COMP105 Lecture 6

List Comprehensions

List comprehensions

List ranges can produce simple arithmetic sequences

List comprehensions can produce more complex lists

```
ghci> [x*x | x <- [1..10] ]
[1,4,9,16,25,36,49,64,81,100]

ghci> [x / 10 | x <- [2,4..10] ]
[0.2,0.4,0.6,0.8,1.0]
```

List comprehensions

You can add **predicates** to a list comprehension

```
ghci> [x*x | x <- [1..10], x*x > 40]
[49,64,81,100]

ghci> [x*x | x <- [1..10], x*x > 40, x*x < 80]
[49,64]

ghci> [x*x | x <- [1..10], 2*x > 10]
[36,49,64,81,100]
```

You can have any number of predicates, and they can test anything

List comprehensions in functions

The body of a function can be a list comprehension

```
evens_less_than y = [x | x <- [0..(y-1)], x  mod  2 == 0]

ghci> evens_less_than 10
[0,2,4,6,8]
```

```
lt10 xs = [ if x < 10 then "Yes" else "No" | x <- xs]

ghci> lt10 [8..11]
["Yes","Yes","No","No"]
```

Multiple variables

You can use **more than one** sublist in a list comprehension

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

```
ghci> [ x*y | x <- [2,5,10], y <- [8,10,11], x*y > 50]
[55,80,100,110]
```

List comprehension examples

```
join xs ys = [ x ++ " " ++ y | x <- xs, y <- ys]

ghci> join ["big", "hot", "red"] ["dog", "ball", "car"]
["big dog","big ball","big car","hot dog","hot ball",
  "hot car","red dog","red ball","red car"]
```

List comprehension examples

```
removeLowercase st = [ c | c <- st, c  elem  [ A .. Z ]]

ghci> removeLowercase "The Big Dog"
"TBD"
```

List comprehension examples

```
length  xs = sum [1 | _ <- xs]
```

```
ghci> length  [2,4..100]
50
```

List comprehension examples

```
factors n = [x | x <- [1..n], n  mod  x == 0]

ghci> factors 100
[1,2,4,5,10,20,25,50,100]
```

```
primes n = [x | x <- [1..n], length (factors x) == 2]

ghci> primes 40
[2,3,5,7,11,13,17,19,23,29,31,37]
```

Lists of lists

There is no problem with **lists of lists**

- But all sublists must hold the same types

Nested list comprehensions

You can even **nest** list comprehensions

```
f xxs = [ [ x | x <- xs, even x ] | xs <- xxs]
```

```
ghci> f [[1,2,3],[4],[5,6]]
[[2],[4],[6]]
```

List comprehensions in other languages

List comprehensions arose in the functional programming world

- But they have appeared in imperative languages

For example, **Python** allows list comprehensions:

```
squares = [x**2 for x in range(10)]

[x.lower() for x in ["A","B","C"]]
```

Exercises

1. Write a function `cubesupto` that takes one parameter `x` and returns the cubes of all numbers between `1` and `x`
2. Write a function `nospaces` that takes a string and returns a copy of that string will all spaces removed
3. Write a function `allpairs` that takes two numbers `x` and `y` and returns all pairs of numbers `(a, b)` where `1 <= a <= x` and `1 <= b <= y`