고급 프로그래밍

haskell

https://open.kakao.com/o/gqyIXs0c

KYUNGSUNG UNIVERSITY SINCE 1955

Haskell keywords

https://wiki.haskell.org/Keywords

Haskell 이름짓기

함수와 인자의 이름 : 소문자, 뒤이어 0개 또는 그 이상의 숫자, 문자, 밑줄, 작은 따옴표
리스트의 이름 : 관례상 s로 끝나곤 함
숫자리스트 : ns, 문자리스트 : cs, 임의의 종류 : xs, etc

Haskell 들여쓰기

지역정의

```
a = b + c
    where
      b = 1
      c = 2
d = a * 2
```

```
a = b + c
    where
     { b = 1
       c = 2 }
d = a * 2
```

Haskell 주석

라인 주석 : --
블록 주석 : {- -}

GHCI 명령어

```
:l 파일불러오기
:m 모듈불러오기
:m +ModuleName - 모듈추가하기
:m -ModuleName - 모듈제거하기
:m - 모든 모듈제거하기
import ModuleName - 모듈전체를 불러온다.
import ModuleName (component) - 모듈안에 포함된 특정 기능만 가져온다.
:q - 나가기
```

Haskell 맛보기

http://learnyouahaskell.com/starting-out

GHCI 여러 줄 입력

```
:{
let fac 0 = 1
    fac n = n * fac (n-1)
:}
```

사칙연산

```
>2 + 15
>49 * 100
>1892 - 1472
>5 / 2
```

괄호

```
>(50 * 100) - 4999
>50 * 100 - 4999
>50 * (100 - 4999)
>5 * -3
>5 * (-3)
```

논리

```
>True && False
>True && True
>False || True
>not False
>not (True && True)
```

관계연산

```
>5 == 5
>1 == 0
>5 /= 5
>5 /= 4
>"hello" == "hello"
```

```
>5 + "llama"
>5 == True
```

함수

```
>succ 8
>min 9 10
>min 3.4 3.2
>max 100 101
>succ 9 + max 5 4 + 1
>(succ 9) + (max 5 4) + 1

>succ 9 * 10
>succ (9 * 10)
```

Baby.hs

```
doubleMe x = x + x
doubleUs x y = doubleMe x + doubleMe y
doubleSmallNumber x = if x > 100
                      then x
                      else x*2
doubleSmallNumber' x = (if x > 100 then x else x*2) + 1
```

```
>:l baby
>doubleMe 9
>doubleMe 8.3
>doubleUs 4 9
>doubleUs 2.3 34.2
>doubleUs 28 88 + doubleMe 123
>doubleSmallNumber 101
>doubleSmallNumber 6
>doubleSmallNumber' 101
>doubleSmallNumber' 7
```

리스트

```
>let lostNumbers = [4,8,15,16,23,42]
>lostNumbers
>[1,2,3,4] ++ [9,10,11,12]
>"hello" ++ " " ++ "world"
>['w','o'] ++ ['o','t']
>'A':" SMALL CAT"
>5:[1,2,3,4,5]
>"Steve Buscemi" !! 6
>[9.4,33.2,96.2,11.2,23.25] !! 1
>let b = [[1,2,3,4],[5,3,3,3],[1,2,2,3,4],[1,2,3]]
>b
>b ++ [[1,1,1,1]]
>[6,6,6]:b
>b !! 2
>[3,2,1] > [2,1,0]
>[3,2,1] > [2,10,100]
>[3,4,2] > [3,4]
>[3,4,2] > [2,4]
>[3,4,2] == [3,4,2]
```

```
>head [5,4,3,2,1]
>tail [5,4,3,2,1]
>last [5,4,3,2,1]
>init [5,4,3,2,1]
>length [5,4,3,2,1]
>null [1,2,3]
>null []
>reverse [5,4,3,2,1]
>take 3 [5,4,3,2,1]
>take 1 [3,9,3]
>take 5 [1,2]
>take 0 [6,6,6]
>drop 3 [8,4,2,1,5,6]
>drop 0 [1,2,3,4]
>drop 100 [1,2,3,4]
>minimum [8,4,2,1,5,6]
>maximum [1,9,2,3,4]
>sum [5,2,1,6,3,2,5,7]
>product [6,2,1,2]
>product [1,2,5,6,7,9,2,0]
>4 `elem` [3,4,5,6]
>10 `elem` [3,4,5,6]
```

```
>[1..20]
>['a'..'z']
>['K'..'Z']
>[2,4..20]
>[3,6..20]
>[0.1, 0.3 .. 1]
>take 10 (cycle [1,2,3])
>take 12 (cycle "LOL ")
>take 10 (repeat 5)
```

```
>[x*2 | x <- [1..10]]
>[x*2 | x <- [1..10], x*2 >= 12]
>[ x | x <- [50..100], x `mod` 7 == 3]
>boomBangs xs = [ if x < 10 then "BOOM!" else "BANG!" | x <- xs, odd x]
>boomBangs [7..13]
>[ x | x <- [10..20], x /= 13, x /= 15, x /= 19]
>[ x*y | x <- [2,5,10], y <- [8,10,11]]
>[ x*y | x <- [2,5,10], y <- [8,10,11], x*y > 50]
>let nouns = ["hobo","frog","pope"]
>let adjectives = ["lazy","grouchy","scheming"]
>[adjective ++ " " ++ noun | adjective <- adjectives, noun <- nouns]
>length' xs = sum [1 | _ <- xs]
>removeNonUppercase st = [ c | c <- st, c `elem` ['A'..'Z']]
>removeNonUppercase "Hahaha! Ahahaha!"
>removeNonUppercase "IdontLIKEFROGS"
>let xxs = [[1,3,5,2,3,1,2,4,5],[1,2,3,4,5,6,7,8,9],[1,2,4,2,1,6,3,1,3,2,3,6]]
>[ [ x | x <- xs, even x ] | xs <- xxs]
```

```
>fst (8,11)
>fst ("Wow", False)
>snd (8,11)
>snd ("Wow", False)
>zip [1,2,3,4,5] [5,5,5,5,5]
>zip [1 .. 5] ["one", "two", "three", "four", "five"]
>zip [5,3,2,6,2,7,2,5,4,6,6] ["im","a","turtle"]
>zip [1..] ["apple", "orange", "cherry", "mango"]
>let triangles = [ (a,b,c) | c <- [1..10], b <- [1..10], a <- [1..10] ]
>let rightTriangles = [ (a,b,c) | c <- [1..10], b <- [1..c], a <- [1..b], a^2 + b^2 == c^2]
>let rightTriangles' = [ (a,b,c) | c <- [1..10], b <- [1..c], a <- [1..b], a^2 + b^2 == c^2, a+b+c == 24]
>rightTriangles'
```

