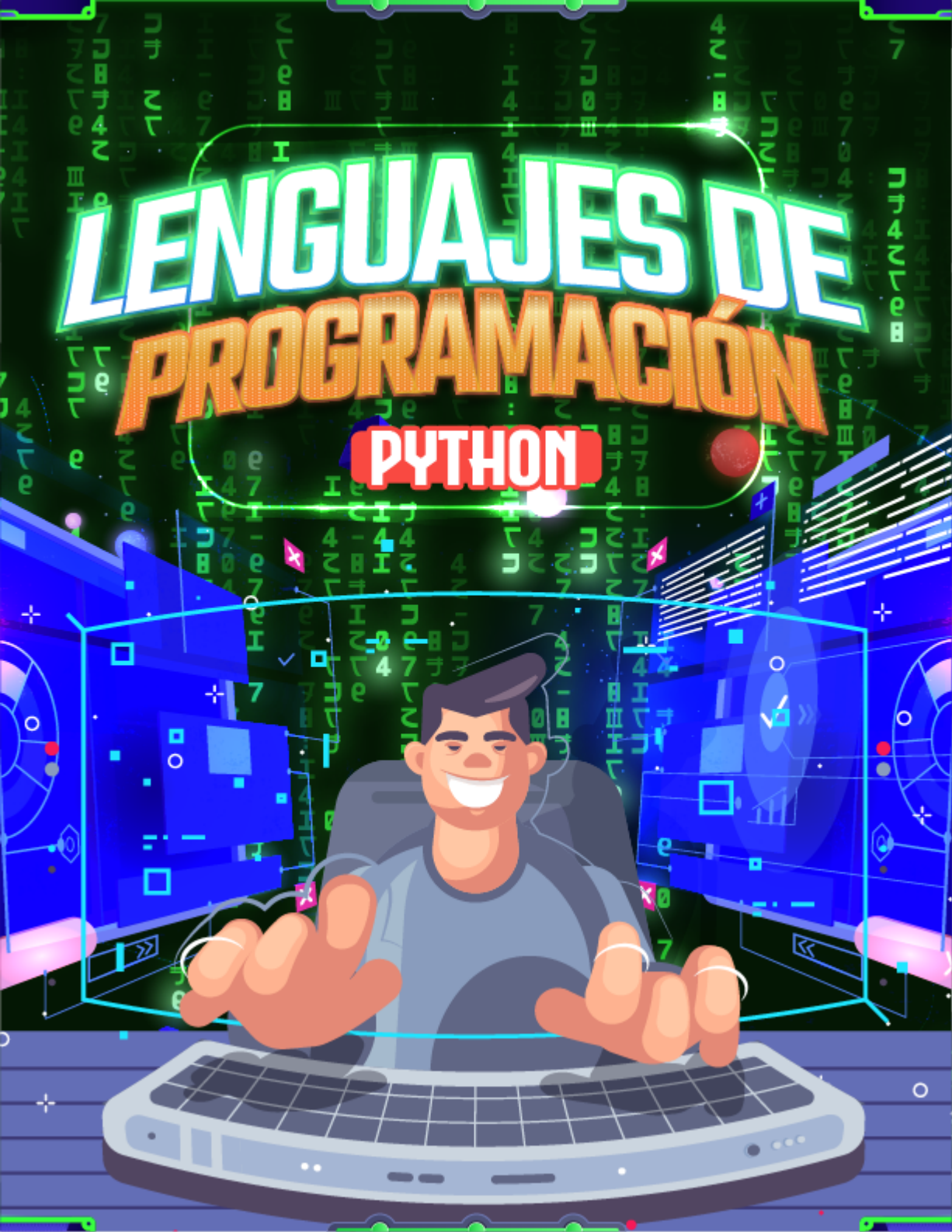
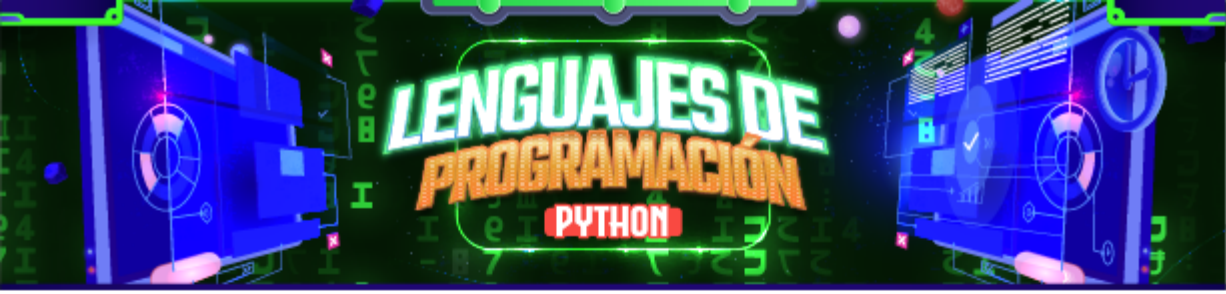
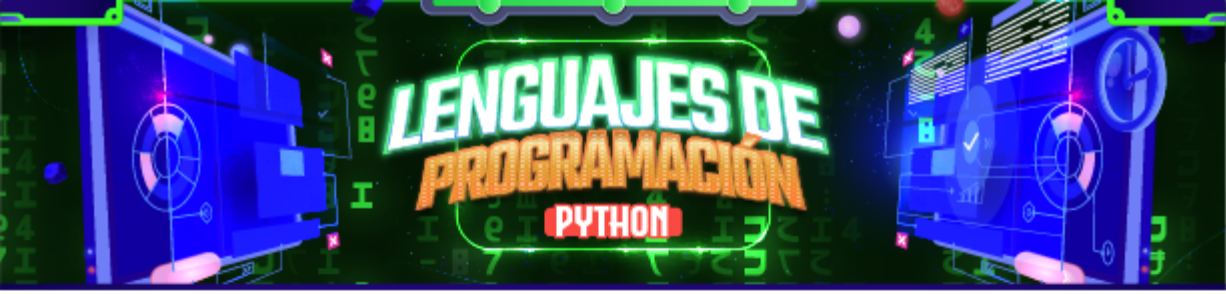

