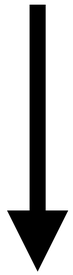
CONQUER BLOCKS

PYTHON

PARADIGMAS DE PROGRAMACIÓN

CODIFICAMOS LA INFORMACION EN SUCESIONES DE 0s y 1s



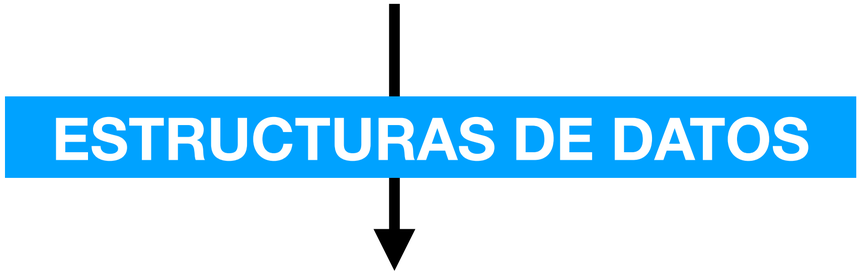
ESTRUCTURACION, MANIPULACION Y ALMACENAMIENTO DE DATOS
(Listas, arrays, tuplas, sets y diccionarios / bucles y condicionales)

CODIFICAMOS LA INFORMACION EN SUCESIONES DE 0s y 1s

ESTRUCTURAS DE DATOS

ESTRUCTURACION, MANIPULACION Y ALMACENAMIENTO DE DATOS
(Listas, arrays, tuplas, sets y diccionarios / bucles y condicionales)

CODIFICAMOS LA INFORMACION
EN SUCESIONES DE 0s y 1s



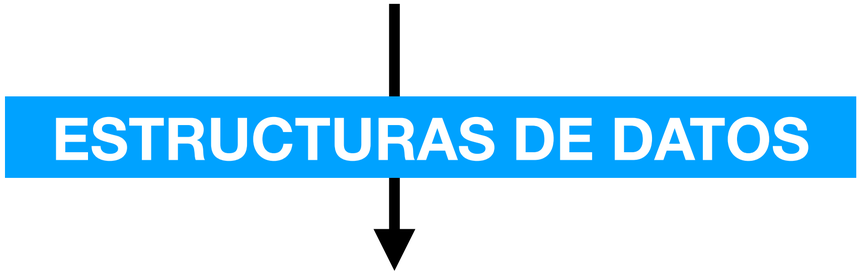
ESTRUCTURACION, MANIPULACION Y ALMACENAMIENTO DE DATOS
(Listas, arrays, tuplas, sets y diccionarios / bucles y condicionales)

LA PROGRAMACION SE BASA EN CREAR UNA RECETA O SECUENCIA DE INSTRUCCIONES

MANIPULAREMOS LA INFORMACION (LOS DATOS) PARA OBTENER UN RESULTADO DE LA MANERA MAS ÓPTIMA POSIBLE

CODIFICAMOS LA INFORMACION
EN SUCESIONES DE 0s y 1s

LA PROGRAMACION SE BASA EN CREAR UNA
RECETA O SECUENCIA DE INSTRUCCIONES



ESTRUCTURACION, MANIPULACION Y
ALMACENAMIENTO DE DATOS
(Listas, arrays, tuplas, sets y diccionarios /
bucles y condicionales)

