HASKELL *from the very beginning*

In *Haskell from the Very Beginning* John Whitington takes a no-prerequisites approach to teaching the basics of a modern general-purpose programming language. Each small, self-contained chapter introduces a new topic, building until the reader can write quite substantial programs. There are plenty of questions and, crucially, worked answers and hints.

Haskell from the Very Beginning will appeal both to new programmers, and to experienced programmers eager to explore functional languages such as Haskell. It is suitable both for formal use within an undergraduate or graduate curriculum, and for the interested amateur.

JOHN WHITINGTON founded a company which sells software for document processing. He taught functional programming to students of Computer Science at the University of Cambridge. His other books include the textbooks *"PDF Explained"* (O'Reilly, 2012) and *"OCaml from the Very Beginning"* (Coherent, 2013), and the Popular Science book *"A Machine Made this Book: Ten Sketches of Computer Science"* (Coherent, 2016).

HASKELL

from the very beginning

John Whitington

COHERENT PRESS

COHERENT PRESS
Cambridge

Published in the United Kingdom by Coherent Press, Cambridge

First published October 2019
Reprinted with corrections May 2020

A catalogue record for this book is available from the British Library

ISBN 978-0-9576711-3-3 Paperback

Contents

Preface

This book is based on the Author's experience of teaching programming to students in the University of Cambridge supervisions system. In particular, working with students for the first-year undergraduate course "Foundations of Computer Science", lectured for many years by Lawrence C. Paulson.

An interesting aspect of supervising students from a wide range of backgrounds – some with no previous experience at all, taking Computer Science as an additional subject within the Cambridge Natural Sciences curriculum, and some with a great deal of programming experience already – is the level playing field which the functional family of languages (like Haskell) provide. Sometimes, those students with least prior programming experience perform the best.

I have tried to write a book which has no prerequisites – and with which any intelligent undergraduate ought to be able to cope, whilst trying to be concise enough that someone coming from another language might not be too annoyed by the tone.

One caveat: most things in life are small and elegant, or large and unwieldy. Haskell, as practised, is in the unusual position of being large and elegant. This may be the first Haskell book you read, but it will probably not be the last.

Special note to those who have already written programs

When I was a boy, our class was using a word processor for the first time. I wanted a title for my story, so I typed it on the first line and then, placing the cursor at the beginning, held down the space bar until the title was roughly in the middle. My friend taught me how to use the centring function, but it seemed more complicated to me, and I stuck with the familiar way – after all, it worked. Later on, of course, when I had more confidence and experience, I realized he had been right.

When starting a language which is fundamentally different from those you have seen before, it can be difficult to see the advantages, and to try to think of every concept in terms of the old language. I would urge you to consider the possibility that, at the moment, you might be the boy holding down the space bar.

Acknowledgments

The book was improved markedly by the comments of the technical reviewer, Stephen Dolan. Helpful additional review was provided by Michał Gajda, Jim Stuttard, Stuart Kurtz, Krystal Maughan, David Feuer, Gregory Popovitch, Steven Leiva, Charles Parker, and Dirk Markert. Any remaining errors are due solely to the author.

Getting Ready

This book is about teaching the computer to do new things by writing *computer programs*. Just as there are different languages for humans to speak to one another, there are different *programming languages* for humans to speak to computers.

We are going to be using a programming language called **Haskell**. A Haskell system might already be on your computer, or you may have to find it on the internet and install it yourself. We will be using the Glasgow Haskell system. You will know that you have it working when you see something like this:

```
GHCi, version 8.6.1: http://www.haskell.org/ghc/  :? for help
Prelude>
```

Please make sure the version number is at least 8. Haskell is waiting for us to type something. Try typing $\boxed{\text{1}}$ $\boxed{\text{space}}$ $\boxed{\text{+}}$ $\boxed{\text{space}}$ $\boxed{\text{2}}$ followed by the $\boxed{\text{Enter}}$ key. You should see this:

```
GHCi:
Prelude> 1 + 2
3
Prelude>
```

Haskell tells us the result of the calculation. You may use the left and right arrow keys on the keyboard to correct mistakes and the up and down arrow keys to look through a history of previous inputs. To leave Haskell, give the `:quit` command, again followed by $\boxed{\text{Enter}}$:

```
GHCi:
Prelude> :quit
Leaving GHCi.
```

You should find yourself back where you were before. If you make a mistake when typing, you can use the arrow keys on your keyboard to edit the text. To abandon typing, and ask Haskell to forget what you have already typed, enter $\boxed{\text{Ctrl-C}}$ (hold down the $\boxed{\text{Ctrl}}$ key and tap the $\boxed{\text{c}}$ key). This will allow you to start again.

We are ready to begin.