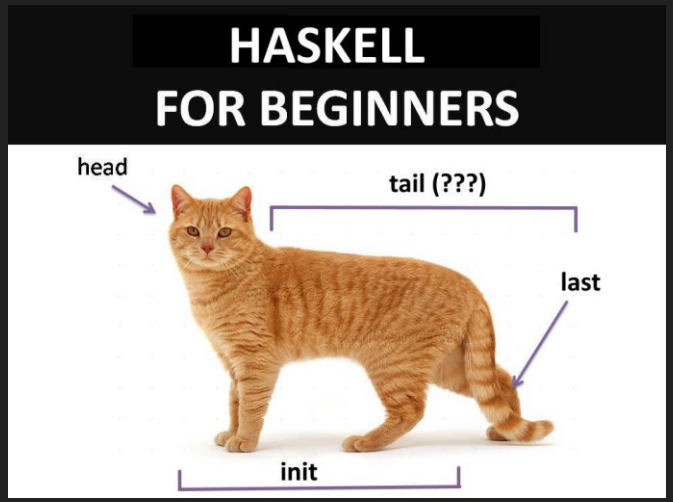
FIT2102 Programming Paradigms Tutorial 7

Maybe? Nothing...



Faculty of Information Technology

Housekeeping

- Assignment due on friday!
- Make sure you have your work open from last week.
- Cameras :)

Lambda functions in Haskell

A lambda function in haskell looks like: `\x -> <some expression of x>`
Compare to a lambda in JavaScript: `x => <some expression of x>`
and lambda calculus: `λx . <some expression of x>`

```
> map (\x->2*x) [1..4]
[2,4,6,8]
```

...but often we can avoid explicit lambdas with partially applied functions (to achieve a point-free style):

```
> map (2*) [1..4]
[2,4,6,8]
```

Typeclasses in Haskell

A typeclass is a sort of interface that defines some behavior. If a type is a part of a typeclass, that means that it supports and implements the behavior the typeclass describes.

http://learnyouahaskell.com/types-and-typeclasses

Basic Typeclasses

The Eq Typeclass

```
(==) :: Eq a => a -> a -> Bool
```

The Ord Typeclass

```
(<=) :: Ord a => a -> a -> Bool
```

The Num Typeclass

```
5 :: Num p => p
```

Why are typeclasses useful?

The real usefulness comes with the functions that act as the interface for typeclasses, such as Eq, namely `==` and `/=`

If a type is a part of the Eq typeclass, we can use the == functions with values of that type.

That's why expressions like 4 == 4 and "foo" /= "bar" typecheck!

An example...

```
class Named a where
    name :: a -> String

data Person = Person String String

instance Named Person where
    name (Person first last) = first ++ " " ++ last

bruce = Person "Bruce" "Willis"

> name bruce
"Bruce Willis"
```

Pattern matching Maybes

```
printNumber name = msg $ lookup name phonebook
   where
       msg (Just number)  = print number
       msg Nothing        = print $ name ++ " not found in database"
```

We can pattern match *Just a* or *Nothing*

```
> printNumber "Fred"

"01624 556442"

> printNumber "Tim"

"Tim not found in database"
```