Functional Programming

Paul Mellies

Lecture 1: getting started with Haskell
Get Programming With Haskell

We follow the introductory chapters of this nice book on Haskell:

The content of the book is freely accessible at

https://www.manning.com/books/get-programming-with-haskell
Glasgow Haskell – Interactive Mode

You will find the Glasgow Haskell Compiler (GHC) here:

https://www.haskell.org/downloads

Once you have installed GHC in your computer, type on your terminal:

> ghci

You will then see the following message on your screen:

GHCi, version 7.10.3:
http://www.haskell.org/ghc/ :? for help
Glasgow Haskell – Interactive Mode

You can try the interactive mode and play with it:

```
ghci > 2 + 15
17
ghci > 49 * 100
4900
ghci > 5 / 2
2.5

ghci > 2^10
1024
ghci > 2^158
365375409332725729550921208179070754913983135744
```
Glasgow Haskell – Interactive Mode

ghci > let x = 2 + 2
ghci > x
4

ghci > let doubleMe x = x + x
ghci > doubleMe 7
14
ghci > let doubleUs x y = doubleMe x + doubleMe y
ghci > doubleUs 3 5
ghci > 16

ghci > [1..20]
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
ghci > ['a'..'z']
"abcdefghijklmnopqrstuvwxyz"
Glasgow Haskell – Hello World!

Type the following program in your terminal

```haskell
--hello.hs my first Haskell file!
main = do
    print "Hello World!"
```

and then store it as a text file:

```
hello.hs
```

which can be then compiled into an executable file.
Hello World!

```
-- hello.hs my first Haskell file!
main = do
  print "Hello World!"
```

A commented line with the name of your file

The start of your ‘main’ function

The main function prints out Hello World!
Hello World!

In order to compile the program, use the command:

```
$ ghc -o hello hello.hs
[1 of 1] Compiling Main
Linking hello ...
```

which will then produce three files

- **hello** called the **executable** file also **hello.exe** in Windows
- **hello.hi** called the **interface** file
- **hello.o** called the **object** file also **hello.obj** in Windows
Two ways to load a file in the interactive mode

This can be done externally, in the following way:

```
ghci hello.hs
[1 of 1] Compiling Main
Ok, modules loaded: Main
```

or internally, in the following way:

```
ghci> :l hello.hs
ghci> main
"Hello World!"
```
An example of messy code

```haskell
main :: IO()
main = do
    print "Who is the email for?"
    recipient <- getLine
    print "What is the title of the book?"
    title <- getLine
    print "Who is the author of the email?"
    author <- getLine
    print ("Dear " ++ recipient ++ ",\n" ++ "Thanks for buying " ++ title ++ ".\nSee you soon, \n" ++ author)
```

Do you see why this code is messy? And how you would improve it?
Same Haskell code, improved (part 1)

First, observe that the email message can be constructed independently, using the following purely functional code:

toPart recipient = "Dear " ++ recipient ++ ",\n"
bodyPart title = "Thanks for buying " ++ title ++ ".\n"
fromPart author = "See you soon,\n" ++ author

createEmail recipient title author = toPart recipient ++
    bodyPart title ++
    fromPart author
Then construct the main function using these functions:

```haskell
main :: IO()
main = do
    print "Who is the email for?"
    recipient <- getLine
    print "What is the title of the book?"
    title <- getLine
    print "Who is the author of the mail?"
    author <- getLine
    putStrLn (createEmail recipient title author)
```

Why is this code better than the previous one?