Contracts

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

Name		Domain		Range
`; +`	`::`	`Number, Number`	`->`	`Number`
`(+ 3 2)`				
`; -`	`::`	`Number, Number`	`->`	`Number`
`(- 5 3)`				
`; *`	`::`	`Number, Number`	`->`	`Number`
`(* 2 4)`				
`; /`	`::`	`Number, Number`	`->`	`Number`
`(/ 8 2)`				
`; sqrt`	`::`	`Number`	`->`	`Number`
`; (sqrt 25)`				
`; sqr`	`::`	`Number`	`->`	`Number`
`; (sqr 5)`				
`; string-length`	`::`	`String`	`->`	`Number`
`(string-length "Rainbow")`				
`; <`	`::`	`Number, Number`	`->`	`Boolean`
`(< 3 2)`				
`; >`	`::`	`Number, Number`	`->`	`Boolean`
`(> 3 2)`				

Contracts

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

Name		Domain		Range
`; =`	`::`	`Number, Number`	`->`	`Boolean`
`(= 3 2)`				
`; <=`	`::`	`Number, Number`	`->`	`Boolean`
`(<= 3 2)`				
`; >=`	`::`	`Number, Number`	`->`	`Boolean`
`(>= 3 2)`				
`; <>`	`::`	`Number, Number`	`->`	`Boolean`
`(<> 3 2)`				
`; string=?`	`::`	`String, String`	`->`	`Boolean`
`(string=? "cat" "kitten")`				
`; string>=?`	`::`	`String, String`	`->`	`Boolean`
`(string>=? "ape" "zebra")`				
`; string<=?`	`::`	`String, String`	`->`	`Boolean`
`(string<=? "Abena" "Zoe")`				
`; string<>?`	`::`	`String, String`	`->`	`Boolean`
`(string<>? "crab" "crawfish")`				
`; string-append`	`::`	`String, String`	`->`	`String`
`(string-append "sun" "shine")`				

Contracts

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

Name		Domain		Range
`; triangle`	`::`	`Number, String, Sting`	`->`	`Image`
`(triangle 80 "solid" "green")`				
`; star`	`::`		`->`	
`; circle`	`::`		`->`	
`; square`	`::`		`->`	
`; rectangle`	`::`		`->`	
`; text`	`::`		`->`	
`; ellipse`	`::`		`->`	
`; regular-polygon`	`::`		`->`	
`; rhombus`	`::`		`->`	

Contracts

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

Name		Domain		Range
`; right-triangle`	`::`		`->`	
`; isosceles-triangle`	`::`		`->`	
`; radial-star`	`::`		`->`	
`; star-polygon`	`::`		`->`	
`; triangle/sas`	`::`		`->`	
`; triangle/asa`	`::`		`->`	
`; image-url`	`::`		`->`	
`(image-url "https://www.bootstrapworld.org/images/icon.png")`				
`; scale`	`::`		`->`	
`; rotate`	`::`		`->`	

Contracts

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

Name		Domain		Range
`; overlay`	`::`		`->`	
`; put-image`	`::`		`->`	
`; flip-horizontal`	`::`		`->`	
`; flip-vertical`	`::`		`->`	
`; above`	`::`		`->`	
`; beside`	`::`		`->`	
`; or`	`::`		`->`	
`; and`	`::`		`->`	`Boolean`
`;`	`::`		`->`	
`;`				