LENGUAJES DE
PROGRAMACIÓN
PYTHON



Funciones y los Procedimientos en Python

En la programación, las funciones y los procedimientos son herramientas clave que nos permiten organizar y reutilizar bloques de código. Estas construcciones nos ayudan a dividir un programa en partes más pequeñas y manejables, lo que facilita su comprensión y mantenimiento. En esta lección, exploraremos las funciones y los procedimientos en Python, aprendiendo cómo definirlos, invocarlos y utilizarlos para mejorar la estructura y eficiencia de nuestros programas.



Funciones en Python

Una función en Python es un bloque de código reutilizable que realiza una tarea específica. Permite agrupar una serie de instrucciones relacionadas y darles un nombre para que puedan ser invocadas desde cualquier parte del programa. Las funciones ayudan a reducir la duplicación de código, promueven la modularidad y facilitan el mantenimiento.

Para definir una función en Python, utilizamos la palabra clave "def" seguida del nombre de la función y paréntesis que pueden contener parámetros. A continuación, se muestra la sintaxis básica de una función:

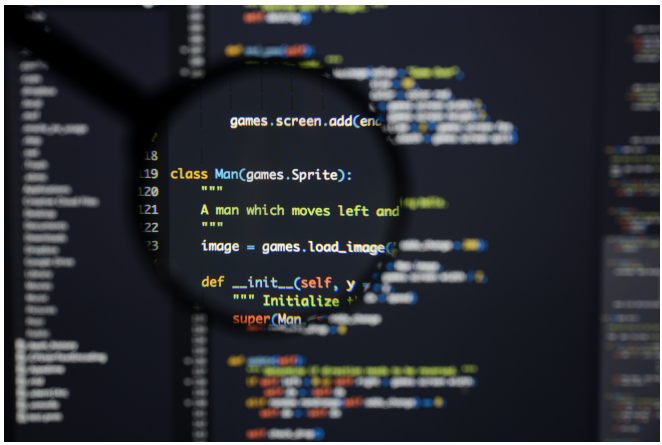
```
def nombre_de_funcion(parametros):
    # bloque de código de la función
    # instrucciones a ejecutar
    return valor_de_retorno
```

