classmate

Date 9.9.2019
Page 1

STARTING OUT

GHCi, version 6.8.2 : http://www.haskell.org/ghc/ :?
Loading package base ... linking ... done.
Prelude>

:set prompt "ghci>"

~~ghci>~~
λ 2 + 15
17
λ 49 * 100
4900
λ 1892 - 1472
420
λ 5 / 2
2.5
λ

λ (50 * 100) - 4999
1
λ 50 * 100 - 4999
1
λ 50 * (100 - 4999)
-244490

λ 5 * -3 ✗

λ 5 * (-3)
-15

λ True && False

False

λ True && True

True

λ False || True

True

λ not False

True

λ not (True && True)

False

λ 5 == 5

True

λ 1 == 0

False

λ 5 /= 5

False

λ 5 != 4

True

λ "hello" == "hello"

True

only matching types can be compared.

automatic type casting is allowed.

functions : infix / prefix

↳ most common

```
λ min 9 10
9
λ min 3.4 3.2
3.2
λ max 100 101
101

λ succ 8
9

λ succ 9 + max 5 4 + 1
16
λ (succ 9) + (max 5 4) + 1
16

λ succ (9 * 10)
91

λ div 92 10
9
λ 92 `div` 10
9

λ doubleMe x = x + x   } → baby.hs

λ :l baby.hs
[1 of 1] compiling Main          (baby.hs, interpreted)
Ok, modules loaded: Main.
```

λ doubleMe 9

18

λ doubleMe 8.3

16.6

doubleUs x y = x*2 + y*2 } ++ baby.hs

λ :l baby

λ doubleUs 4 9

26

λ doubleUs 2.3 34.2

73.0

λ doubleUs 28 88 + doubleMe 123

478

doubleUs x y = doubleMe x + doubleMe y

functions can be defined in any order

doubleSmallNumber x = if x > 100
then x
else x*2

mandatory

doubleSmallNumber' x = (if x > 100
then x
else x*2) +1

conanO'Brien = "It's a-me, Conan O'Brien!"

the above is basically a function. hence it cannot be changed, and can be used interchangeably.

strings are lists. and, lists are homogenous.

λ let a = 1 } in GHCi) equivalent
a = 1 } in script

λ let lostNumbers = [4, 8, 15, 16, 23, 42]
λ lostNumbers
[4, 8, 15, 16, 23, 42]

"hello" is syntactic sugar for ['h','e', 'l', 'l', 'o'].

λ [1, 2, 3, 4] ++ [9, 10, 11, 12]
[1, 2, 3, 4, 9, 10, 11, 12]
λ "hello" ++ " " ++ "world"
"hello world"
λ ['w', 'o'] ++ ['o', 't']
"woot"

append (++) to list is expensive.
but putting something at the begining of a list using ':' operator (cons) is instantaneous.

λ 'A':"SMALL CAT"
"A SMALL CAT"
λ 5:[1,2,3,4,5]
[5,1,2,3,4,5]

[1,2,3] is just syntactic sugar for 1:2:3:[].
to get element using index (!!).

λ "Steve Buscemi" !! 6
'B'
λ [9.4, 33.2, 96.2, 11.2, 23.25] !! 1
33.2

((((RESTRICT matrix size))))

λ let b = [[1,2,3,4],[5,3,3,3],[1,2,2,3,4],[1,2,3]]
λb ←
λ b ++ [[1,1,1,1]]
[..., [1,1,1,1]]
λ [6,6,6]:b
[[6,6,6], ...]
λ b !! 2
[1,2,2,3,4]

λ [3,2,1] > [2,1,0]
True
λ [3,2,1] > [2,10,100]
True
λ [3,4,2] > [3,4]
True

λ [3,4,2] > [2,4]

False

λ [3,4,2] == [3,4,2]

True

λ head [5,4,3,2,1]

5

λ tail [5,4,3,2,1]

[4,3,2,1]

λ last [5,4,3,2,1]

1

λ init [5,4,3,2,1]

[5,4,3,2]

λ head []

*** Exception: Prelude.head: empty list

λ length [5,4,3,2,1]

5

λ null [1,2,3]

False

λ null []

True

λ reverse [5, 4, 3, 2, 1]
[1, 2, 3, 4, 5]

λ take 3 [5, 4, 3, 2, 1]
[5, 4, 3]
λ take 1 [3, 9, 3]
λ take 5 [1, 2]
[1, 2]
λ take 0 [6, 6, 6]
[]

λ drop 3 [8, 4, 2, 1, 5, 6]
[1, 5, 6]
λ drop 0 [1, 2, 3, 4]
[1, 2, 3, 4]
λ drop 100 [1, 2, 3, 4]
[]

maximum
λ minimum [8, 4, 2, 1, 5, 6]
1
λ maximum [1, 9, 2, 3, 4]
9

λ sum [5, 2, 1, 6, 3, 2, 5, 7]
31
λ product [6, 2, 1, 2]
24
λ product [1, 8, 5, 6, 7, 9, 2, 0]
0

λ 4 `elem` [3, 4, 5, 6]

True

λ 10 `elem` [3, 4, 5, 6]

False

λ [1..20]

[1, 2, 3, ... 20]

λ ['a'..'z']

"abc... z"

λ ['k'..'z']

"klm... z"

λ [2, 4, ... 20]

[2, 4, 6, ... 20]

λ [3, 6, ... 20]

[3, 6, 9, ... 18]

[20..1] ✗

[20, 19, ... 1] ✓

λ [0.1, 0.3 .. 1]

[0.1, 0.3, 0.5, ... 1.09999...]

