

Informatics 1  
Functional Programming Lecture 1

**Introduction**

Don Sannella  
University of Edinburgh

# Welcome to Informatics 1, Functional Programming!

Informatics 1 course organiser: Paul Anderson

Functional programming (Inf1-FP)

Lecturer: Don Sannella

Teaching assistant: Stefan Fehrenbach

Computation and logic (Inf1-CL)

Lecturer: Michael Fourman

Teaching assistant: ???

Informatics Teaching Organization (ITO)

Rob Armitage

# Where to find us

IF – Informatics Forum

AT – Appleton Tower

Inf1 course organiser: [Paul Anderson](mailto:dcspaul@inf.ed.ac.uk) dcspaul@inf.ed.ac.uk IF-1.24

## Functional programming (Inf1-FP)

Lecturer: [Don Sannella](mailto:Don.Sannella@ed.ac.uk) Don.Sannella@ed.ac.uk IF-5.12

Teaching assistant: [Stefan Fehrenbach](mailto:Stefan.Fehrenbach@ed.ac.uk) Stefan.Fehrenbach@ed.ac.uk IF-5.34

Informatics Teaching Organization (ITO):

[Rob Armitage](mailto:Rob.Armitage@ed.ac.uk) Rob.Armitage@ed.ac.uk AT-6.05

# Lectures

- Monday 14:10–15:00, George Square Theatre
- Tuesday 11:10–12:00, George Square Theatre

Sometimes, Inf1-FP swaps lecture slots with Inf1-CL.

# Lectures

- Monday 14:10–15:00, George Square Theatre
- Tuesday 11:10–12:00, George Square Theatre

Sometimes, Inf1-FP swaps lecture slots with Inf1-CL.

*Like next week* — extra Inf1-CL lecture on Tuesday!

# Required text and reading

*Haskell: The Craft of Functional Programming (Third Edition)*,  
Simon Thompson, Addison-Wesley, 2011.

or

*Learn You a Haskell for Great Good!*  
Miran Lipovača, No Starch Press, 2011.

## Reading assignment

This week	Thompson: parts of Chap. 1-3 Lipovača: parts of intro, Chap. 1-2
Later weeks	See the course web page

The assigned reading covers the material very well with plenty of examples.

There will be no lecture notes, just the books. *Get one of them and read it!*

# Linux / DICE Tutorial

Monday	18 September 2017	3–5pm	Appleton Tower 5.05
Tuesday	19 September 2016	2–4pm	Appleton Tower 6.06
Wednesday	20 September 2016	2–4pm	Appleton Tower 6.06
Thursday	21 September 2016	2–4pm	Appleton Tower 6.06

# Get You Installed a Haskell

Monday	18 September 2017	3–5pm	Appleton Tower 5.05
Tuesday	19 September 2016	2–4pm	Appleton Tower 6.06
Wednesday	20 September 2016	2–4pm	Appleton Tower 6.06
Thursday	21 September 2016	2–4pm	Appleton Tower 6.06
<i>Friday</i>	<i>22 September 2016</i>	<i>3–5pm</i>	<i>Appleton Tower 5.05</i>



# Lab Week Exercise and Drop-In Labs

Monday	3–5pm (demonstrator 3–4pm)	Appleton Tower 5.05
Tuesday	2–5pm (demonstrator 3–4pm)	Appleton Tower 6.06
Wednesday	2–5pm (demonstrator 3–4pm)	Appleton Tower 6.06
Thursday	2–5pm (demonstrator 3–4pm)	Appleton Tower 6.06
Friday	3–5pm (demonstrator 3–4pm)	Appleton Tower 5.05

## Lab Week Exercise

submit by 5pm Friday 29 September 2017

*Do all the parts*

# Tutorials

ITO will assign you to tutorials, which start in Week 3.

Attendance is compulsory.

Tuesday/Wednesday      Computation and Logic

*Thursday/Friday*      *Functional Programming*

Contact the ITO if you need to change to a tutorial at a different time.

You *must* do each week's tutorial exercise! Do it *before* the tutorial!

Bring a *printout* of your work to the tutorial!

You may *collaborate*, but you are responsible for knowing the material.

Mark of 0% on tutorial exercises means you have no incentive to *plagiarize*.

But *you will fail the exam if you don't do the tutorial exercises!*

# Beginner-Friendly Tutorials

Some tutorials are labelled as *beginner friendly*.

Ask for one of these if:

- you have no previous programming experience; and/or
- you just aren't so confident

All tutorial exercises will cover the same tutorial exercises.

The beginner-friendly tutorials will proceed more carefully.

Their priority is to make sure that all students are keeping up.

Contact the ITO if you need to change into or out of a beginner-friendly tutorial.

# Automated Feedback

CamlBack (UCLA) will give automated feedback on tutorial exercises.

Use is optional, but it's good to get immediate feedback.

