Informatics 1
Functional Programming Lecture 2

Functions

Don Sannella
University of Edinburgh
Part I

Functions
What is a function?

- A recipe for generating an output from inputs:
  “Multiply a number by itself”

- A set of (input, output) pairs:
  (1,1) (2,4) (3,9) (4,16) (5,25) ...

- An equation:
  \[ f(x) = x^2 \]

- A graph relating inputs to output (for numbers only):
Kinds of data

- **Integers**: 42, −69
- **Floats**: 3.14
- **Characters**: 'h'
- **Strings**: "hello"
- **Booleans**: True, False
- **Pictures**: 🖼️
Applying a function

invert :: Picture -> Picture
knight :: Picture

invert knight

invert is a function. Every value in Haskell has a type, maybe more than one. We write value :: type. A type is a category of values. Types of functions contain arrows. When we write an expression (example: invert knight) then Haskell will complain if it can't make sense of the types.
Composing functions

beside :: Picture -> Picture -> Picture
flipV :: Picture -> Picture
invert :: Picture -> Picture
knight :: Picture

beside (invert knight) (flipV knight)

beside is a function with two arguments. There is a reason for writing the type this way, to be explained later.
Defining a new function

double :: Picture -> Picture
double p = beside (invert p) (flipV p)

double knight

Functions are defined using equations. The variable name (p) is irrelevant - we could use pic or x instead. double produces the picture we had before, but packaged to work on any picture, not just knight.
Defining a new function

```haskell
define double :: Picture -> Picture
    double p = beside (invert p) (flipV p)
```

double knight

We could write beside as an infix function instead:

double p = (invert p) `beside` (flipV p)

Any function can be written as infix by enclosing it in backquotes.
Terminology

Type signature

\[
double :: \text{Picture} \rightarrow \text{Picture}
\]

Function declaration

\[
double \ p = \text{beside (invert } p) \ (\text{flipV } p)
\]
double knight

function definition

expression

double \( p \) = beside (invert \( p \)) (flip\( V \) \( p \))