

PROGRAMMING IN HASKELL

0



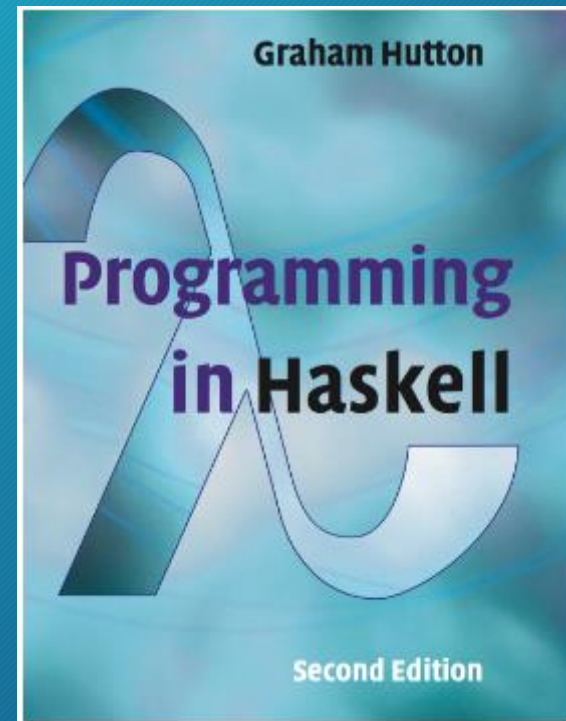
Topic 1 - Introduction

Book Title

1

This course(including slides) is largely based on the book:

Programming in Haskell,
Graham Hutton, 2nd Ed,
Cambridge University Press,
ISBN 978-1-316-62622-1.



What is a Functional Programming?

2

- A programming style using functions to compute results.
- Emphasizes immutability, pure functions, and recursion.

