University of BRISTOL

Programming and Algorithms I

Functional Programming I

Dr Nicolas Wu

nicolas.wu@bristol.ac.uk

Department of Computer Science
University of Bristol

Introduction

Functional Programming

- Functional programming is about programming with values rather than actions
- This is radically different from imperative programming!
- Although you may not end up using functional languages, understanding this paradigm will improve your understanding of computation

Statements and Expressions

- C is made up of statements and expressions

```
x = e;
s1; s2;
while ( e ) { s3; }
```

statements

```
(a + b) * c
a && !b
```

expressions

- Expressions appear on the right-hand-side of assignments and as predicates
- The ordering of statements is crucial, but is arbitrary for pure expressions

Statements and Expressions

- Making the expression language more powerful makes code simpler to understand
- Consider this program that uses an expression:

```
z = (2 * a * y + b) * (2 * a * y + c);
```

- Compared to the equivalent that uses statements but no expressions

```
ac = 2; ac *= a; ac *= y; ac += b; t = ac;
ac = 2; ac *= a; ac *= y; ac += c; ac *= t;
z = ac;
```

- Functional programming is about significantly extending expressions

Poll

- Which languages have you already used?
  - None
  - C, C++, C#, Java, PHP, JavaScript
  - Scala, Clojure, Ruby, Python
  - F#, LISP, Clojure, ML, OCaml, Haskell
  - Idris, Agda, Coq

Functional Languages?

Clojure

python™

Who Uses Haskell?

