Cheatsheets / **Learn Recursion with Python**

Recursion: Conceptual

Base Case of a Recursive Function

A recursive function should have a base case with a condition that stops the function from recursing indefinitely. In the example, the base case is a condition evaluating a negative or zero value to be true.

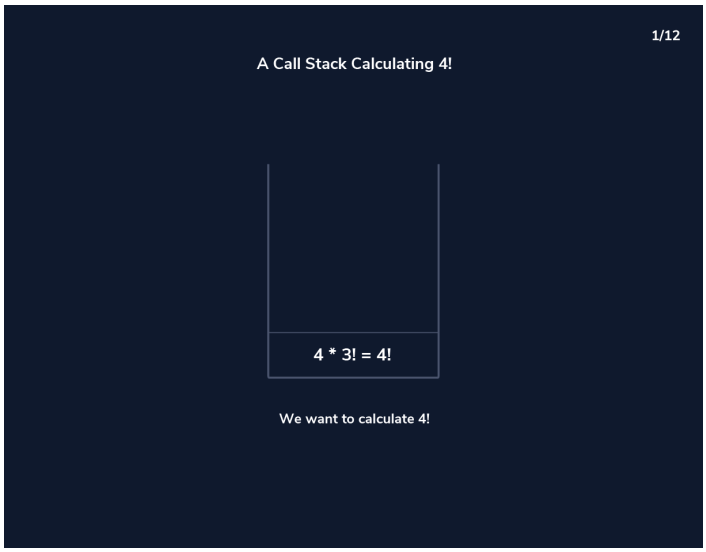
```
function countdown(value)
  if value is negative or zero
    print "done"
  otherwise if value is greater than zero
    print value
    call countdown with (value-1)
```

What is Recursion

Recursion is a strategy for solving problems by defining the problem in terms of itself. A recursive function consists of two basic parts: the base case and the recursive step.

Call Stack in Recursive Function

Programming languages use a facility called a **call stack** to manage the invocation of recursive functions. Like a stack, a call stack for a recursive function calls the last function in its stack when the **base case** is met.



Big-O Runtime for Recursive Functions

The big-O runtime for a recursive function is equivalent to the number of recursive function calls. This value varies depending on the complexity of the algorithm of the recursive function. For example, a recursive function of input N that is called N times will have a runtime of O(N). On the other hand, a recursive function of input N that calls itself twice per function may have a runtime of O(2^N).

Weak Base Case in Recursive Function

A recursive function with a weak base case will not have a condition that will stop the function from recursing, causing the function to run indefinitely. When this happens, the call stack will overflow and the program will generate a *stack overflow* error.

Execution Context of a Recursive Function

An execution context of a recursive function is the set of arguments to the recursive function call. Programming languages use execution contexts to manage recursive functions.

Print Share