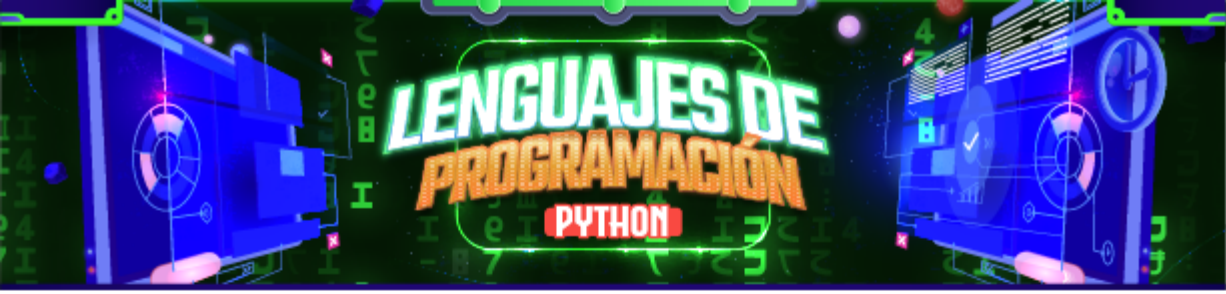
En este ejemplo, la función calcular_area_rectangulo recibe dos parámetros, base y altura. Calcula el área del rectángulo multiplicando ambos valores y devuelve el resultado.

Para invocar o llamar a una función, simplemente escribimos su nombre seguido de paréntesis y, si es necesario, pasamos los valores de los argumentos. Por ejemplo:

```
resultado = calcular_area_rectangulo(5, 3)
print(resultado) # Imprime 15
```

En este caso, la función calcular_area_rectangulo se invoca con los argumentos 5 y 3. El resultado se asigna a la variable resultado y se imprime en la pantalla.





Procedimientos en Python

Un procedimiento en Python es similar a una función, pero no devuelve un valor. Es un bloque de código reutilizable que realiza una serie de instrucciones sin necesidad de un valor de retorno. Los procedimientos también ayudan a organizar el código y evitar la duplicación.

La sintaxis para definir un procedimiento es similar a la de una función, pero no incluye la palabra clave return. A continuación, se muestra un ejemplo de un procedimiento que muestra un mensaje de saludo:

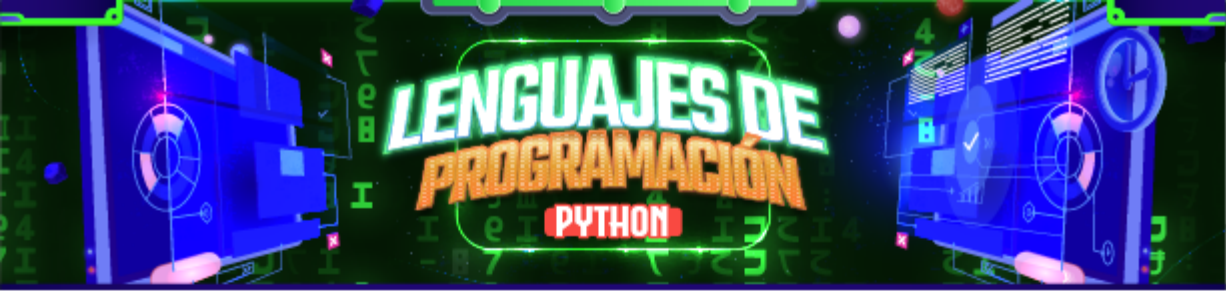
```
def saludar(nombre):
    print("Hola,", nombre)
```

En este caso, el procedimiento saludar recibe un parámetro nombre y muestra en la pantalla un mensaje de saludo utilizando ese nombre.

Para invocar un procedimiento, se utiliza la misma sintaxis que para una función:

```
saludar("Ana") # Imprime "Hola, Ana"
```

En este ejemplo, el procedimiento saludar se invoca con el argumento "Ana" y muestra el mensaje de saludo en la pantalla.



