Recursion

Most modern programming languages support functional recursion using the identical mechanism that is used to support traditional forms of function calls. When one invocation of the function makes a recursive call, that invocation is suspended until the recursive call completes.

class Recursion:

    def factorial(self, num: int) -> int:

        if num == 0: # base case

            return 1

        else:

            return num \* self.factorial(num - 1)

A diagram of a number

Description automatically generated

In Python, each time a function (recursive or otherwise) is called, a structure known as an ***activation record*** or ***frame*** is created to store information about the progress of that invocation of the function. When the execution of a function leads to a nested function call, the execution of the former call is suspended and its

activation record stores the place in the source code at which the flow of control should continue upon return of the nested call. This process is used both in the standard case of one function calling a different function, or in the recursive case in which a function invokes itself. The key point is that there is a different activation record for each active call.