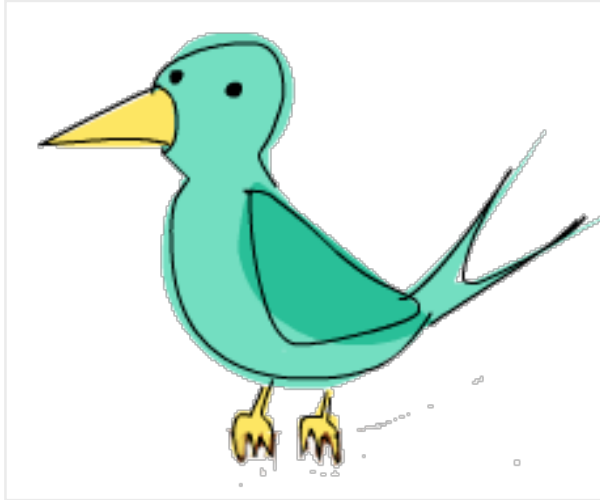


Introduction to Haskell



1-imperative programming languages (C, C++, Java, Python ...) (usually things done by giving the computer a sequence of tasks and then it executes them.)

2-functional language before (Haskell, ML, OCaml ...) (ou don't tell the computer what to do as such but rather you tell it what stuff *is*.)



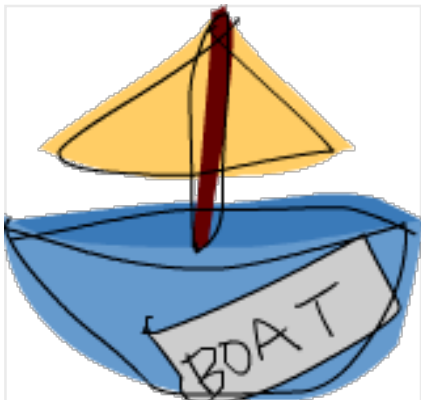
The only thing a function can do is calculate something and return it as a result.

Why Haskell is lazy?



That means that unless specifically told otherwise, Haskell won't execute functions and calculate things until it's really forced to show you a result.

Haskell is statically typed ?

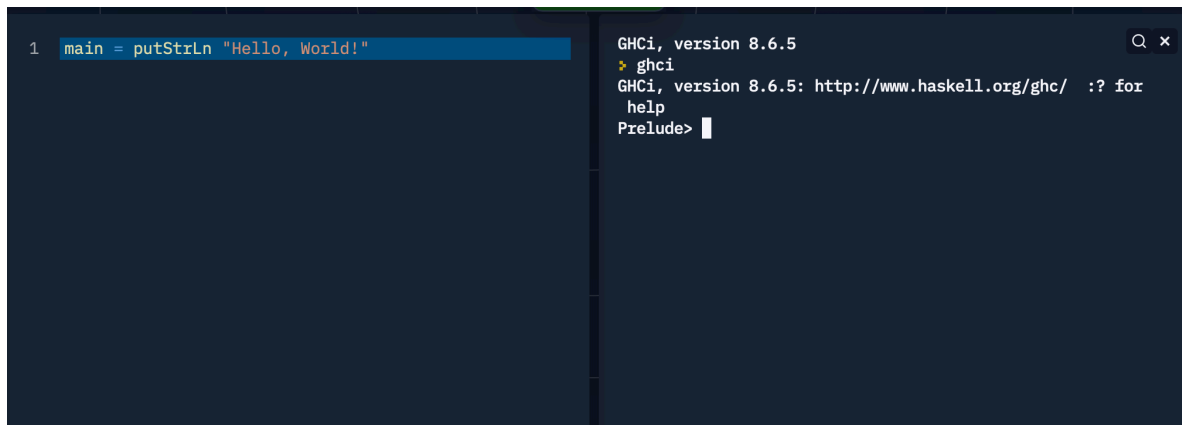


When you compile your program, the compiler knows which piece of code is a number, which is a string and so on.

Haskell is elegant and concise?

Haskell programs are usually shorter than their imperative equivalents.

A text editor and a Haskell compiler ?



The image shows a dark-themed interface with two panels. The left panel is a text editor with a single line of Haskell code: `1 main = putStrLn "Hello, World!"`. The right panel is a terminal window for the Haskell Interactive Compiler (GHCi), version 8.6.5. It shows the prompt `Prelude>` with a cursor. Above the prompt, there are search and close icons, and some text including `ghci`, the GHCi version, a URL `http://www.haskell.org/ghc/`, and a prompt `:? for help`.

GHC can take a Haskell script (they usually have a `.hs` extension) and compile it but it also has an interactive mode which allows you to interactively interact with scripts. For learning it's a lot easier and faster than compiling every time you make a change and then running the program from the prompt.

The interactive mode is invoked by typing in **ghci** at your prompt. If you have defined some functions in a file called, say, **myfunctions.hs**, you load up those functions by typing in **:l myfunctions** and then you can play with them, provided **myfunctions.hs** is in the same folder from which **ghci** was invoked.

If you change the `.hs` script, just run **:l myfunctions** again or do **:r**, which is equivalent because it reloads the current script. The usual workflow for me when playing around in stuff is defining some functions in a `.hs` file, loading it up and messing around with them and then changing the `.hs` file, loading it up again and so on.