Haskell Literacy in Six Slides

Greg Price (price)

2008 Jan 29

Haskell Literacy in Six Slides: Running

- log into Athena (e.g., SIPB's dialup: `ssh linux.mit.edu`)
- add ghc, and grab this class's examples
- for interactive prompt, type `ghci`
- enter expressions, or say `:t` for their type
- `:i` for info on identifiers, `:browse` to list a module

```
$ ssh linux.mit.edu
$ add ghc
$ cp -a /mit/sipb-iap/www/hackhaskell/examples haskell
$ cd haskell
$ runhaskell hello.hs  # or just ./hello.hs
$ ghci
... banner ...
Prelude> 1 + 2
3
Prelude> :t 1 + 2
1 + 2 :: (Num t) => t
```

Haskell Literacy in Six Slides: Basic Syntax

- function application: `function arg1 ... argn`
- function definition: `f x y = x + 2*y`
- optional type signatures: `f :: argtype -> argtype -> outtype`
- at `ghci` prompt, start with `let`
- `{- comments -}` or `-- one-line comments`

```
f :: Int -> Int -> Int   -- type signature
f x y = x + 2*y          -- definition, when in a file

{- at the interactive prompt:
> let f x y = x + 2*y
> f 1 2
5
-}
```

Haskell Literacy in Six Slides: Numbers

- types `Integer`, `Double`; also `Rational`, like 3%2; etc
- ops `+ - * / ^ < > <= >= ==` — note /= for ≠
- / for floats etc, `` `div` `` is integer division, `` `mod` `` mod
- `Integer` is big integers, using the state-of-the-art GMP library
- `Int` for machine integers
- `odd, even, gcd, lcm`

```
factorial :: Integer -> Integer
factorial n = if n < 2 then 1 else n * factorial (n-1)
-- I'll use leading > for the interactive prompt
> factorial 30  -- = 265252859812191058636308480000000
```

Haskell Literacy in Six Slides: More Functional Goodness

- anonymous functions: \ arg arg -> body
- (the \ was chosen to look like a λ)
- operator “slices”, with parens: (>1) $\equiv$ \x -> x > 1
- save parens with application operator $
- composition operator .

```
> (\ x y -> x^2 + y^2) 3 4
25
>   (`mod` 7) $ (^3) $ (1+) $ 10
1
> ( (`mod` 7) . (^3) . (1+) ) 10
1
```

Haskell Literacy in Six Slides: Lists

- “literals” `[1, 1, 2, 5, 14]`
- colon `x:xs` is cons, `[]` is nil
- `xs !! n` for indexing/nthcdr; `(++)` concat/append; `map` mapcar
- `length xs`, `all/any pred xs`, `elem x xs`
- `take/drop n xs`, `takeWhile/dropWhile pred xs`
- `:browse Prelude` for more
- `[1 ..]`, `[2, 4 ..]`, “comprehensions” `[x | x <- stuff]`

```
> let sums = [ x^2 + y^2 | x <- [1..15], y <- [1..x] ]
> any (\x -> x `mod` 4 == 3) sums
False
```

Haskell Literacy in Six Slides: Libraries

- at the prompt, `:m +LibraryName`
- in a file, `import qualified LibraryName`
- then `LibraryName.member`
- `List, Complex, Random, System.IO, Data.Char`

```
> let sums = [ x^2 + y^2 | x <- [1..15], y <- [1..x] ]
> :m +List
> take 20 $ List.sort sums
[2,5,8,10,13,17,18,20,25,26,29,32,34,37,40,41,45,50,50,52]
```