

Funciones en Python

Funciones incorporadas, recursivas

Por Sebastián Herrera y Julio Zaldaña



Universidad de San Carlos de Guatemala
Student Branch Chapter

**IEEE
COMPUTER
SOCIETY**



Clase 6

1. Funciones predeterminadas
2. ¿Cómo hacer una función en Python?
3. Ejemplos de funciones.
4. Funciones Recursivas.
5. Ejemplo de una función recursiva.



Funciones en Python



¿Qué son las funciones en Python?

Las funciones en Python constituyen unidades lógicas de un programa y tienen un doble objetivo:

- Dividir y organizar el código en partes más sencillas.
- Encapsular el código que se repite a lo largo de un programa para ser reutilizado.

Forma de declarar una función en Python

Python tienen las siguientes características:

1. `def`: palabra clave.
2. Nombre de la función.
3. `()`: paréntesis que incluyen los parámetros de entrada (opcionales).
4. `:` : dos puntos.
5. Bloque de código.
6. Sentencia de retorno (opcional).

Ejercicios de Funciones en Python

Función para
decir "Hola"

Función para sumar
2 números

Función de potencias

Funciones Recursivas

¿Qué es Recursividad?

Recursividad

La recursividad es un concepto fundamental en las matemáticas y en programación.

- Es una alternativa para la implementación de ciclos.
- La recursividad es un concepto que se indica cuando una función se llama a sí misma.



¿Qué es una función recursiva?

- El procedimiento se llama a sí mismo.



- Resuelve el procedimiento de forma simplificada.



- Se llegará siempre a una solución simple, y esperada.

¿Cómo es una función recursiva?

Caso Base

Es una condición

Se puede acceder, usando else/if statements.

Caso Recursivo

Aquí se utiliza la función original

Utiliza parámetros que hacen que se acerque a una solución al caso base.

Ejemplo: Suma de un número

Se debe de escribir un programa que calcule la suma desde uno al numero dado.

Ejemplo

$$5 = 5+4+3+2+1=15$$

$$4 = 4+3+2+1=10$$

Identificar patrón:

$$\text{numero} = \text{numero} + (\text{numero}-1)$$

Se tiene que trabajar la recursividad para llegar a una condición y que logré parar.

- ¿Qué condición puedo generar?
- Analizar lo que debo de realizar; puede ser:

$$1 = 1+0 = 1$$



- Cuando llegue al 1, se parará la recursividad

Ejemplo: Uso del factorial

Se debe de escribir un programa que calcule el factorial (!) de un entero no negativo.

- Recordémonos de cómo funciona el factorial

$$0! = 1$$

$$1! = 1$$

$$2! = 1 \cdot 2 = 2$$

$$3! = 1 \cdot 2 \cdot 3 = 6$$

$$4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24$$

$$5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$$

$$6! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 = 720$$

Identificar patrón del factorial:

$$n! = n \times (n - 1) \times (n - 2) \times \dots \times 1$$

$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

$$N! = \begin{cases} 1 & \text{si } N=0 \text{ (base)} \\ 0 & \text{si } N=1 \\ N \times (N-1)! & \text{si } N > 0 \text{ (recursión)} \end{cases}$$



Se puede plantear una función por partes para facilitar y trabajar la función.

Ejemplo: Recorrer una lista

se pueden recorrer listas de forma recursiva

- Recordémonos de cómo funciona una lista en python.

```
lista = ['Toyota', 'Mazda', 'BMW', 'Nissan', 'Hyundai']
```

- Identificar parámetros para la función que recorrerá la lista

1. lista
2. índice

- Recordar funciones útiles de Python para manejo de listas:

- len(): devuelve la cantidad de elementos en una lsita

Ventajas y Desventajas

VENTAJAS:

1. Se simplifica el código.
2. Las estructuras de datos pueden ser recursivas.

DESVENTAJAS:

1. Cuando no se sigue una tendencia como tal, patrón o relación.
2. Pueden ser complejas.
3. Realizar iteraciones o utilizar ciclos puede ser una mejor opción.



¡Muchas
Gracias!

Próxima Clase: Introducción a POO en Python -> 13 de octubre