Contracts

Contracts tell us how to use a function. For example: `ellipse :: (Number, Number, String, String) -> Image` tells us that the name of the function is `ellipse`, it takes four inputs (two Numbers and two Strings), and it evaluates to an `Image`. From the contract, we know `ellipse(100, 50, "outline", "red")` will evaluate to an `Image`.

Name		Domain		Range
`# num-sqr`	`::`	`(Number)`	`->`	`Number`
`num-sqr(9)`				
`# num-sqrt`	`::`	`(Number)`	`->`	`Number`
`num-sqrt(25)`				
`# star`	`::`	`(Number, String, String)`	`->`	`Image`
`star(50, "solid", "teal")`				
`# circle`	`::`	`(Number, String, String)`	`->`	`Image`
`circle(30, "outline", "fuchsia")`				
`# triangle`	`::`	`(Number, String, String)`	`->`	`Image`
`triangle(80, "solid", "darkgreen")`				
`# square`	`::`	`(Number, String, String)`	`->`	`Image`
`square(10, "outline", "red")`				
`# rectangle`	`::`	`(Number, Number, String, String)`	`->`	`Image`
`rectangle(20, 80, "solid", "gold")`				
`# ellipse`	`::`	`(Number, Number, String, String)`	`->`	`Image`
`ellipse(30, 70, "outline", "lightblue")`				
`# regular-polygon`	`::`	`(Number, Number, String, String)`	`->`	`Image`
`regular-polygon(8, 40, "solid", "red")`				
`# radial-star`	`::`	`(Number, Number, String, String)`	`->`	`Image`
`radial-star(17, 50, 10, "solid", "orange")`				