SignalVine

Microsoft

Standard Chartered

Google

NASA

Well-Typed

twitter

(,) tupil

facebook

pollarity

BARCLAYS

at&t

IAG
Insurance Australia Group

galois

SOOSTONE

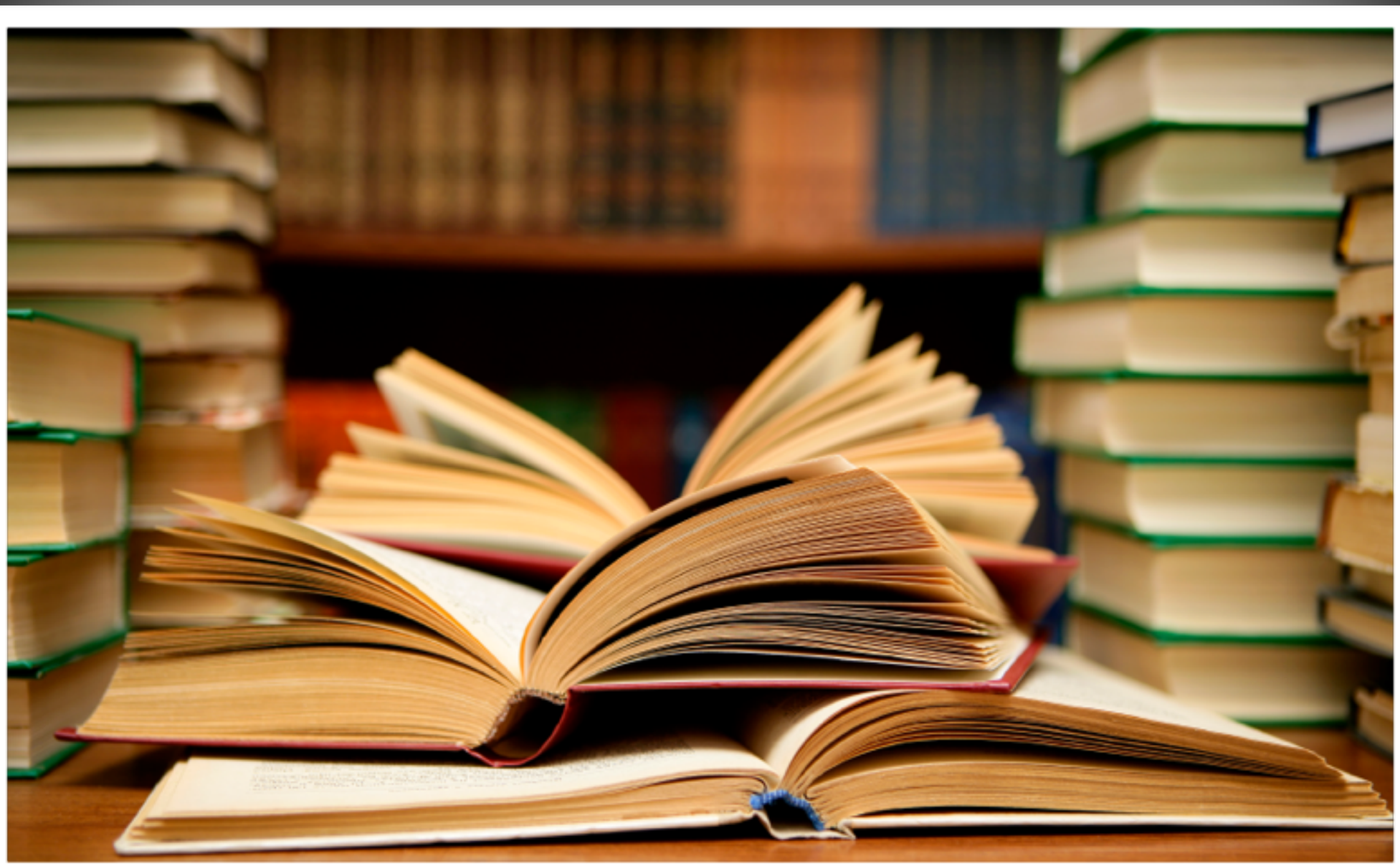
Language Laboratory

Language Laboratory

- Haskell is a petri-dish for programming language features
- Lots of features have made their way from functional into mainstream 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++)

Extra Material



Extra Material

These websites are a fantastic resource:

Real World Haskell
http://book.realworldhaskell.org/read/

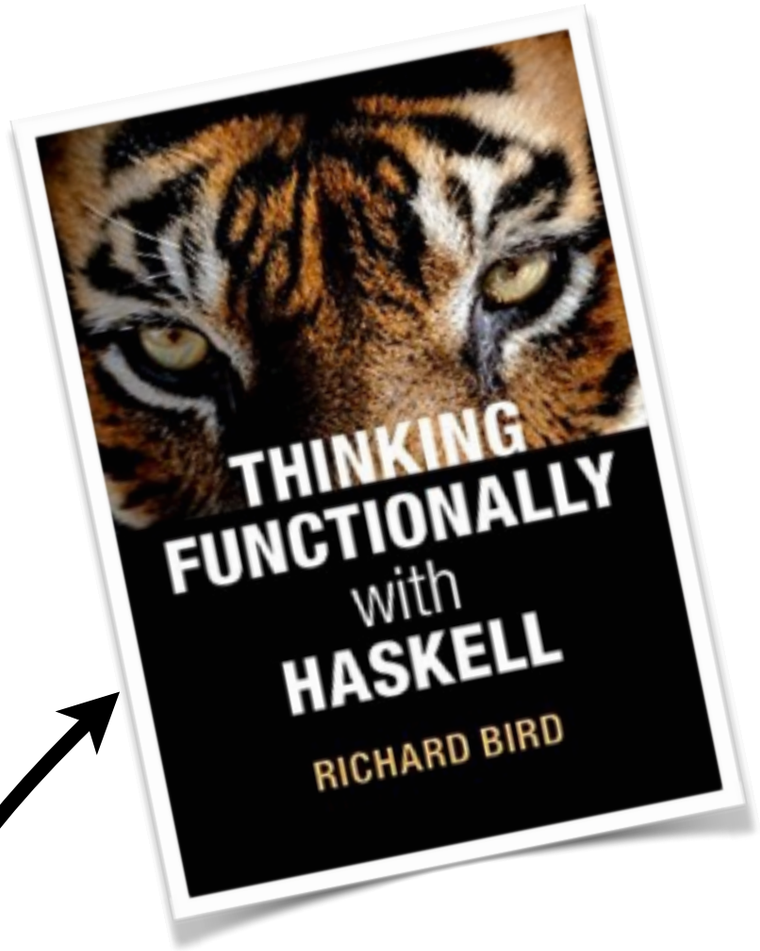
Learn You a Haskell
http://learnyouahaskell.com/

