Introduction to Programming/ Week 8: Functions and recursion / Recursion

Recursion

Recursion is a tricky concept, but perhaps a nice mathematical one. The basic idea is that a function can call itself. This can lead to a program that never ends, but if handled carefully it can be useful and elegant.

To avoid a function that never ends, your function needs some stop condition that, when reached, simply returns a value. Other than that, your function can take its input, do something which moves the input nearer to the stop condition, then call itself with the new value of the input.

Here is a basic recursion example. It is a function that, if the input is zero, prints out a message; or, if the input is not zero, it outputs the input and runs the function again decreasing the input by one. Can you see what behaviour this will produce? Run the code and see. Try changing the initial input number.

```
def countdown(count):
    if count == 0:
        print("Lift off.")
    else:
        print(count)
        countdown(count-1)

countdown(10)
```

Here is another example. Here the function takes in a string and, at each stage, it prints out the left-most character and then calls itself with the remainder of the string.

```
def left_string(in_str):
    print(in_str[0])
    if len(in_str)>1:
        left_string(in_str[1:])

left_string("make me tall and long")
```

Recursion doesn't have to mean reducing something to zero. Here is an example that increases a value. Can you see what it is doing? Have a think through the code before running it to see if you are correct.

```
def multiples(multiples_of, start_at):
    x = multiples_of * start_at
    if x<1000:
        print(x)
        multiples(multiples_of, start_at+1)

multiples(67,1)
```