UTN AVELLANEDA
Tecnicatura Universitaria en Programación

Recursividad

Programación I

Versión 24

UTN AVELLANEDA
Tecnicatura Universitaria en Programación

Introducción

Introducción

Matrushka

La Matrushka es una artesanía tradicional rusa. Es una muñeca de madera que contiene otra muñeca más pequeña dentro de sí. Ésta muñeca, también contiene otra muñeca dentro. Y así, una dentro de otra.

Introducción

Dado el ejemplo de la **Matrushka** en la programación se lo asemeja con el concepto de Recursividad

Recursividad

Se denomina **Recursividad** al proceso de definir algo en **términos de si mismo**.

Son usados en la matemática y en la programación y se asemejan a la estructura de una **Matrushka**

En programación es la forma en la cual se especifica a un **proceso (función)** basado en su **propia** definición

Funciones Recursivas

- En **Python** una función recursiva hace referencia a una función que se llama a sí misma (**las veces que sea necesario**) consiguiendo el objetivo de depender solo de ella.
- También es una alternativa diferente para implementar estructuras de repetición (**bucles**)
- Se implementan totalmente en el **paradigma funcional** ya que en el mismo no existen estructuras de repetición

Ejemplo Factorial

Si recordamos la definición de factorial

*n!= n * (n-1)!*

si n>=0

Si queremos calcular el factorial de 5 tendremos

5! = 5 . 4 . 3 . 2 . 1 = 5 . 4! = 120

4! = 4 . 3 . 2 . 1 = 4 . 3! = 24

3! = 3 . 2 . 1 = 3 . 2! = 6

2! = 2 . 1 = 2 . 1! = 2

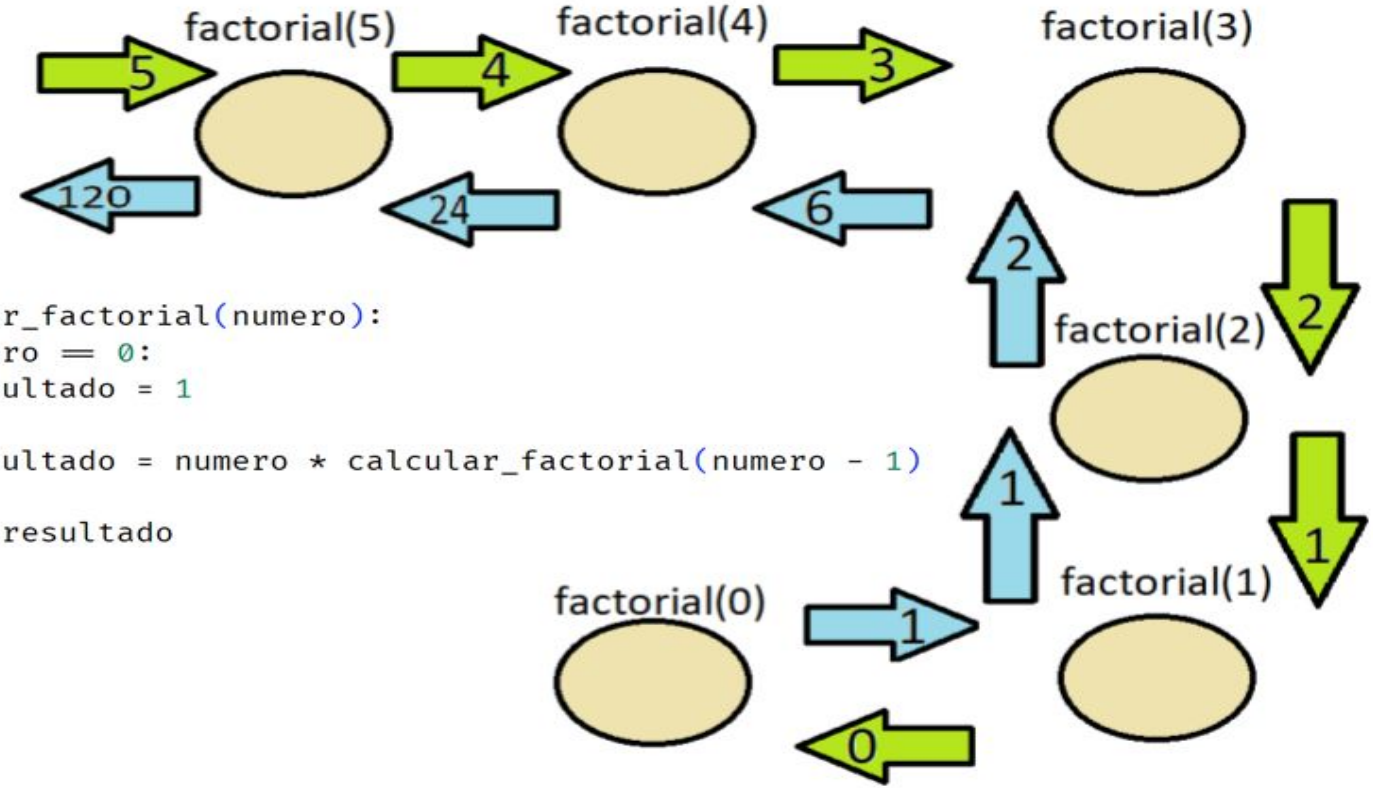
1! = 1

0! = 1

Ejemplo Factorial

```python
def calcular_factorial(numero):
    if numero == 0:
        resultado = 1
    else:
        resultado = numero * calcular_factorial(numero - 1)

    return resultado
```



Otro ejemplo

```python
def imprimirNumerosHastaElCero(numero:int):
    '''
        Función que se encarga de imprimir desde el número ingresado hasta el cero de manera descendente.
        Recibe el número inicial (Debe ser 0 o positivo)
        No retorna nada, solo imprime la información.
    '''
    if numero != -1:
