```haskell
--TYPES AND TYPECLASSES--

-- Int = number from -2147483647 to 2147483647
addThree :: Int -> Int -> Int -> Int
addThree x y z = x+y+z

-- Integer = really big numbers with no limit
factorial :: Integer -> Integer
factorial n = product [1..n]

-- Float = Number with a set precision. Out to the one hundred thousandths place.
circumference :: Float -> Float
circumference n = 2 * pi * n

-- Double = Number with a set precision twice the precision as "Float"
circumference' :: Double -> Double
circumference' n = 2 * pi * n

-- Bool is a boolean type that only returns True or False
booleans :: Bool -> Bool
booleans bool = not bool

-- Char is a character denotated by single quotes
capsCharacters :: String -> Char
capsCharacters st = if length st > 0 then head [c | c <- st, c `elem` ['A'..'Z']] else '~'

        -- TYPECLASSES

-- show = showing the String representation of an input or value. So: 4 -> "4"
showThing thing = show thing

-- read = getting the type value of a certian input of value. Opposite of `show`. So: "[1,42,5]" -> [1,42,5].
-- use :: [type] to denote the type of the output. So like :: Int or :: Char or even :: [Int] or :: (Int, Char)
readInt thing = read thing :: Int

-- Integral = typeclass that contains Integer and Int
integralExample :: (Show a, Integral a) => a -> String
integralExample var = show var

factorial :: (Integral a) => a -> a
factorial 0 = 1
factorial n = n * factorial (n - 1)
```