Try Haskell
http://tryhaskell.org/

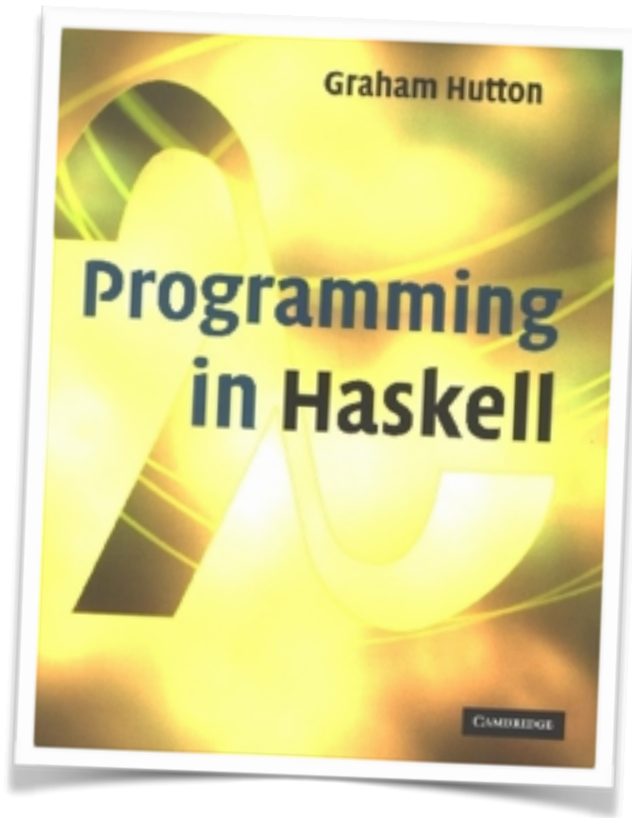
Homework: the 60 minute challenge

Extra Material

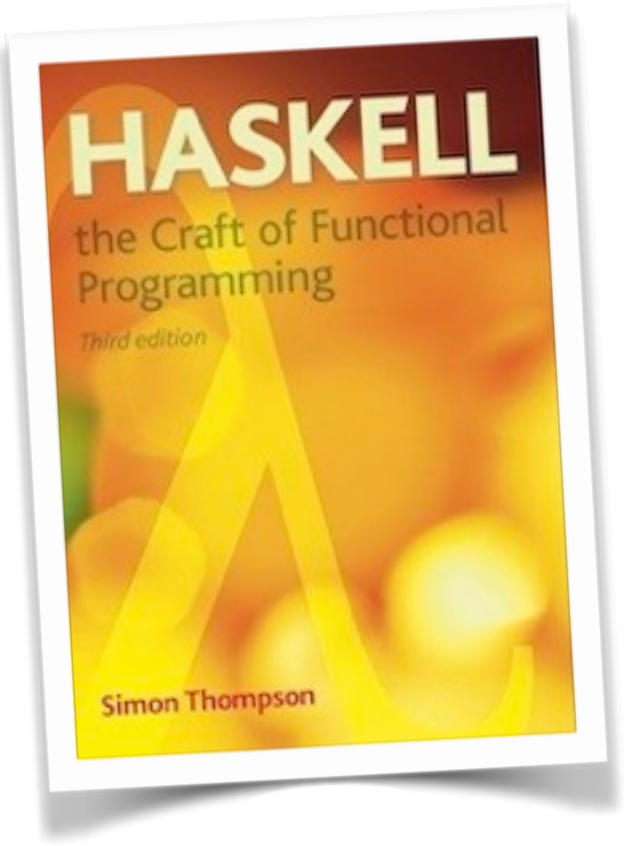
I would recommend you also read one of these:



