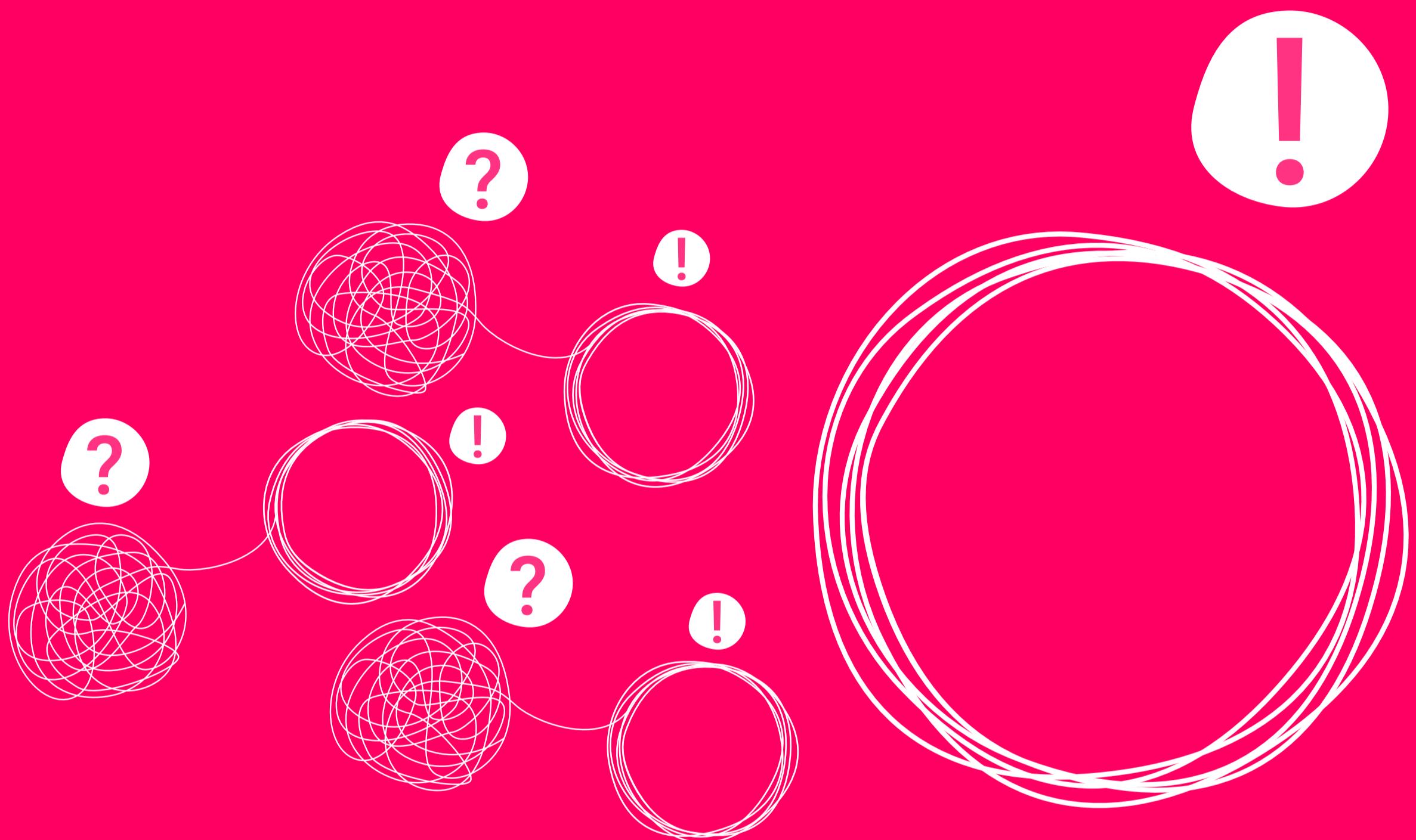




El futuro digital
es de todos

MinTIC



Algoritmos Recursivos



Universidad de Caldas

Hola:

Vamos a ver un tema que tiene un poquito más de complejidad con los temas que hemos visto anteriormente y la idea es dar una introducción a un esquema muy general de lo que es una nueva estrategia de solución de problemas que vamos a llamar **Recursión**.

```
def functionA:  
    # hacer cosas  
    functionB()  
    # llamado a funciones  
functionA()  
  
def functionB:  
    # hacer cosas  
    functionC()  
    # llamado a funciones  
  
def functionC:  
    # hacer cosas  
    functionD()  
    functionE()  
    functionF()  
    # llamado a funciones
```

Para poder entender lo que es **Recursión** es importante primero entender cómo es el manejo del llamado a funciones que hace el intérprete de Python. Supongamos que en el intérprete hay un llamado a la función A y cuando eso sucede él se va a la definición de la función y empieza a hacer las instrucciones que hay allí. Si en esa función A hay una instrucción que hace un llamado a la función B, lo que hace el intérprete es ir a esa función B y empezar a ejecutar las instrucciones que están allí, y así sucesivamente hasta que ya no llame más funciones.

