26/Noviembre/2018

***Matemáticas Aplicadas Y Ciencias de la Computación / MACC***

***Entrega Final Proyecto de Programación 2018-II***

**Presentado por:** **Presentado a:**

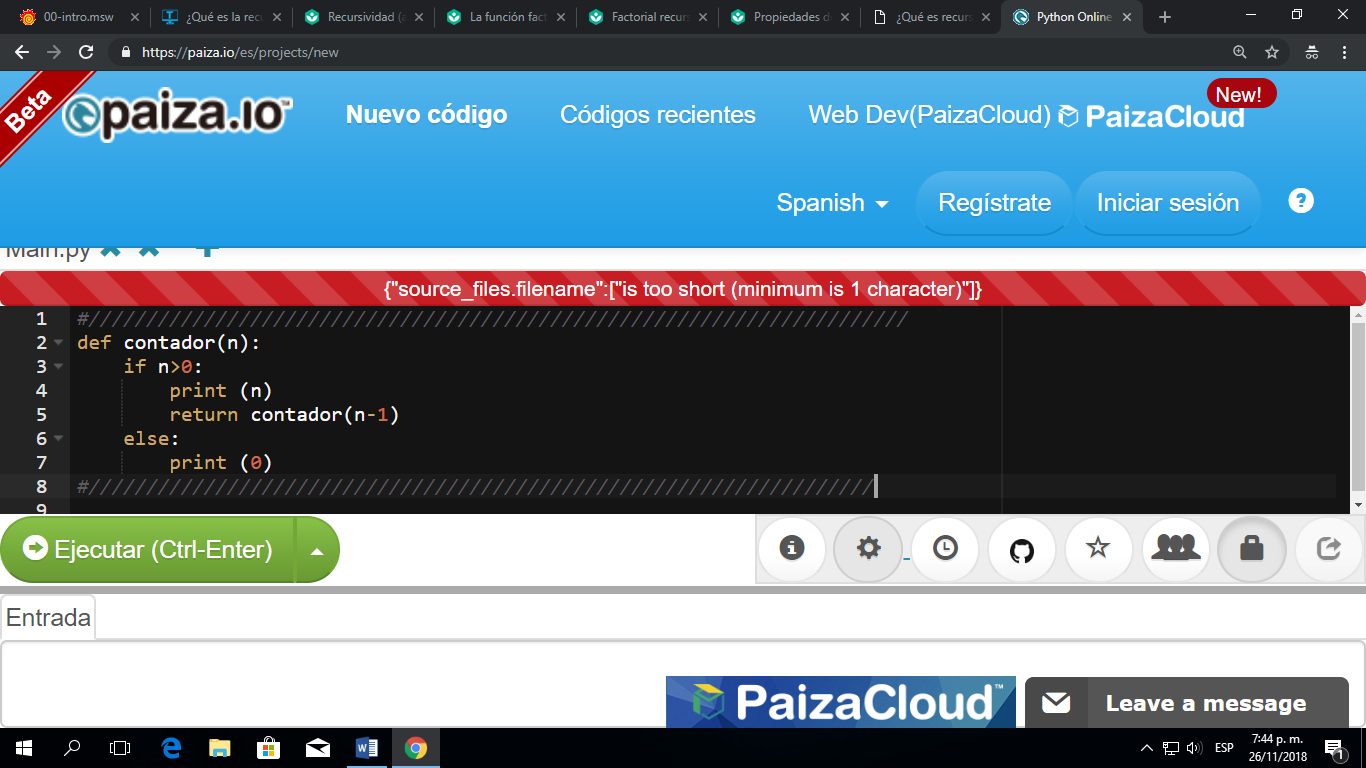
Santiago Uribe Luna José Julián Jiménez Rincón

Juan Camilo Rodríguez Jiménez

Uno de los conceptos más poderosos de la programación es la “**Recursión**”. Es un concepto simple pero a la vez confuso al mismo tiempo, porque esto implica abrir tú mente a una nueva forma de pensar. La recursión, también conocida como recursividad o recurrencia, es un concepto abstracto que hace referencia a que una función o método se invoca a sí misma, este se basa en la solución de problemas que dependen de pequeños resultados de instancias del mismo problema. Otra forma más sencilla de apreciar esta definición es asimilar que el método se utiliza para realizar una llamada a una función dentro del bloque de código que la define.

Un ejemplo simple con el que se puede relacionar esta técnica es observando un juego de muñecas rusas. Al principio, solo se puede ver una muñeca, pero puedes quitar la parte de arriba de esta y lo que encuentras adentro es otra muñeca rusa, solo que ahora un poco más pequeña. Al igual que la anterior puedes separar la muñeca y en su interior podrás ver, una vez más, otra muñeca más pequeña. Y así puedes continuar hasta que eventualmente encontraras la muñeca rusa más pequeña la cual es de solo una pieza y la cual no se va poder abrir.

