DIGITALHOUSE

Funciones en Python

01

¿Qué son las Funciones en Python?

Funciones

Una función en Python es un bloque de código reutilizable que realiza una tarea específica. Las funciones permiten dividir un programa en partes más pequeñas y manejables, lo que facilita la lectura, la depuración y la reutilización del código.

```python
# La sintaxis básica para definir una función en Python es la siguiente:

def nombre_de_la_funcion(parametros):
    """Docstring de la función"""
    # Cuerpo de la función
    # Puede contener una o más líneas de código
    return resultado
```

EJEMPLO:

```
def nombre_de_la_funcion(parametros):
    """Docstring de la función"""
    return resultado
```

01 **def**: Es una palabra clave que indica el inicio de la definición de la función.

02 **nombre_de_la_funcion**: Es el nombre que le damos a la función. Debe seguir las reglas de nombrado de variables en Python.

03 **parametros**: Son las variables que la función espera recibir cuando es llamada. Pueden ser opcionales.

04 **"""Docstring de la función"""**: Es un comentario de cadena (docstring) que describe brevemente la función y su funcionalidad. Es opcional, pero es una buena práctica incluirlo para documentar nuestras funciones.

05 **return *resultado***: La instrucción return indica el valor que la función devuelve cuando es llamada. Es opcional y puede devolver cualquier tipo de objeto, o incluso no devolver nada.

02

Parámetros vs Argumentos

Parámetros y Argumentos

Los **parámetros** son ***variables declaradas*** en la definición de la función que sirven como marcadores de posición para los valores que se pasan a la función cuando es llamada.

Los **argumentos** son los ***valores*** reales que se pasan a la función cuando es llamada. Estos valores se asignan a los parámetros correspondientes de la función.

```python
# a y b acá son parámetros (variables)
def suma(a, b):
    """Esta función suma dos números."""
    resultado = a + b
    return resultado

# 3 y 5 acá son los argumentos (valores)
resultado_suma = suma(3, 5)
print(resultado_suma)  # Output: 8
```

03

Consideraciones importantes

Consideraciones importantes

- El valor devuelto por una función se especifica mediante la instrucción return.
- Una función puede devolver cualquier tipo de objeto, o incluso no devolver nada.
- El valor devuelto por una función puede ser asignado a una variable o utilizado directamente en otra expresión.
- Una función puede tener múltiples parámetros, tanto obligatorios como opcionales.
- Los parámetros pueden tener valores predeterminados, lo que los convierte en parámetros opcionales.
- Una función puede devolver múltiples valores utilizando tuplas o estructuras de datos más complejas.

Ejemplo:

El parámetro **nombre** es *obligatorio* y debe proporcionarse al llamar a la función.
El parámetro **saludo** tiene un *valor predeterminado* de "Hola", por lo que es *opcional*.

La función *devuelve* un mensaje de saludo personalizado utilizando los valores proporcionados para **nombre** y **saludo**.

```python
def saludo(nombre, saludo="Hola"):
    """Esta función muestra un saludo personalizado."""
    mensaje = f"{saludo}, {nombre}!"
    return mensaje

# Llamando a la función con un solo argumento
print(saludo("Juan"))  # Output: Hola, Juan!

# Llamando a la función con dos argumentos
print(saludo("Sergie", "Buenos días"))  # Output: Buenos días, Sergie!
```

04 Scope y Namespaces

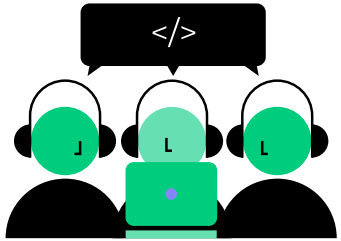
Alcance (Scope)

El alcance de una variable se refiere a la parte del programa en la que la variable es accesible. En Python, el alcance determina dónde en el código una variable puede ser utilizada o referenciada. El alcance de una variable está influenciado por dónde y cómo se define la variable en el código.

- **Alcance local**: Las variables definidas dentro de una función tienen un alcance local. Estas variables solo son accesibles dentro de la función en la que se definen. Una vez que la función termina de ejecutarse, las variables locales se eliminan de la memoria.
- **Alcance global**: Las variables definidas fuera de todas las funciones, es decir, en el nivel superior del script, tienen un alcance global. Estas variables son accesibles desde cualquier parte del código, incluidas las funciones. Sin embargo, es posible acceder a variables globales dentro de una función utilizando la palabra clave global.

Namespace

Un espacio de nombres (namespace) en Python es un contenedor que mapea nombres a objetos en el programa. Cada espacio de nombres tiene su propio alcance y contiene los nombres de las variables y sus respectivos objetos. Python utiliza varios espacios de nombres, incluidos el espacio de nombres global (módulo) y los espacios de nombres locales (funciones).



Ejemplo:

global_var es una variable global, por lo que es accesible dentro y fuera de la *función*.
local_var es una variable local, por lo que solo es accesible dentro de la función *mi_funcion*().

Dentro de la función, podemos acceder tanto a la **variable local** como a la **variable global**.

Fuera de la función, solo podemos acceder a la **variable global**.

```python
global_var = 10 # Variable global

def mi_funcion():
    local_var = 20 # Variable local
    print("Variable local dentro de la función:", local_var)
    print("Variable global dentro de la función:", global_var)

mi_funcion()
print("Variable global fuera de la función:", global_var)

# Output:
# Variable local dentro de la función: 20
# Variable global dentro de la función: 10
# Variable global fuera de la función: 10
```