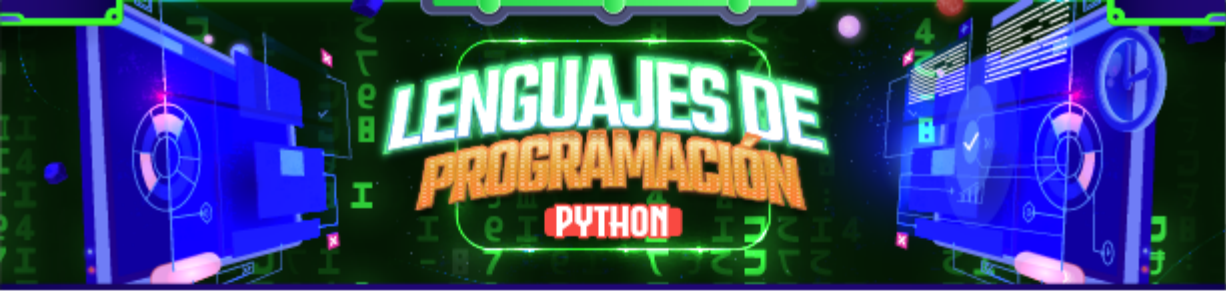
Ventajas de utilizar funciones y procedimientos

Reutilización de código: Al definir funciones y procedimientos, podemos escribir un bloque de código una vez y utilizarlo en múltiples lugares del programa. Esto evita la duplicación de código y facilita el mantenimiento.

Modularidad: Las funciones y los procedimientos nos permiten dividir un programa en partes más pequeñas y manejables. Cada función o procedimiento se encarga de una tarea específica, lo que facilita la comprensión y el diseño modular del programa.

Abstracción: Las funciones y los procedimientos nos permiten abstraer la lógica y los detalles internos de un bloque de código. Al utilizar funciones, podemos enfocarnos en el "qué" en lugar del "cómo", lo que facilita la lectura y el uso del código.

Facilidad de mantenimiento: Al dividir un programa en funciones y procedimientos más pequeños y específicos, es más fácil realizar cambios y actualizaciones en el código. Si se requiere una modificación, solo es necesario realizarla en un lugar, lo que mejora la eficiencia y reduce el riesgo de introducir errores.



Paso de parámetros a funciones y procedimientos

En Python, podemos pasar valores a funciones y procedimientos utilizando parámetros. Los parámetros son variables que actúan como marcadores de posición para los valores que se pasarán a la función o procedimiento cuando se invoquen. Hay dos tipos de parámetros: parámetros posicionales y parámetros con nombre.

Los parámetros posicionales se definen en el orden en que se esperan los valores al llamar a la función o procedimiento. Por ejemplo, considera la siguiente función que suma dos números:

```
def sumar(a, b):
    resultado = a + b
    return resultado
```

En este caso, los parámetros a y b son parámetros posicionales. Al invocar la función, debemos proporcionar los valores en el mismo orden:

```
resultado = sumar(5, 3)
print(resultado) # Imprime 8
```

Aquí, 5 se asigna a a y 3 se asigna a b.

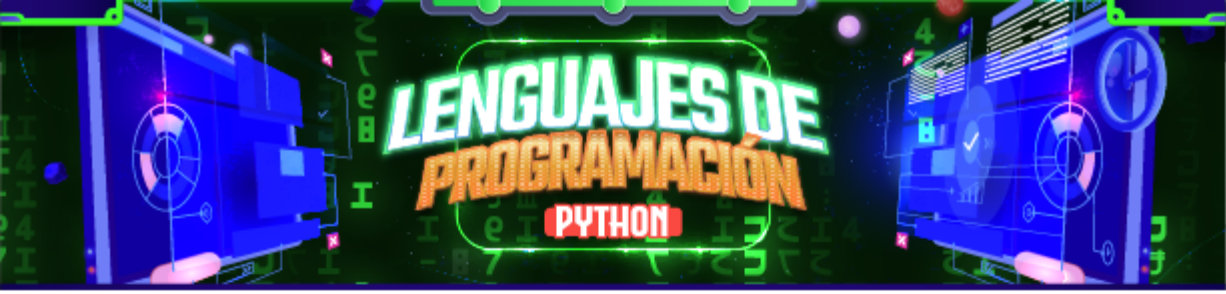
Los parámetros con nombre se especifican junto con su valor correspondiente al llamar a la función o procedimiento. Esto nos permite asignar valores específicos a los parámetros, incluso si están fuera de orden. Por ejemplo:

```
def saludar(nombre, saludo):
    mensaje = saludo + ", " + nombre
    print(mensaje)
```

