COMP105 Lecture 6

List Ranges

Ranges

A list **range** lets us write a long list in a compact way

```
ghci> [1..20]
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]

ghci> ['a'..'z']
"abcdefghijklmnopqrstuvwxyz"

ghci> ['K'..'Z']
"KLMNOPQRSTUVWXYZ"
```

Ranges

A **step size** can be added like so

```
ghci> [2,4..20]
[2,4,6,8,10,12,14,16,18,20]

ghci> [3,6..20]
[3,6,9,12,15,18]

ghci> [20,19..15]
[20,19,18,17,16,15]
```

Infinite lists!?

You can use this notation to define an **infinite** list

```
ghci> [1..]
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21...
```

You can pass infinite lists to functions

```
ghci> take 5 [1..]
[1,2,3,4,5]
```

This works because Haskell is **lazy**

- We will cover this in more detail later in the course

Functions that produce infinite lists

The **repeat** function repeats its input infinitely

```
ghci> repeat 5
[5,5,5,5,5,5,5,5,5,5,...
```

The **cycle** function repeats a list infinitely

```
ghci> take 10 (cycle "abc")
"abcabcabca"
```

Exercises

1. Write a function `onetox` that takes one argument `x` and returns a list with all numbers between one and `x` inclusive
2. Write a function `evensupto` that takes one argument `x` and returns all even numbers between 0 and `x` inclusive
3. Write a function `countdown` that takes one argument `x` and returns all numbers between `x` and 0 inclusive (so, counting downwards)