24 multiples of 13

λ [13, 26, ... 24*13]

λ take 24 [13, 26..]

λ take 10 (cycle [1,2,3])
[1,2,3,1,2,3,1,2,3,1]
λ take 12 (cycle "LOL")
"LOL LOL LOL"

λ take 10 (repeat 5)
[5,5,5,5,5,5,5,5,5,5]

λ replicate 3 10
[10,10,10]

Set comprehension:
building more specific set out of general sets.

first 10 even natural numbers
$S = \{2 * x \mid x \in \mathbb{N}, x \leq 10\}$
(predicate; output function; input set)

λ [x*2 | x ← [1..10]]
[2,4,6,8,... 20]

λ [x*2 | x <- [1..10], x*2 >= 12]
[12,14,16,... 20]

(filtering)
λ [x | x <- [50..100], x `mod` 7 == 3]
[52,59,66,73,80,87,94]

```
boomBangs xs = [ if x < 10 then "BOOM!" else "BANG!"
               | x ← xs, odd x ]
```

```
λ boomBangs [7..13]
["BOOM!", "BOOM!", "BANG!" "BANG!" ]
```

```
λ [x | x <- [10..20], x /= 13, x /= 15, x /= 19 ]
[10, 11, 12, 14, 16, 17, 18, 20]
```

an element must satisfy all predicates to be included in the resulting list. when drawing from several lists, comprehensions produce all combination of the given lists & join them.

```
λ [x*y | x<-[2, 5, 10], y<- [8, 10, 4] ]
[16, 20, 22, 40, 50, 55, 80, 100, 110]
λ [x*y | x <- [2, 5, 10], y <- [8, 10, 11], x*y > 50]
[55, 80, 100, 110]
```

```
λ let nouns = ["hobo", "frog", "pope"]
λ let adjectives = ["lazy", "grouchy", "scheming"]
λ [adjective ++ " " ++ noun | adjective <- adjectives,
                              noun <- nouns]
["lazy hobo", "lazy frog", "lazy pope",
 "grouchy hobo", "grouchy frog", "grouchy pope",
 "scheming hobo", "scheming frog", "scheming pope"]
```

```
length' xs = sum [1 | _ <- xs]
                    └ don't care
```

```
removeNonUppercase st = [c | c ← st, c `elem` ['A'..'Z']]

λ removeNonUppercase "Hahaha! Ahahaha!"
"HA"
λ removeNonUppercase "IdontLIKEFROGS"
"ILIKEFROGS"

λ 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]]
λ [[x | x <- xs, even x] | xs ← xxs]
[[2,2,4], [2,4,6,8], [2,4,2,6,2,6]]
```

tuples have fixed lengths and type, but they are not homogenous.

```
[(1,2), (8,11,5), (4,5)] X
[(1,2), (8,11), (4,5)] ✓

[(1,2), ("One",2)] X
```

```
λ fst (8,11)
8
λ fst ("Wow", False)
"Wow"

λ snd (8,11)
11
λ snd ("Wow", False)
False
```

work only on pairs

X triples, 4-tuples, 5-tuples

λ zip [1, 2, 3, 4, 5] [5, 5, 5, 5, 5]
[(1, 5), (2, 5), (3, 5), (4, 5), (5, 5)]
λ zip [1..5] ["one", "two", "three", "four", "five"]
[(1, "one"), (2, "two"), (3, "three"), (4, "four"), (5, "five")]

λ zip [5, 3, 2, 6, 2, 7, 2, 5, 4, 6, 6] ["im", "a", "turtle"]
[(5, "im"), (3, "a"), (2, "turtle")]

6
8
10
} 24

λ zip [1..] ["apple", "orange", "cherry", "mango"]
[(1, "apple"), (2, "orange"), (3, "cherry"), (4, "mango")]

λ let triangles = [(a, b, c) | c <- [1..10], b <- [1..10], a <- [1..10]]

λ let rightTriangles = [(a, b, c) |
  c <- [1..10], b <- [1..c], a <- [1..b], a^2 + b^2 == c^2]

λ let rightTriangles' = [(a, b, c) |
  c <- [1..10], b <- [1..c], a <- [1..b],
  a^2 + b^2 == c^2, a + b + c == 24]
λ rightTriangles'
[(6, 8, 10)]