The screenshot displays the CamlBack interface. At the top, there are two tabs: "Problems" and "Feedback". The "Feedback" tab is active, showing a call graph for a function named "fib". The graph consists of three nodes: "fib 2", "fib 1", and "fib 0". The "fib 2" node is a multi-colored circle (red, green, blue), while "fib 1" and "fib 0" are green circles. Two curved lines connect "fib 2" to "fib 1" and "fib 0". A legend in the top right corner explains the colors: a red circle for "incorrect function calls", a green circle for "correct function calls; incorrect result", and a blue circle for "correct function calls and result". Below the graph, there is a section for "Problem 1" containing a code editor with the following code:

```
1 fib :: Integer -> Integer
2 fib 0 = 0
3 fib 1 = 1
4 fib n = fib (n-1) * fib (n-2)
```

At the bottom of the interface, there are buttons for "Check Function" and "Reset", followed by a status message: "Failed a test. Hover over the call nodes above for more information."

# Automated Feedback

CamlBack (UCLA) will give automated feedback on tutorial exercises.

Use is optional, but it's good to get immediate feedback.

Problems Feedback

incorrect function calls  
correct function calls; incorrect result  
correct function calls and result

fib 1

argument(s) : 2  
returned: 0  
expected: 1  
Results of child function calls are not combined properly

fib 2

fib 0

• Problem 1

```
1 fib :: Integer -> Integer
2 fib 0 = 0
3 fib 1 = 1
4 fib n = fib (n-1) * fib (n-2)
```

Check Function ||| Reset ||| Failed a test. Hover over the call nodes above for more information.

# Formative vs. Summative

0%	Lab week exercise
0%	Tutorial 1
0%	Tutorial 2
0%	Tutorial 3
10%	Class Test
0%	Tutorial 4
0%	Tutorial 5
0%	Tutorial 6
0%	Tutorial 7
0%	Mock Exam
0%	Tutorial 8
90%	Final Exam

# Course Webpage

See <http://www.inf.ed.ac.uk/teaching/courses/inf1/jp/> for:

- Course content
- Organisational information: what, where, when
- Annotated lecture slides, reading assignment, *tutorial exercises*, solutions
- Past exam papers
- Programming competition
- Other resources

Any questions?

Questions make you *look good!*

Don's *secret technique* for asking questions.

Don's *secret goal* for this course



# Piazza

Use the Piazza online Inf1-FP forum:

- For *asking questions* outside lectures
- For *reading answers* to questions asked by others
- For *writing answers* to questions asked by others

See the course webpage for the link and for sign-up instructions

Part I

# Functional Programming

# Why learn functional programming in Haskell?

- Important to learn many languages over your career
- Functional languages increasingly important in industry
- Puts experienced and inexperienced programmers on an equal footing
- Operate on data structure *as a whole* rather than *piecemeal*
- Good for concurrency, which is increasingly important

Operating on data structures as a whole rather than piecemeal:

In an imperative language, you often use a loop to operate on the items in a data structure, one at a time.

In a functional programming language, you tend to operate on the whole data structure. Whoosh!

This leads to higher-level thinking about algorithms.

# Linguistic Relativity

“Language shapes the way we think, and determines what we can think about.”

Benjamin Lee Whorf, 1897–1941

“The limits of my language mean the limits of my world.”

Ludwig Wittgenstein, 1889–1951

“A language that doesn’t affect the way you think about programming, is not worth knowing.”

Alan Perlis, 1922–1990

Look at these web pages:

Jane Street Capital: [www.janestreet.com/technology/](http://www.janestreet.com/technology/)

Facebook:

[www.wired.com/2015/09/](http://www.wired.com/2015/09/)

[facebook-new-anti-spam-system-hints-future-coding/](http://www.wired.com/2015/09/facebook-new-anti-spam-system-hints-future-coding/)

Functional Programming is Black Magic

<http://www.quora.com/...> (see course web page)

# Families of programming languages

- Functional

Erlang, F#, Haskell, Hope, Javascript, Miranda, OCaml, Racket, Scala, Scheme, SML

- More powerful
- More compact programs

- Object-oriented

C++, F#, Java, Javascript, OCaml, Perl, Python, Ruby, Scala

- More widely used
- More libraries

# Functional programming in the real world

- Google MapReduce, Sawzall
- Ericsson AXE phone switch
- Perl 6
- DARCS
- XMonad
- Yahoo
- Twitter
- Facebook
- Garbage collection

# Functional programming is the new new thing

Erlang, F#, Scala attracting a lot of interest from developers

Features from functional languages eventually move into other languages

- **Garbage collection** Java, C#, Python, Perl, Ruby, Javascript
- **Higher-order functions** Java, C#, Python, Perl, Ruby, Javascript
- **Generics** Java, C#
- **List comprehensions** C#, Python, Perl 6, Javascript
- **Type classes** C++ “concepts”