Esta es la idea básica detrás de la recursión, es decir, para resolver un problema se resuelve un subproblema que sea de instancia más pequeña del mismo problema y después usa la solución de esa instancia más pequeña para resolver el problema original. Para que quede más claro a continuación observaremos un breve ejemplo de un código donde se usa recursión:



En el código anterior, creamos una función que según una cantidad ***n*** de números ingresada, los imprime de una forma regresiva sin necesidad del uso de ciclos. Lo que está ocurriendo al invocar la función es que dado un numero ***n***, la función lo comprobara si este es mayor a 0, si este resulta ser igual o menor a 0, el programa imprimirá 0 y terminara la ejecución. Pero si la ***n*** resulta ser mayor a 0. El programa va a imprimir el número ***n*** y luego se volverá a inicializar la función con el argumento ***n-1***, y así seguirá imprimiendo todos los números consecutivos al ingresado hasta llegar al 0.

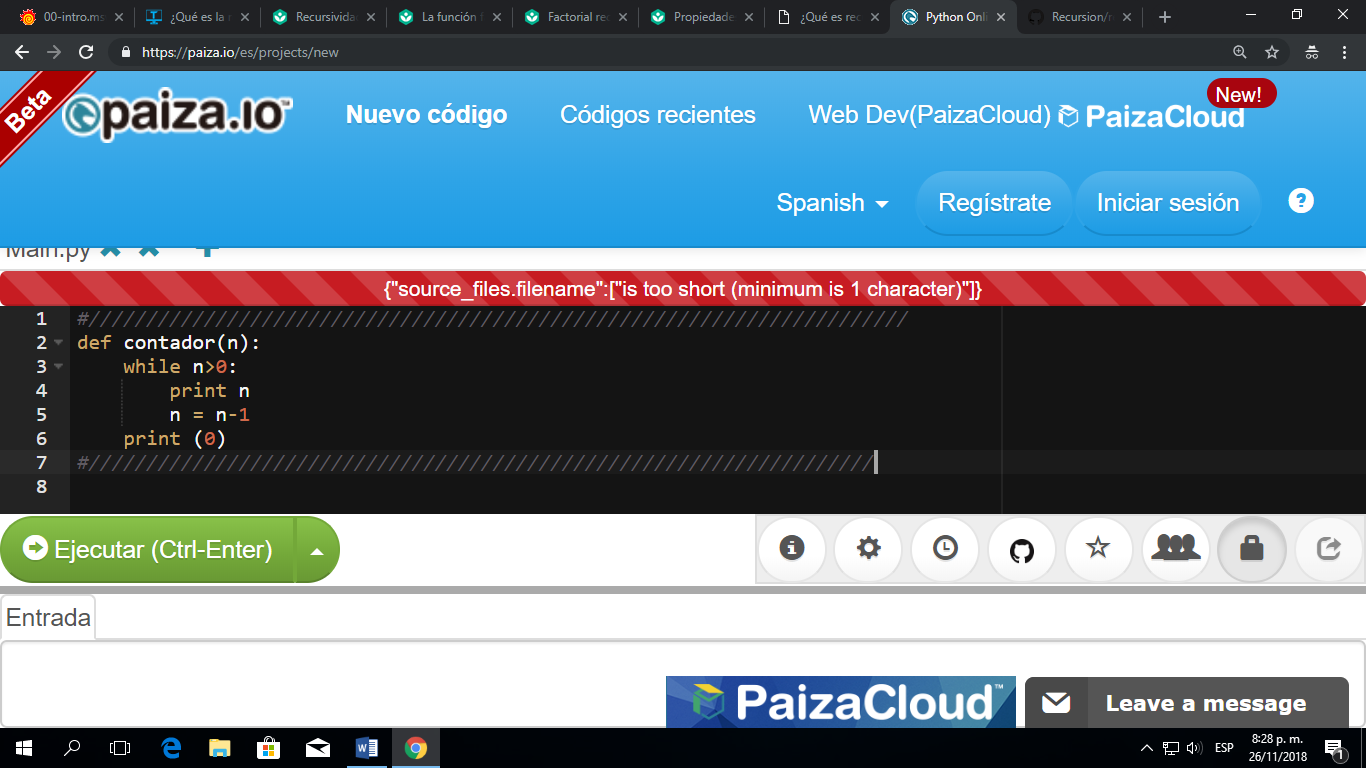
Todos los algoritmos recursivos deben obedecer tres leyes importantes:

1. Un algoritmo recursivo tiene que tener un caso base.
2. Un algoritmo recursivo debe cambiar su estado y moverse hacia su caso base.
3. Un algoritmo recursivo debe llamarse a sí mismo.