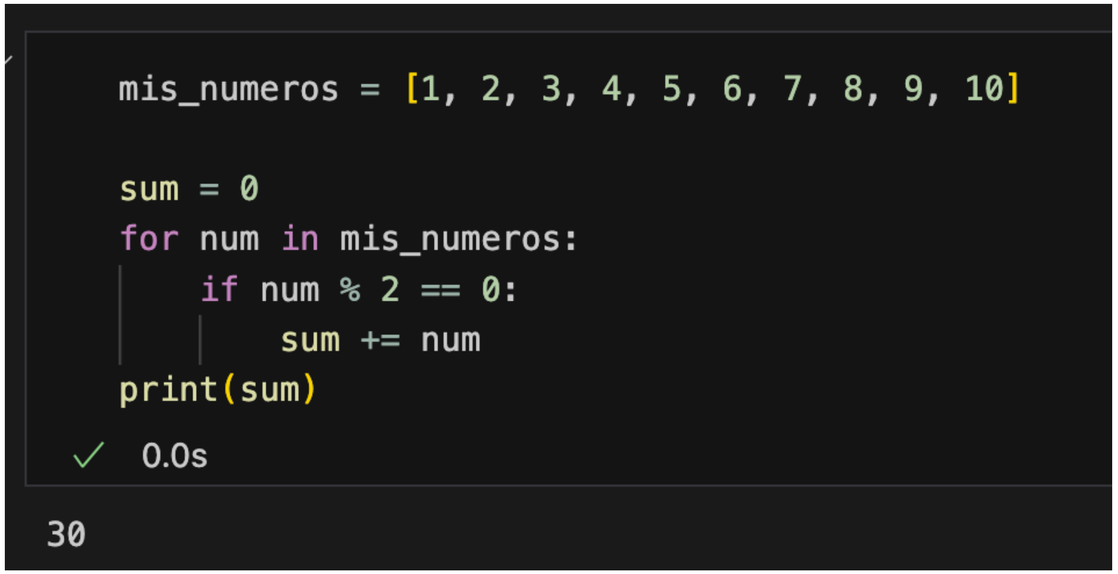
MANIPULAREMOS LA INFORMACION (LOS
DATOS) PARA OBTENER UN RESULTADO DE LA
MANERA MAS ÓPTIMA POSIBLE

PARADIGMAS DE PROGRAMACIÓN

Enfoques o modelos que definen la forma en la que se deben diseñar, estructurar y escribir los programas

PARADIGMAS DE PROGRAMACIÓN

Programación imperativa



Foco: Cómo se deben ejecutar las instrucciones. Estructuración en una secuencia de comandos que modifican el estado del programa y realizan acciones específicas.

PARADIGMAS DE PROGRAMACIÓN

Programación funcional

```
mis_numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
numeros_pares = filter(lambda x: x % 2 == 0, mis_numeros)
suma = sum(numeros_pares)
print(suma)
```

✓ 0.0s

```
30
```

Foco: Evaluación de funciones matemáticas. Enfoque en la inmutabilidad y en evitar cambios de estado o sus efectos secundarios.

PARADIGMAS DE PROGRAMACIÓN

Programación estructurada

```
mis_numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

sum = 0
for num in mis_numeros:
    if num % 2 == 0:
        sum += num
print(sum)
```

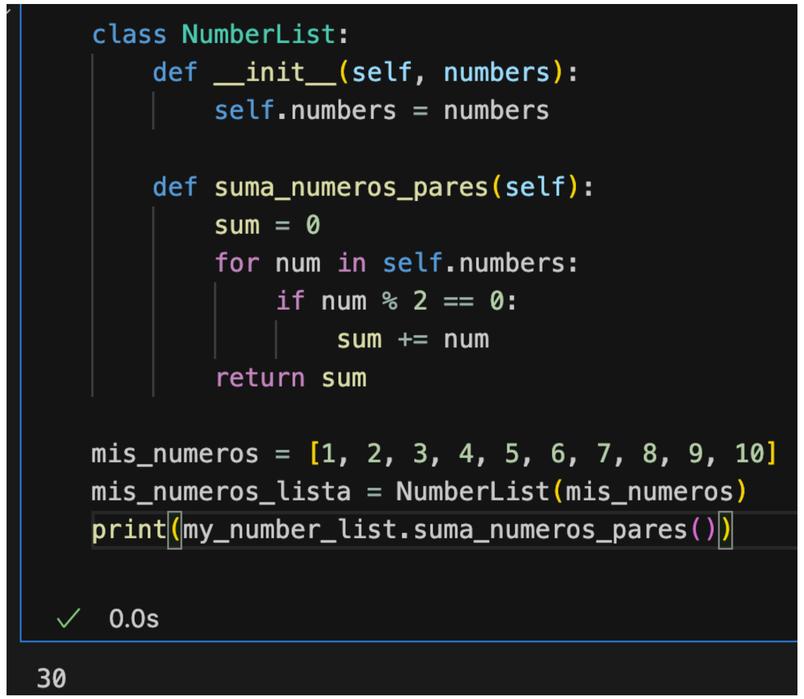
✓ 0.0s

```
30
```

Foco: División en estructuras de control como bucles y decisiones (if statements). Se busca la claridad y la organización del código evitando el uso de saltos no estructurados.

