Recursion is the process in which a function calls itself and repeats its behaviour until a condition is met to return a result. These functions are called recursive functions due to their operations.

All recursive functions share a common structure made up of two parts: base condition and recursive case. While the base condition is false, the function will keep placing execution contexts on top of the stack until we have a “stack overflow”. A stack overflow is when we run out of memory to hold items in the stack. When the base condition is true, the function returns a result and stops executing itself.

A good application of recursion is when we want to retrieve a factorial number. The factorial number is defined as the product of all positive integers less than or equal to its argument.

This process can be done using a recursive function, but also with an iterative function. The first function below is an iterative function that finds our factorial number, let's say 5!.

In [3]:
```
def iterative_fact(num):
    x = 1
    for i in range(1, num + 1):
        x = x * i
    return x
```

In [4]:
```
# Now let's retrieve our 5!

print(iterative_fact(5))
```

```
120
```

The function below is a recursive function that finds our 5!.

In [5]:
```
def recursive_fact(num):
    if num <= 1:
        return 1
    else:
        return num * recursive_fact(num-1)
```

In [6]:
```
# Now let's retrieve our 5!

print(recursive_fact(5))
```

```
120
```

The time complexity of these two functions is O(n).