En primer lugar, un caso base es la condición que permite que el algoritmo detenga la recursividad. Un caso base es típicamente un problema que es lo suficientemente pequeño como para resolverlo directamente. Para obedecer la segunda ley debemos organizar un cambio de estado que mueva el algoritmo hacia el caso base. Un cambio de estado significa que se modifican algunos datos que el algoritmo está usando. Por lo general, los datos que representan nuestro problema se hacen más pequeños.

La última ley es que el algoritmo debe llamarse a sí mismo, esta es la definición misma de la recursividad. La recursividad es confusa para programar en especial si se lleva tan poco tiempo con un tema como estos y es más difícil aun el pensar que muchas veces la definen prácticamente con su mismo nombre

Esto puede compararse con el método de la iteración, o también conocido como los ciclos “***loops***” o ciclos repetitivos los cuales son usados para realizar tareas repetitivas, mientras que la recursión, como ya se mencionó antes, consta de la llamada repetitiva de las funciones. Y aunque matemáticamente estos dos tipos de solución de problemas son equivalentes, existen muchas diferencias entre las dos, entre las que más sobresalen se encuentran la legibilidad y el rendimiento. Veamos el mismo ejemplo anterior pero con un ciclo:



Este código a pesar de que realiza la misma tarea que el anterior, se puede evidenciar claramente que su estructura y su forma de actuar en la maquina son completamente diferentes pues cada uno tiene su propia forma de desarrollar el ejercicio. En este caso por ejemplo el programa va a tomar el número ***n*** y va a definir si este es mayor a ***1***, si este es el caso el programa imprimirá el numero ***n*** y luego le va a restar ***1***  y va a volver a comprobar si el numero ahora es mayor a ***1***. Si este es el caso volverá a restarle una unidad hasta que este sea menor o igual a 0. Aunque el número de líneas es muy similar, podemos argumentar que la solución recursiva es mucho más clara y fácil de comprender a simple vista. Y como se pude evidenciar, ambos métodos llegan al mismo resultado siguiendo una filosofía muy similar, pero con estructuras completamente diferentes.

Una vez mencionado esto es común comparar estos dos métodos para identificar cual es mejor. Sin embargo, ambas técnicas no son exclusivas entre si y cada una tiene su propio momento para ser implementada.

La recursión es un método muy elegante el cual mejora la legibilidad del código, ya que existen problemas que, planteados de forma recursiva, son muchísimo más sencillos de entender y de resolver que si se plantearan de forma de ciclos. En la mayoría de los casos el código se reduce notoriamente además de que permite la comprensión de la función sea más rápida y evidente.

Mientras que por otro lado los ciclos repetitivos son mucho mejor eficientes al momento de calcular grandes cálculos ya que esta técnica administra de forma mucho más eficiente el procesador de la computadora, ya que la complejidad de los cálculos que tiene que realizar la recursión es mucho mayor mientras que se tenga que aumentar la cantidad de operaciones realizadas.

En conclusión, la recursividad es un método que, de ser bien empleado, mejorará la calidad de nuestro código y permitirá que este sea más comprensible y elegante, pero este también debe usarse con cuidado y utilizándose en los momentos adecuados y necesarios pues los ciclos repetitivos también son una gran herramienta que nos puede facilitar en otros casos la solución del problema.

Dirección del repositorio:

<https://github.com/JuankRodriguez/Recursion-Tutorial>

# Referencias

Guerra, N. R. (15 de Junio de 2017). *Technodyan*. Obtenido de Technodyan: https://www.technodyan.com/recursividad-aplicar-recursion-programas/

Khan Academy. (13 de Abril de 2016). *Khan Academy*. Obtenido de Khan Avademy: https://es.khanacademy.org/computing/computer-science/algorithms/recursive-algorithms/a/recursion

Khan Academy. (13 de Abril de 2016). *Khan Academy*. Obtenido de Khan Academy: https://es.khanacademy.org/computing/computer-science/algorithms/recursive-algorithms/a/properties-of-recursive-algorithms

Roberts, E. S. (2014). *Programming Abstractions in C++.* Agosto : 7.

Wikipedia. (6 de Octubre de 2018). *Wikipedia La enciclopedia libre*. Obtenido de Wikipedia La enciclopedia libre: https://es.wikipedia.org/wiki/Recursi%C3%B3n\_(ciencias\_de\_computaci%C3%B3n)