**Haskell. Fast, Purely Functional, Lazy. Strongly Typed, Pattern Matching.**

**Download** = <http://www.haskell.org/platform/> | apt-get|yum install haskell-platform

**IDE:** eclipsefp, visualhaskell **Package manager:** cabal **Repl:** ghci

**Learn For Free:** #haskell on freenode irc, Learn you a haskell, Real world haskell, wikibooks haskell, tryhaskell.org

**Search by Type:** Haskell.com/hoogle/

**Types:** Int, Integer, Float, Bool, Char, String, (Tuples, ..), [Lists, ..] (these can only be of the same type)

**Quirks: “**-1” means we partially apply the “-” function with “1” as first arg. (-1) = negative 1.

Instead of saying myFunc(x,y) we say myFunc x y. Instead of g(f(X)) we say “g . f x”.

But what if we want fn(x) as an argument of f? “f (fn x)” works and “f $ fn x” is the shorthand.

|  |  |
| --- | --- |
| -- hello.hs -- This is a comment  main = putStrLn "Hello, World!" | > ghc -o hello hello.hs  > ./hello (hello.exe on windows) |

|  |  |  |
| --- | --- | --- |
| let a=1; b=2 in a+b == 3  let a=1  b=2 in a+b == 3 | isEmptyList xs = case xs of  [] -> True  (first: rest) -> False | isEmptyList [] = True  isEmptyList xs = False |

bodyMassIndex:: (RealFloat a) => a -> a -> String

This is a function signature. The function name is bodyMassIndex. The last “->” value is always the return value, in this case “String”, the previous two values must be the function arguments. “a” is a type variable, indicating the two arguments will be of the same type “RealFloat”.

|  |  |  |
| --- | --- | --- |
| **Partial Application**  plusOne=(+) 1  **List Comprehension**  [x\*2 | x <-[1..3]] == [1,2,3] | **Sequences**  [1..10] => [1,2,3,4,5,6,7,8,9,10]  [1,3..10] => [1,3,5,7,9]  [1,3..] => [1,3,5,..] infinite seq | **User defined types**  data Color = Red | Green | Blue |  Indigo | Violet  “hello “ ++ “world”==“Hello World” |

|  |  |  |  |
| --- | --- | --- | --- |
| **GHCI**  :? = Help  it = last value evaluated  :type it = last type  :m +Data.Int=add module  :load xyz = load xyz.hs | let x=1 --Define var or Fn  (-3) == negative 3.  -3 == fn x = x-3  type String=[Char] | **Lambdas**  addOne = (\x -> x + 1)  (\x -> x + 1) 4 == 5 | != is /= in haskell  “!true” is “not True” in hs  2 ^ 3 = 8  \*\* = float version of ^ |

**map ((\*) 2) [1..5]** == Use the function x\*2 on each member of a list from 1 to 5 and return a new list.

**foldl (+) 0 [1..5]** == Snowball the list 1 to 5 into one output value using the (+) operator.

**elem 5 in [1..10]** == is the number 5 in the range of 1 to 10?

**reverse . take 3 $ “foobar”** == “oof”. Take the first 3 chars from “foobar” which is “foo”, then reverse them

**reverse . take 3 “foobar”** == This fails. composing reverse and take together expects both functions to take 1 arg. If we

provide two arguments to take, we have filled all arguments and we cannot compose the two functions.

**Record syntax (pretty much a struct but it generates getter helper functions)**

data Person = Person { firstName :: String, lastName :: String, age :: Int} deriving (Show)