PARADIGMAS DE PROGRAMACIÓN

Programación orientada a objetos (POO)



```python
class NumberList:
    def __init__(self, numbers):
        self.numbers = numbers

    def suma_numeros_pares(self):
        sum = 0
        for num in self.numbers:
            if num % 2 == 0:
                sum += num
        return sum

mis_numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
mis_numeros_lista = NumberList(mis_numeros)
print(my_number_list.suma_numeros_pares())
```

✓ 0.0s

```
30
```

Foco: Los programas se organizan alrededor de objetos que representan entidades del mundo real. Encapsulamiento de datos y comportamientos en objetos. Conceptos particulares: herencia y polimorfismo.

PARADIGMAS DE PROGRAMACIÓN

Muchos lenguajes de programación admiten múltiples paradigmas y permiten combinarlos según sea necesario.

La elección del paradigma depende del problema a resolver, la preferencia del desarrollador y los requisitos del proyecto.

CONQUER BLOCKS

PYTHON

INTRODUCCIÓN AL USO DE FUNCIONES

¿Qué es una función?

Bloques de código a los que asignamos un nombre

Están diseñados para realizar una tarea en específico

Pueden ser usados repetidamente

¿Qué es una función?

```
mi_lista = [1,2,3,4]
longitud = len(mi_lista)
print(longitud)
```

✓ 0.0s

```
4
```

¿Qué es una función?

```
mi_lista = [1,2,3,4]
longitud = len(mi_lis
print(longitud)
```

✓ 0.0s

```
4
```

```
import numpy as np

array_1 = np.array([2,2,2])
array_2 = np.array([1,2,3])

array_suma = np.multiply(array_1, array_2)

print(array_suma)
```

✓ 0.0s

```
[2 4 6]
```

¿Qué es una función?

```
mi_lista = [1,2,3,4]
longitud = len(mi_lis
print(longitud)
```

✓ 0.0s

```
4
```

```
import numpy as np

array_1 = np.array([2,2,2])
array_2 = np.array([1,2,3])

array_suma = np.multiply(array_1, array_2)

print(array_suma)
```