A continuación, tenemos el ejemplo de una función recursiva. Factorial, es un número entero que es la multiplicación de números desde 1 hasta ese número entero. Entonces si nos dicen ¿Cuál es el número factorial de cuatro (4)? Sería $1 \times 2 \times 3 \times 4$.

$$n! = \begin{cases} 1 & \text{si } n = 0 \text{ or } n = 1 \\ n * (n - 1)! & \text{si } n > 1 \end{cases}$$

¿Cómo sería una definición recursiva de Factorial? Decimos que N Factorial es igual y hay dos opciones: es igual a 1 si el número al que le vamos a calcular el Factorial es 0 o 1, es decir, el Factorial de 0 es 1 y el Factorial de 1 es 1. Aquí no hay una recursión porque nos está dando una respuesta directa, sin embargo, abajo nos dicen que el Factorial de N es $N \times (N-1)!$, es decir, estamos dando la solución de Factorial en términos de sí mismo, esto cuando N es $>$ a 1, pero aquí hay un detalle, Factorial en términos de Factorial pero es más pequeño porque N Factorial es más grande que N-1, entonces está cumpliendo eso. Estos ya tienen unos nombres definidos así:

$$n! = \begin{cases} 1 & \text{si } n = 0 \text{ or } n = 1 \\ n * (n - 1)! & \text{si } n > 1 \end{cases}$$

Condición de salida

Llamado recursivo

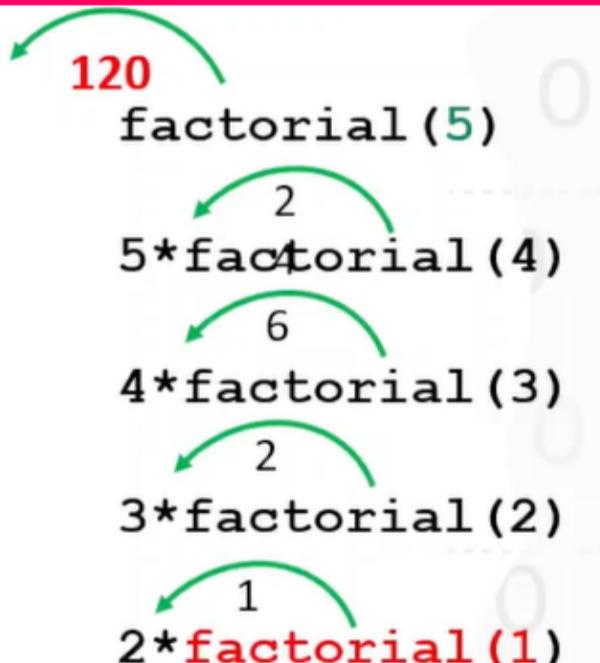
Una función Recursiva en computación debe tener ambos elementos o varias condiciones de salida y un llamado o varios llamados recursivos.

Si hacemos un seguimiento a esta función estamos pasando tal cual la condición a Python como se muestra a continuación.

```
def factorial(n):  
    if n==0 or n==1:  
        return 1  
    else:  
        return n*factorial(n-1)
```

```
factorial(5)
```

Si tenemos la definición **Recursiva** es muy fácil hacer la implementación como lo vemos en el ejemplo de Factorial de cinco (5)



En conclusión, vimos el término de **Recursión**, su definición y que cuando se define se debe esclarecer la condición de salida y el llamado a recursión.



**Mision
TIC2022**

The logo features the text "Mision TIC2022" in a bold, sans-serif font. The word "Mision" is positioned above "TIC2022". A thin, curved gray line starts from the top of the letter "i" in "Mision" and sweeps down to the bottom of the letter "c" in "TIC2022". The entire logo is set against a circular background composed of a grid of small, semi-transparent gray dots.



**Mision
TIC 2022**

The logo features the text "Mision TIC 2022" in a stylized font. The word "Mision" is in blue, "TIC" is in red, and "2022" is in blue. A red curved line starts from the top of the letter "i" in "Mision" and ends at the top of the letter "i" in "2022". The background of the logo is a white circle with a gray halftone pattern, set against a dark red circular frame.

Universidad de Caldas