6 [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 ( / 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 2 continued: Painting the blue and red tile

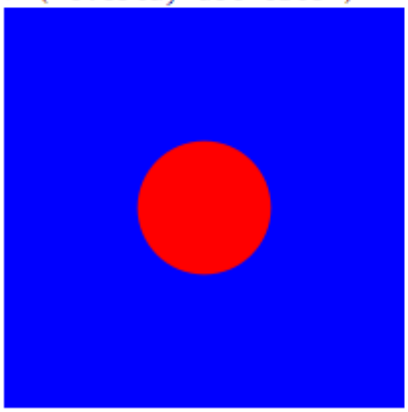
Instead of starting from scratch and typing all of our variables again, I'll simply require `2htdp/image` to our current session and paint our tiles

```
> blue-tile-area
36509.341496011344
> ( require 2htdp/image )
> ( define tile ( square side-of-tile 'solid 'blue ) )
> tile
```



```
> ( define dot ( circle radius-of-dot 'solid 'red ) )
> dot
```

```
> ( overlay dot tile )
```



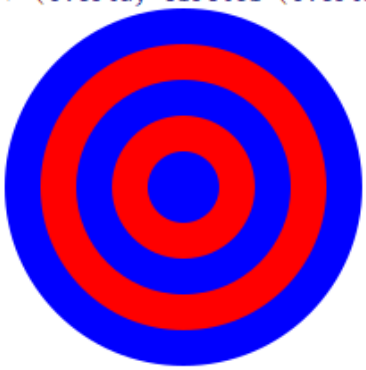
```
> 
```

Interaction 3: Painting Concentric Circles

The following image was created by overlaying a blue disk of radius 20 on top of a red disk of radius 40 on top of a blue disk of radius 60 on top of a red disk of radius 80 on top of a blue disk of radius 100.

```
1  #lang racket
2  ( require 2htdp/image )
3  ( define smallest-circle-radius 20)
4  ( define circle2-radius ( * smallest-circle-radius 2 ) )
5  ( define circle3-radius ( * smallest-circle-radius 3 ) )
6  ( define circle4-radius ( * smallest-circle-radius 4 ) )
7  ( define circle5-radius ( * smallest-circle-radius 5 ) )
8
9  ( define circle1 (circle smallest-circle-radius 'solid 'blue ) )
10 ( define circle2 (circle circle2-radius 'solid 'red ) )
11 ( define circle3 (circle circle3-radius 'solid 'blue ) )
12 ( define circle4 (circle circle4-radius 'solid 'red ) )
13 ( define circle5 (circle circle5-radius 'solid 'blue ) )
```

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (overlay circle1 (overlay circle2 (overlay circle3 (overlay circle4 circle5))))
```



```
>
```

Interaction 3 continued: Compute the blue area

```
1  #lang racket
2  ( require 2htdp/image )
3  ( define smallest-circle-radius 20)
4  ( define circle2-radius ( * smallest-circle-radius 2 ) )
5  ( define circle3-radius ( * smallest-circle-radius 3 ) )
6  ( define circle4-radius ( * smallest-circle-radius 4 ) )
7  ( define circle5-radius ( * smallest-circle-radius 5 ) )
8
9  ( define circle1 (circle smallest-circle-radius 'solid 'blue ) )
10 ( define circle2 (circle circle2-radius 'solid 'red ) )
11 ( define circle3 (circle circle3-radius 'solid 'blue ) )
12 ( define circle4 (circle circle4-radius 'solid 'red ) )
13 ( define circle5 (circle circle5-radius 'solid 'blue ) )
14
15 ( define (get-area radius) ( * pi ( expt radius 2 ) ) )
16
17 ( define circle1-area ( get-area smallest-circle-radius ) )
18 ( define circle2-area ( get-area circle2-radius ) )
19 ( define circle3-area ( get-area circle3-radius ) )
20 ( define circle4-area ( get-area circle4-radius ) )
21 ( define circle5-area ( get-area circle5-radius ) )
```

```
Welcome to DrRacket, version 8.6 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> ( define outer-blue-ring ( - circle5-area circle4-area ) )
> ( define inner-blue-ring ( - circle3-area circle2-area ) )
> ( define total-blue-area ( + circle1-area inner-blue-ring outer-blue-ring ) )
> outer-blue-ring
11309.733552923255
> inner-blue-ring
6283.185307179586
> total-blue-area
18849.55592153876
>
```