        print(f"El número es {numero}")
        numero -=1
        imprimirNumerosHastaElCero(numero)


imprimirNumerosHastaElCero(5)
```

input

```
El número es 5
El número es 4
El número es 3
El número es 2
El número es 1
El número es 0
```

Otro ejemplo

La siguiente función me imprime todos los números desde el número que le pase al **parámetro actual** hasta el cero. Está función preguntara si mientras el **parametro formal** (**numero**) sea distinto de -1, me va a mostrar, decrementar y repetir el proceso llamando a la **misma función** que se está ejecutando y ahí está la **Recursividad**.

Ventajas

- Sintaxis más simple y compacta
- Son más cercanos a la descripción matemática.
- Se adaptan mejor a las estructuras de datos recursivas
- Resolver problemas cuando no exista una solución iterativa simple.

Desventajas

- Creación de múltiples variables en memoria.
- Consumen demasiada memoria en la pila (stack)
- Si son mal programadas pueden desbordar la memoria (StackOverFlow)
- Su mal uso equivale a un bucle infinito

Tipos de Recursividad

Existen cuatro tipos de recursividad en programación

- **Recursividad Simple**
- **Recursividad Múltiple**
- **Recursividad Anidada**
- **Recursividad cruzada o indirecta**

Tipos de Recursividad

- **Recursividad simple**: Aquella en cuya definición sólo aparece una llamada recursiva. Se puede cambiar a una función con ciclo iterativo.

- **Recursividad múltiple**: Se da cuando hay más de una llamada a sí misma dentro del cuerpo de la función

Tipos de Recursividad

- **Recursividad anidada:** En algunos de los argumentos de la llamada recursiva hay una nueva llamada a sí misma.

- **Recursividad cruzada o indirecta:** Son algoritmos donde una función provoca una llamada a sí misma de forma indirecta, a través de otras funciones.

Resumen

- Los algoritmos recursivos son funciones que se llaman a si mismas al menos una vez.
- Su implementación no es recomendada cuando hay una solución iterativa sencilla
- Si no se usan con cautela pueden generar un bucle infinito
- Los algoritmos recursivos son la base del paradigma funcional