Recursion

Recursion is when a function calls itself:

```
def CountDown(n):
    if n == 0:
        print('Liftoff!')
    else:
        print(n)
        CountDown(n - 1)
```

Base Cases and Problems within Problems

For recursive programs, you need to have a *base case*, where the problem is not solved by calling the function again, like the `if n == 0` bit in the function above.

For designing recursive solutions to problems, you try to find a simpler version of the problem (i.e. a version of the problem that is closer to the base case) within the problem itself. It only needs to be a tiny bit simpler.

In the example above, CountDown(10) can be broken into two steps:

1. Print '10'
2. Solve CountDown(9)

Since our base case is 0, and 9 is closer to 0, CountDown(9) is a simpler version of the problem.

This problem, though, is just as easy without recursion.

Haskell

Haskell is a programming language largely based on recursion—it has no loops, statements, or variables.