En este caso, los parámetros nombre y saludo son parámetros con nombre. Al invocar la función, podemos especificar los valores de la siguiente manera:

```
saludar(nombre="Ana", saludo="¡Hola")
```

Aquí, se asigna "Ana" a nombre y "¡Hola" a saludo. El orden de los parámetros no importa en este caso.



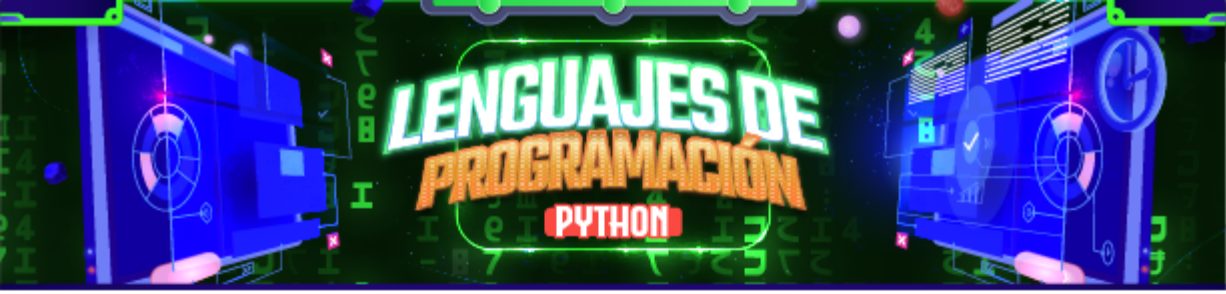
Además de los parámetros posicionales y con nombre, también podemos tener parámetros predeterminados en una función o procedimiento. Estos parámetros tienen un valor asignado por defecto y pueden ser omitidos al invocar la función si no se proporciona un valor explícito. Por ejemplo:

```
def saludar(nombre, saludo="Hola"):
    mensaje = saludo + ", " + nombre
    print(mensaje)
```

En este caso, el parámetro saludo tiene un valor predeterminado de "Hola". Si no se proporciona un valor para saludo al invocar la función, se utilizará el valor predeterminado:

```
saludar("Ana") # Imprime "Hola, Ana"
saludar("Pedro", "¡Hola!") # Imprime "¡Hola!,
Pedro"
```

Aquí, en la primera llamada a saludar, se utiliza el valor predeterminado de saludo. En la segunda llamada, se proporciona un valor explícito y se utiliza ese valor en lugar del predeterminado.



Retorno de valores en funciones

Las funciones en Python pueden devolver valores utilizando la palabra clave return. Un valor de retorno es el resultado que la función produce y se puede utilizar en otras partes del programa. Por ejemplo:

```
def multiplicar(a, b):
    resultado = a * b
    return resultado
```

En este caso, la función multiplicar recibe dos parámetros, a y b, y devuelve el resultado de la multiplicación.

Al invocar la función, podemos almacenar el valor de retorno en una variable y utilizarlo posteriormente:

```
resultado = multiplicar(5, 3)
print(resultado) # Imprime 15
```

Aquí, el resultado de la multiplicación se asigna a la variable resultado y se imprime en la pantalla.

Es importante tener en cuenta que una función puede tener múltiples declaraciones de return, pero solo se ejecutará la primera vez que se encuentre. Después de que se ejecute una declaración de return, la función finalizará y no ejecutará ninguna instrucción adicional.

Las funciones y los procedimientos son herramientas poderosas en Python que nos permiten organizar y reutilizar el código de manera eficiente. Al dividir un programa en funciones y procedimientos más pequeños y específicos, podemos mejorar la modularidad, la reutilización y la mantenibilidad del código.

Recuerda seguir las convenciones de nombrado de funciones, documentar tus funciones y procedimientos, y utilizar comentarios para explicar el propósito y la lógica de cada bloque de código.