Thinking Functionally with Haskell
by Richard Bird



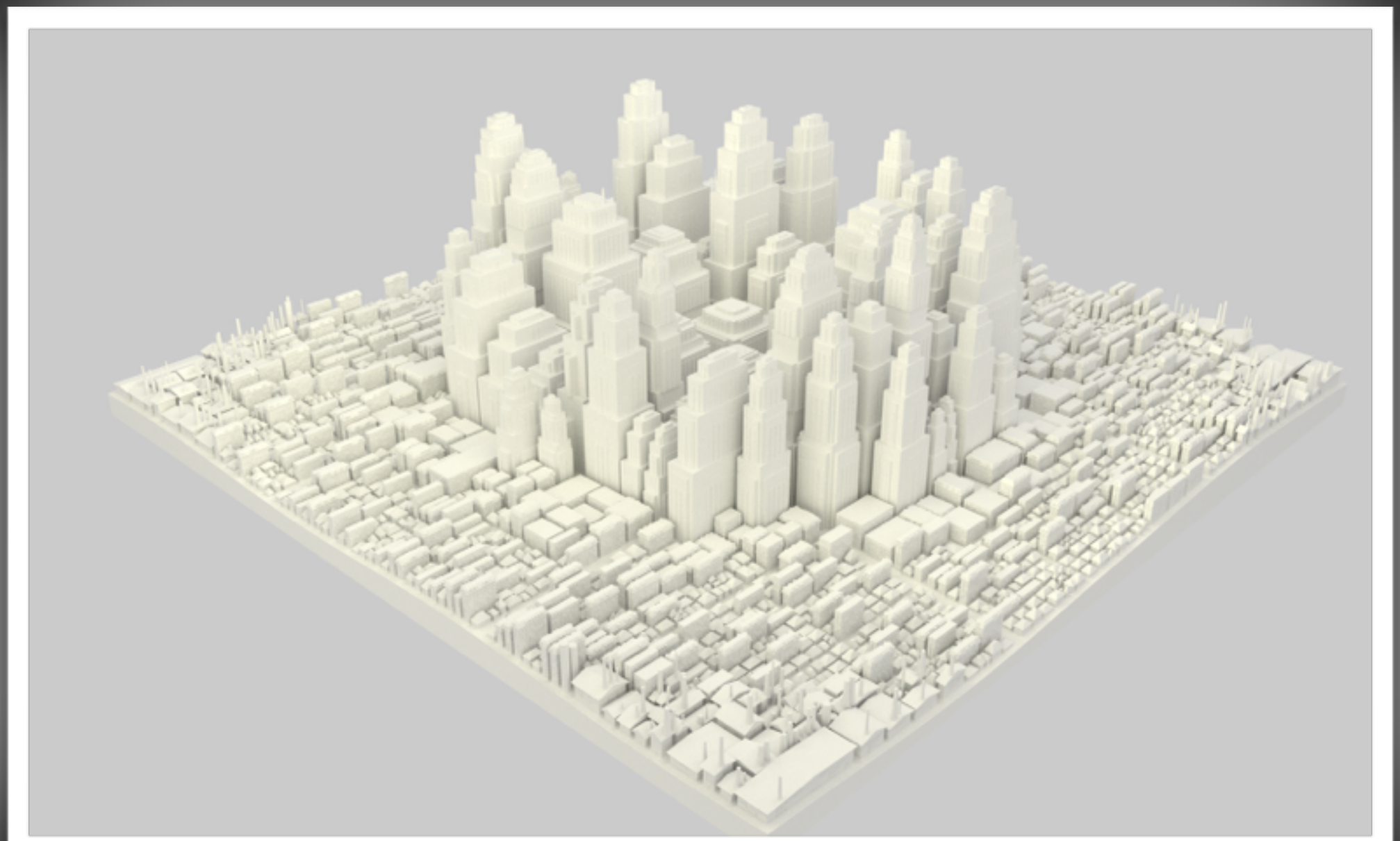
Programming in Haskell
by Graham Hutton



The Craft of Functional Programming
by Simon Thompson

if you get only one book, I recommend this

Practicals



Practicals

- Aim to be busy during your entire practical session: it's now or never!
- Ask for help when you're stuck: help can come from your fellow students, or from the lab demonstrators
- Learn to ask good questions: before asking for help, think about why you are stuck

it might help to talk to your rubber duck

A Quick Taste

Summation

C

```
int sum(int n) {
  total = 0;
  for (int i=1; i≤n; ++i) {
    total= total + i;
  }
  return total;
}

sum(10);
```

Haskell

```
foldr (+) 0 [1 .. 10]
```

Insertion Sort

C

```c
void isort(int n, int a[n]) {
    for (int i = 1; i < n; i++) {
        int j = i;
        while (j > 0 && a[j-1] > a[j]) {
            int t = a[j];
            a[j] = a[j-1];
            a[j-1] = t;
            j = j - 1;
        }
    }
}
```

Haskell

```haskell
isort :: [Int] → [Int]
isort [] = []
isort (x:xs) = insert x (isort xs)

insert :: Int → [Int] → [Int]
insert x (y:ys)
  | x ≤ y     = x : y : ys
  | otherwise = y : x : ys
```

Quick Sort

C

```c
void qsort (int n, int a[n]) {
    int i, j, p;
    if (n < 2) return;
    p = a[0];
    for (i = 0, j = n - 1;; i++, j--) {
        while (a[i] < p) i++;
        while (p < a[j]) j--;
        if (i ≥ j) break;
        int t = a[i];
        a[i] = a[j];
        a[j] = t;
    }
    qsort(i, a);
    qsort(n - i, a + i);
}
```

Haskell

```haskell
qsort :: [Int] → [Int]
qsort [] = []
qsort (x:xs) =
  qsort ys ++ [x] ++ qsort zs
  where ys = [y | y ← xs, y <  x]
        zs = [z | z ← xs, x ≤ z]
```

