Introduction

Haskell → purely functional programming language. In imperitive langs., a comp. is given a seq. of tasks which it executes. While executing these tasks, it can change state, meaning, a variable $a = 5$, can be changed to a different value, at a different point in the execution of the program.

↓

In purely functional programming, we don't tell the computer what to do as such but rather we tell it what stuff is. i.e. The factorial of a no. is the product of all no.s from 1 to that no., or the sum of a list of a numbers is the first no. plus the sum of all other no.s and so on.

↓

we express things in form of functions.

↓

we can't also set a variable to something and then set it to something else, meaning functions have no side effects. The only thing that a function can do is to calculate something and return it as a result.

↘

if a fn. is called with same parameters, it is guaranteed to return the same result always.

↓

called referential transparency

↓

This allows us to easily deduce (and prove) that a fn. is correct and then build more complex fns. by gluing simple functions together.

↓

In Haskell, programs can be thought of as a series of transformations on data. Due to its lazy nature, Haskell won't execute fns. and calculate things until it's really forced to show us a result.

↓

Statically typed i.e. when we compile our program, the compiler knows which piece of code is of what type.

↓

Type inference i.e. we don't explicitly have to label every piece of code with a type because the type system can intelligently figure it out. Type inference also allows our code to be more general.

•) What we need to dive in : text editor & Haskell compiler

↓

GHC

(takes .hs script & compiles it)

↓

GHCI → interactive mode.

any fns. written in an external file can be loaded using

:l myfunctions.

The functions library file should be in the same file where ghci was invoked

prospective workflow: defining some functions in a .hs file, loading it up and messing around with them and then changing the .hs file, loading it again and so on.