



## Certificate of Achievement

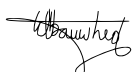
# Roberto Nicolás Savinelli

has completed the following course:

**FUNCTIONAL PROGRAMMING IN HASKELL: SUPERCHARGE YOUR CODING**  
**THE UNIVERSITY OF GLASGOW**

This online course explored the concepts of functional programming using the Haskell language. The course covered standard functional programming techniques, as well as some advanced concepts, which were applied to realistic programming problems.

6 weeks, 4 hours per week



**Wim Vanderbauwhede**  
Senior Lecturer in Computing Science  
The University of Glasgow



**Jeremy Singer**  
Lecturer in Computing Science  
The University of Glasgow



University  
of Glasgow

The person named on this certificate has completed the activities in the attached transcript. For more information about Certificates of Achievement and the effort required to become eligible, visit [futurelearn.com/proof-of-learning/certificate-of-achievement](https://futurelearn.com/proof-of-learning/certificate-of-achievement).

This certificate represents proof of learning. It is not a formal qualification, degree, or part of a degree.



## Roberto Nicolás Savinelli

has completed the following course:

### **FUNCTIONAL PROGRAMMING IN HASKELL: SUPERCHARGE YOUR CODING THE UNIVERSITY OF GLASGOW**

---

This online course explored the concepts of functional programming using the Haskell language. The course covered standard functional programming techniques, as well as some advanced concepts, which were applied to realistic programming problems.

#### **STUDY REQUIREMENT**

6 weeks, 4 hours per week

#### **LEARNING OUTCOMES**

- Develop simple programs involving basic Haskell techniques, including pure function definitions
- Produce definitions of algebraic data types and apply recursion to define functions that traverse such types
- Interpret data structures and function interfaces using types
- Apply formal methods to prove properties of functional programs
- Develop, modify, and explore code using standard Haskell platform tools
- Justify why a program uses common standard monads (including IO and Maybe)
- Explore standard combinators for operating on lists

#### **SYLLABUS**

- Evaluation via expression reduction
- Semantics of function abstraction and application
- Operations involving basic types including integers, characters and booleans
- Definition and traversal of recursive data types including lists and trees
- Techniques for structuring programs of non-trivial size

- Developing custom parsing tools with library support
- Automated testing with the QuickCheck tool
- Infinite data structures and lazy evaluation
- Type classes
- Principles of Lambda calculus
- Monads