General

1. **Inbuilt Functions**
2. **List Comprehensions**
3. **Make your own function**

- Double Data type = Real numbers
- Uses Lazy Evaluation instead of Eager Evaluation
- Outer -> Inner
- In general, lazy does more work, but will always return an answer if an answer can be returned
- Every value has exactly 1 type
- [Char] = String

Number Data types				
Integer	Int	Rational	Float	Double
Unbounded	(-2^29) to (2^29 – 1)	Unbounded	Single	Double
Integers		Rational Numbers		

- -> is right-associative
- **All functions only take in one argument!**
- All variables start with lowercase
- Specific Data types start with uppercase
- Curried function: a function that takes one argument and returns another function
- Higher Order Function: A function that takes in a function
- To force Haskell to compute a value:
    - Pattern Matching
    - Strictness Annotations (!)
        - Use this to make program use less memory
        - Memory allocation and deallocation takes time
        - So this saves time

- `f % x = % f x = (% x) f = (f %) x`, where `%` is an infix operator

User-defined Data types

```
data Victor where
  Nathan :: Victor
  Suresh :: Victor
  etc
```

`data Victor = Nathan | Suresh | etc`

Cardinality

|Victor| = Cardinality of Victor = Number of elements in Victor

Where clause

```
fun a
  | even a    = x
  | otherwise = x + 6
  where x = mod a 5
```

- Introduces a value locally
- Evaluate values once, result is shared

Let Clause

```
fun a
  | even a    = let x = mod a 5 in x
  | otherwise = let x = mod a 5 in x + 6
```