Programming Environment

Programming Environment

editor

Atom

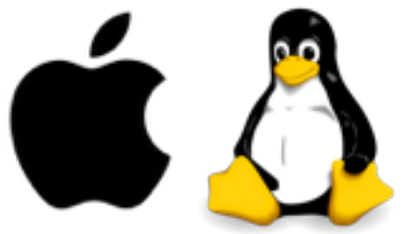
https://atom.io

compiler

GHC

https://www.haskell.org

terminal



Terminal

Command Prompt

Programming Environment

editor

Atom

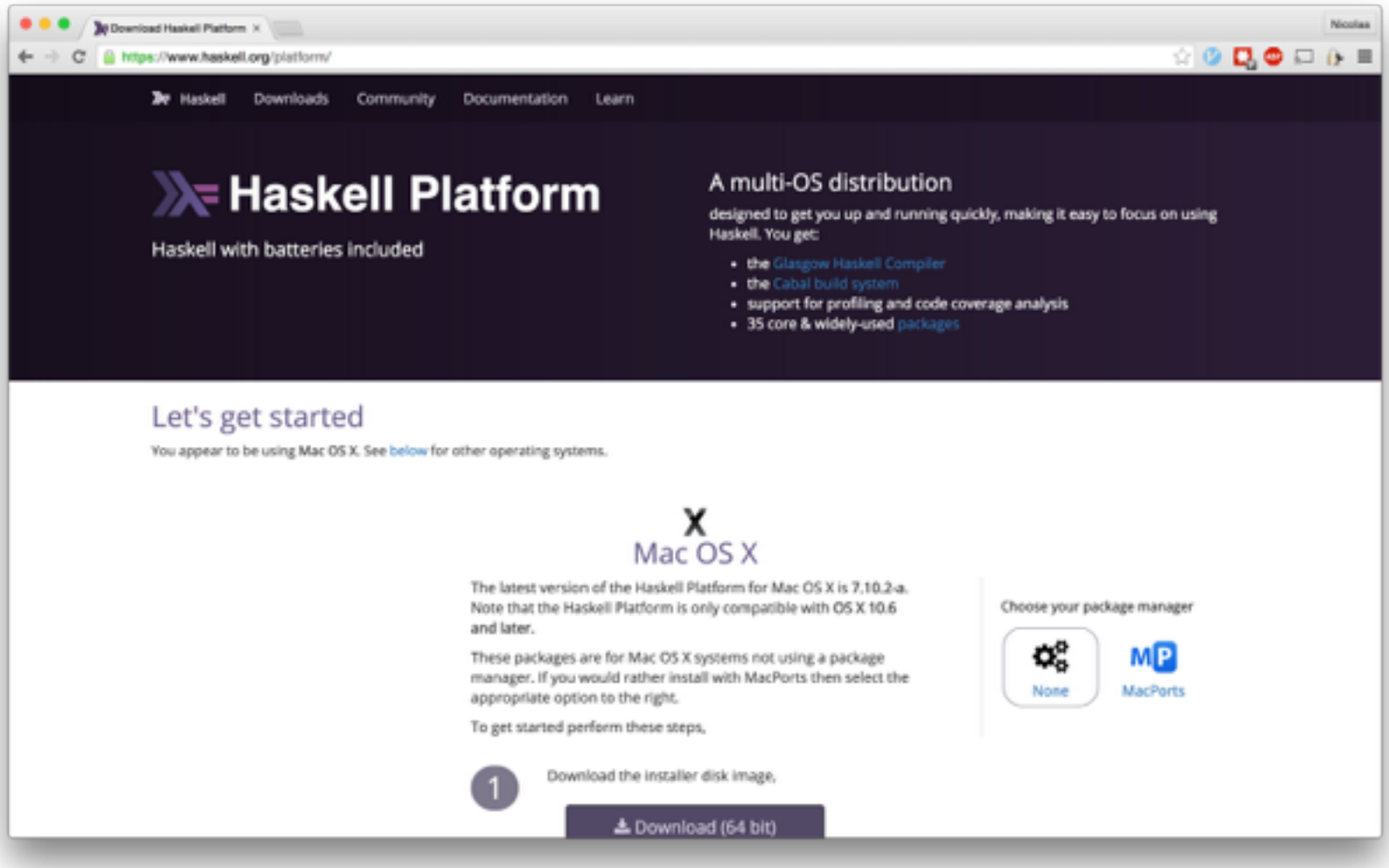
https://atom.io

Programming Environment

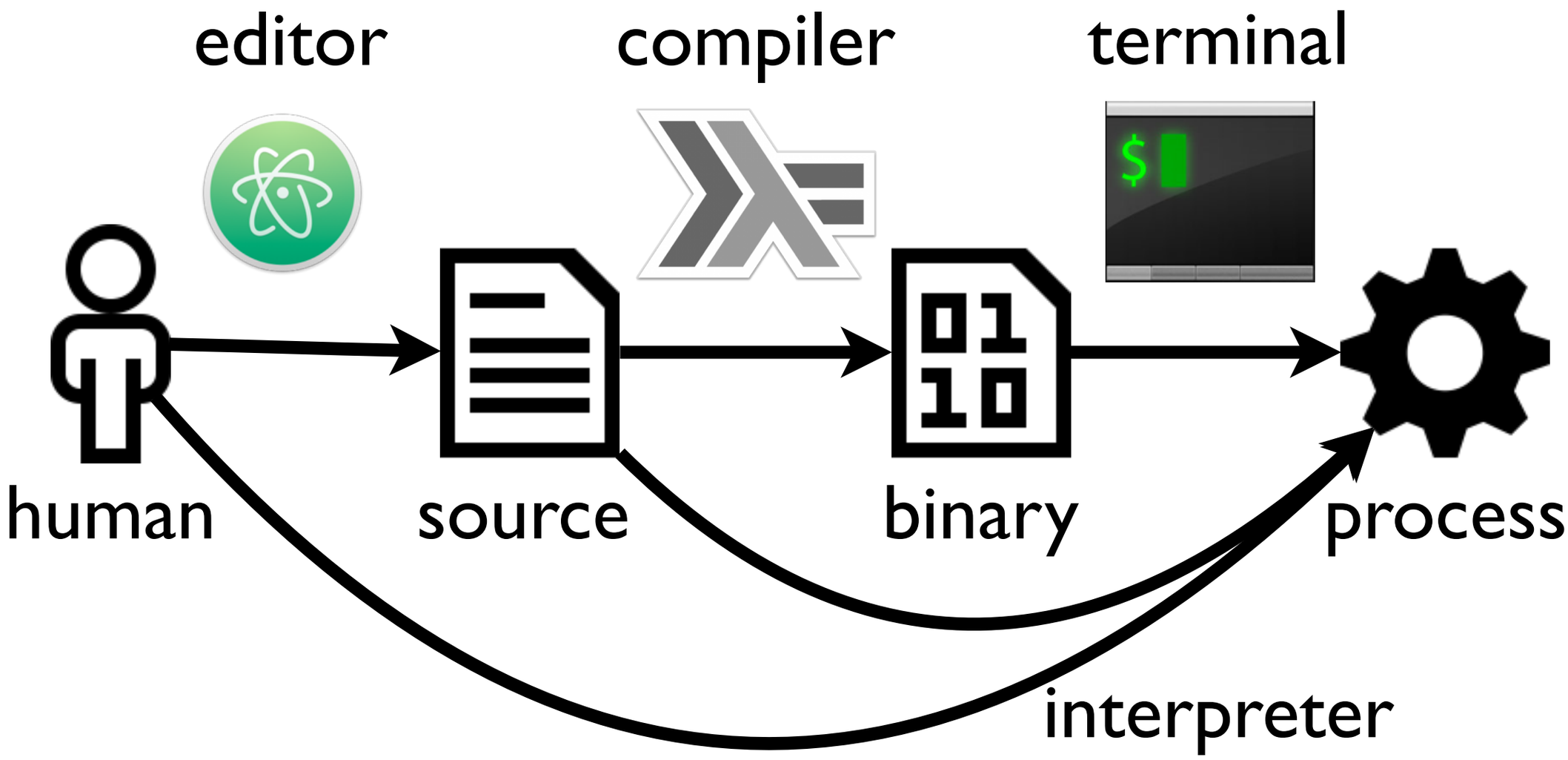
compiler

Haskell Platform

https://www.haskell.org/platform



Programming Environment



Live Coding

Live Coding

- The rest of this lecture will be about showing how to use GHC, GHCi, and playing around with some functions
- We'll cover the material presented more formally in the lectures that follow

Homework

Homework

- Before the next lecture you should:
  - Install the Haskell Platform on your laptop https://www.haskell.org/platform/
  - Understand this lecture (notes online)
  - Beat the 60 minute challenge

Challenge:
Spend 60 minutes reading and working through the examples in
Learn You a Haskell
http://learnyouahaskell.com/