기본 타입

Bool : False, True
Char : 한 개의 글자
String : 여러 개의 글자
Int : 정수
Integer : 정수
Float : 실수

List : 서로 같은 타입의 원소를 한 줄로 늘어놓은 것
대괄호 안에 쉼표로 구분

>head [1, 2, 3, 4, 5]

Pair : 유한 개의 서로 다른 성분을 한 줄로 나열한 것

>fst(1, 3)

함수 타입 : 어떤 타입의 인자를 도 다른 타입의 결과로 대응

>mult x = x + x

여러 모양(polymorphic) 타입 : 다양한 타입에 적용하는 타입

>length[1, 3, 5, 7]
>length[“Yes”, “No”

커리된(curried) 함수 : 함수를 함수의 결과로 돌려줌

>mult x y z

여러 의미(overloaded) 타입 : 다양한 타입에 적용하는 타입

>1 + 2
>1.1 + 2.2

Examine the types

```
>:t 'a'
>:t True
>:t "HELLO!"
>:t (True, 'a')
>:t 4 == 5
```

Type variables

```
>:t head
>:t fst
```

Functions also have types

```
>:{
 factorial :: Integer -> Integer
 factorial n = product [1..n]
:}
>factorial 50
>:{
 circumference :: Float -> Float
 circumference r = 2 * pi * r
:}
>circumference 4.0
>:{
 circumference' :: Double -> Double
 circumference' r = 2 * pi * r
:}
>circumference' 4.0
```

Typeclasses

```
>:t head
>:t fst
>:t (==)
>:t (>)
>"Abrakadabra" < "Zebra"
>"Abrakadabra" `compare` "Zebra"
>5 >= 2
>5 `compare` 3
>show 3
>show 5.334
>show True
>read "True" || False
>read "8.2" + 3.8
>read "5" - 2
>read "[1,2,3,4]" ++ [3]
>:t read
>read "5" :: Int
>read "5" :: Float
>(read "5" :: Float) * 4
>read "[1,2,3,4]" :: [Int]
>read "(3, 'a')" :: (Int, Char)
```

```
>['a'..'e']
>[LT .. GT]
>[3 .. 5]
>succ 'B'
>minBound :: Int
>maxBound :: Char
>maxBound :: Bool
>minBound :: Bool
>maxBound :: (Bool, Int, Char)
>:t 20
>20 :: Int
>20 :: Integer
>20 :: Float
>20 :: Double
>:t (*)
```

Reference

- ✓ 가장 쉬운 하스켈 책, 미란 리포바카, 황반석, 비제이퍼블릭
- ✓ 하스켈로 배우는 프로그래밍, Graham Hurton, 안기영외2, 대림
- ✓ https://tryhaskell.org/ 온라인 인터프리터
- ✓ https://www.tutorialspoint.com/haskell/index.htm
- ✓ https://downloads.haskell.org/~ghc/latest/docs/html/users_guide/ghci.html
- ✓ https://1ambda.github.io/haskell/intro-to-haskell-1/
- ✓ http://learnyouahaskell.com/types-and-typeclasses
- ✓ https://wiki.haskell.org/Programming_guidelines