Introduction to Haskell

Outline

- Functional Languages – Functions vs Orders
- Haskell – A Functional Language
- The Haskell Interpreter
- Haskell Basic Operations
- Haskell Lists
- Haskell Functions
- Haskell List Comprehensions

Functional Paradigm

- Not a series of instructions!
  - “Functional” means Functions
  - Describe the Results
  - No “Function Calling”! Function Applications!
  - No Variables (as we know them)
  - No Control Structures (as we know them)

Haskell

- A Purely Functional Language
  - Many primitive Data Types
  - High-order Functions
  - Lazy Evaluation
  - Pattern Matching
  - List Comprehension
  - Monads (But we won't be seeing them much)

The Haskell Interpreter

- http://www.haskell.org/

- Windows, Linux, MacOS
- Can execute code fragments typed into prompt
- Can compile modules from files

Haskell Basic Operations

- Basic arithmetics: +, -, *, /, ^,
- Function call: my_function x y z
  - Note the lack of parenthesis
- Explicit Type declaration: my_var :: Float
  - Not actually necessary, but you never know!

Haskell Lists

- Powerful List Engine!
  - Easy creation: [1,2,3], ['a','b','c'], [1..10]
  - Add to head: 1:[3..15]
  - Common List Functions:
    - head L
    - tail L
    - length L
    - take n L
    - reverse L
    - concat L
    - L_1 ++ L_2

Haskell Functions

- Easy to declare!

```
my_func x y = x * y
```

- Across multiple lines: use {} or tabs to delimit blocks

```
my_func x y z = x + a
    where
        a = y – z
```

- “where” keyword separates main result from other important operations.

List Comprehensions

- Creating lists procedurally
- [x ^ 2 | x <- [1..10]], what does this do?
- [x ^ 2 | x <- [1..100], isPrime x]
- [x + y | x <- [1..4], y <- [6..10]]
- Lazy evaluation means we can create infinite lists...
  - But only those things we access will be generated

Want to know more?

- The Best Haskell book you will find:

- Learn You a Haskell for Great Good!
  - http://learnyouahaskell.com/

Let's try some stuff!