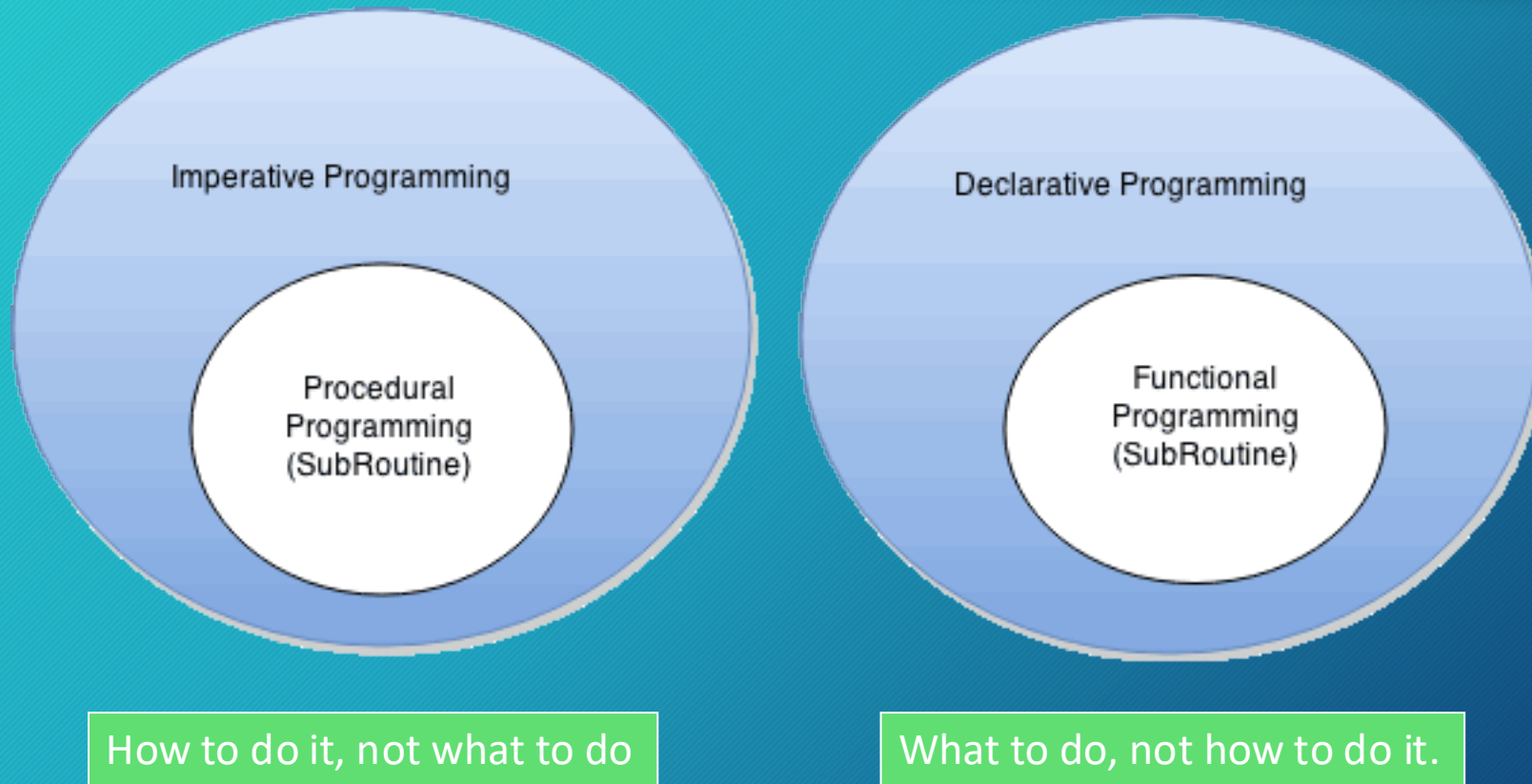
Why Functional Programming

3

- A different way to think and think about programming and solving problems.
- A great way to get good at recursion
- A lot of companies are using it..

What's different from what you've seen

4



What's different from what you've seen

5

Explain to your friend : What is an orange peanut?
(He only knows brown peanuts)

Imperative

How to do it, not what to do

That brown peanut you have : Paint it orange.
That's what an orange peanut is.

Functional

What to do, not how to do it.

That brown peanut that you have:
If you had another peanut that's just like it in every way except that it's orange.
That's what an orange peanut is.

Differences

6

Imperative

Loops

Variables - use them for e.g.
accumulating values

If [condition]
then [command]
else [command]

Functional

No loops !!!

Variables cannot be changed
(**immutability**)

If [condition]
then [value]
else [value]

More later

Comparison

7

Java

```
int total = 0;
for (int i = 1; i ≤ 10; i++)
    total = total + i;
```

The computation
method is variable
assignment.

Haskell

```
sum [1..10]
```

The computation
method is function
application.

Key Characteristics of Functional Programming

8

- - **Immutability**: Variables don't change once set.
- - **Higher-order functions**: Functions as arguments or return values.
- - **Pure functions**: No side effects.
- - **Lazy evaluation**: Values computed only when needed.
- - **Recursion**: Functions calling themselves

Historical Evolution of Functional Programming

9

- 1930's : Lambda calculus by Alonzo Church.
- 1950's: Lisp by John McCarthy.
- 1960's : ISWIM by Peter Landin.
- 1970's : FP (Backus) and ML (Milner).
- 1980's : Miranda (Turner).
- 1987 : Haskell development begins.
- 2003 : Haskell Report published.
- 2010+ : Modern Haskell advancements.

Why Haskell?

10

- **Strong type system** with type inference.
- **Lazy evaluation** for efficiency.
- **Expressive syntax** for clean code.
- Extensive library and tooling support.
- Promotes reasoning about programs.

Basic Syntax in Haskell

11

- **Strong type system** with type inference.
- **Lazy evaluation** for efficiency.
- **Expressive syntax** for clean code.
- Extensive library and tooling support.
- Promotes reasoning about programs.

Real world applications of Haskell

12

- Data analysis and processing.
- Blockchain (e.g., Cardano).
- Compilers and language tooling.
- Web development frameworks.



Any Questions ?