Introduction to Computer Programming

Week 4.3: Recursive functions

Bristol

Recursive functions

Recursive functions are functions that call themselves.

Example: consider the following code, which prints the integers $n$, $n-1$, ..., $1$.

In [1]:
```
def print_numbers(n):
    if n == 0:
        return
    else:
        print(n)
        print_numbers(n-1)

print_numbers(4)
```

```
4
3
2
1
```

If $n \neq 0$, then the function calls itself.

Recursion can help to break up complex calculations into simpler steps.

In general, a recursive function is made up of two parts:

1. A recursive statement, where the function calls itself using a different argument
2. A stopping condition, which determines the value of the function for a specific argument

This seems a bit abstract, so let's examine a specific example.

Example: Compute the value of $x^n$, where $n$ is an integer, using recursion

Solution: We do this by noticing that we can write $x^n = x \cdot x^{n-1}$ for any $n \geq 1$ (recursive statement). Moreover, we have that $x^0 = 1$ (stopping condition).

In [1]:
```
def power(x, n):
    if n == 0:
        return 1.0 # stopping condition
    else:
        return x * power(x, n-1) # recursive statement
```

In [5]:
```
p = power(2,3)
print(p)
```

```
8.0
```

Summary

- Recursive functions are those which call themselves
- They consist of two parts: a recursive statement and a stopping condition