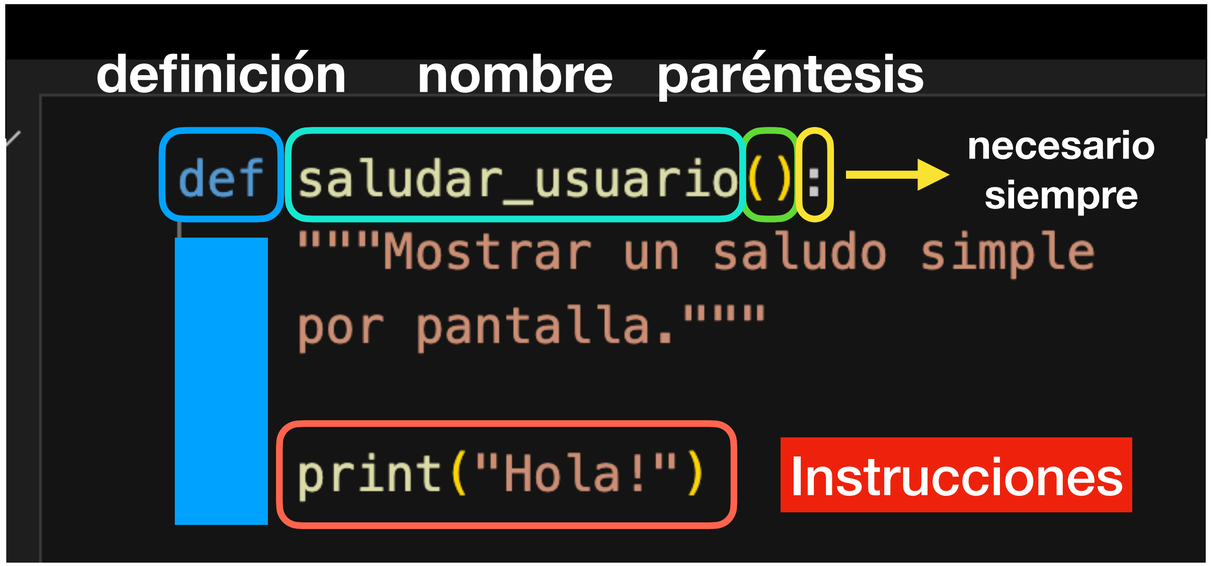
✓ 0.0s

```
[2 4 6]
```

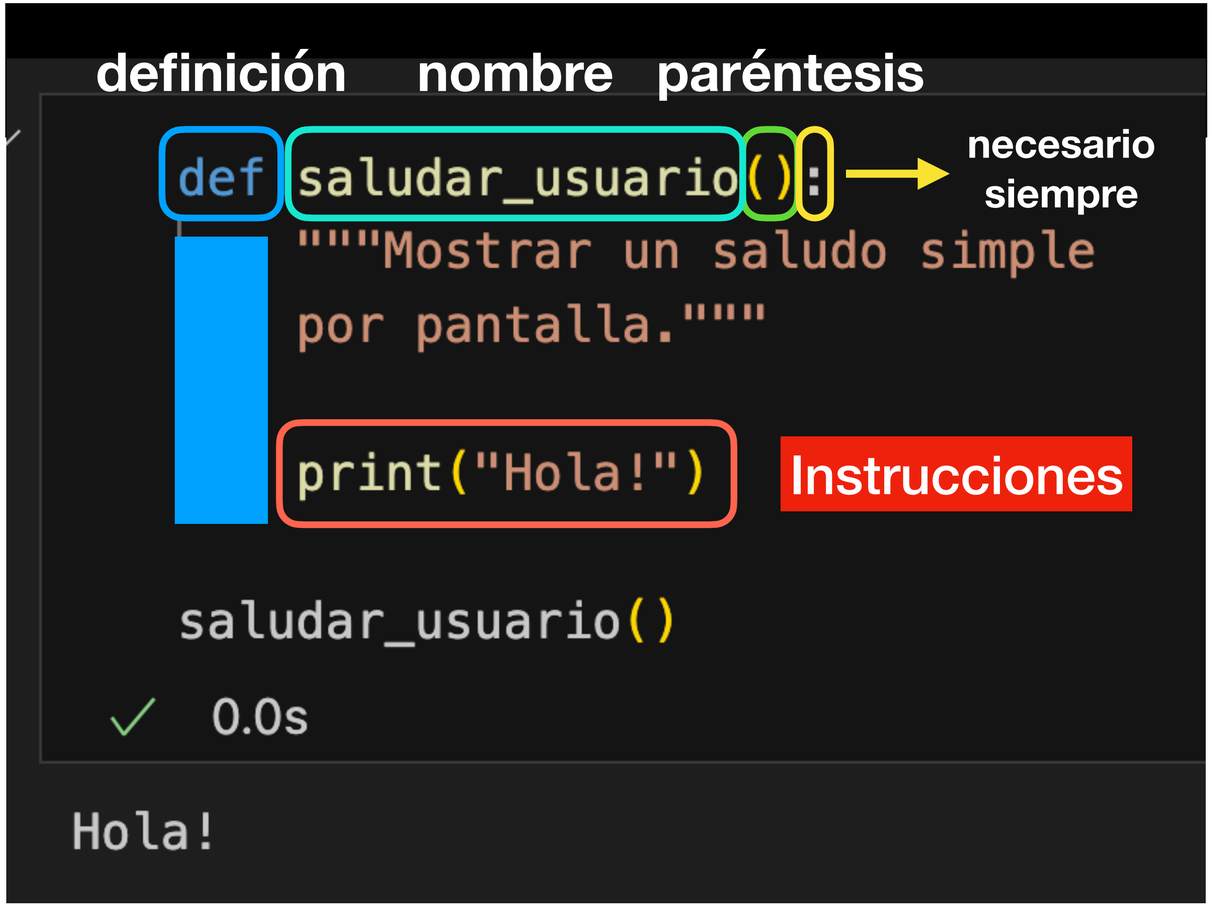
¿Cómo creamos una función?

```
def saludar_usuario():
    """Mostrar un saludo simple
    por pantalla."""

    print("Hola!")
```

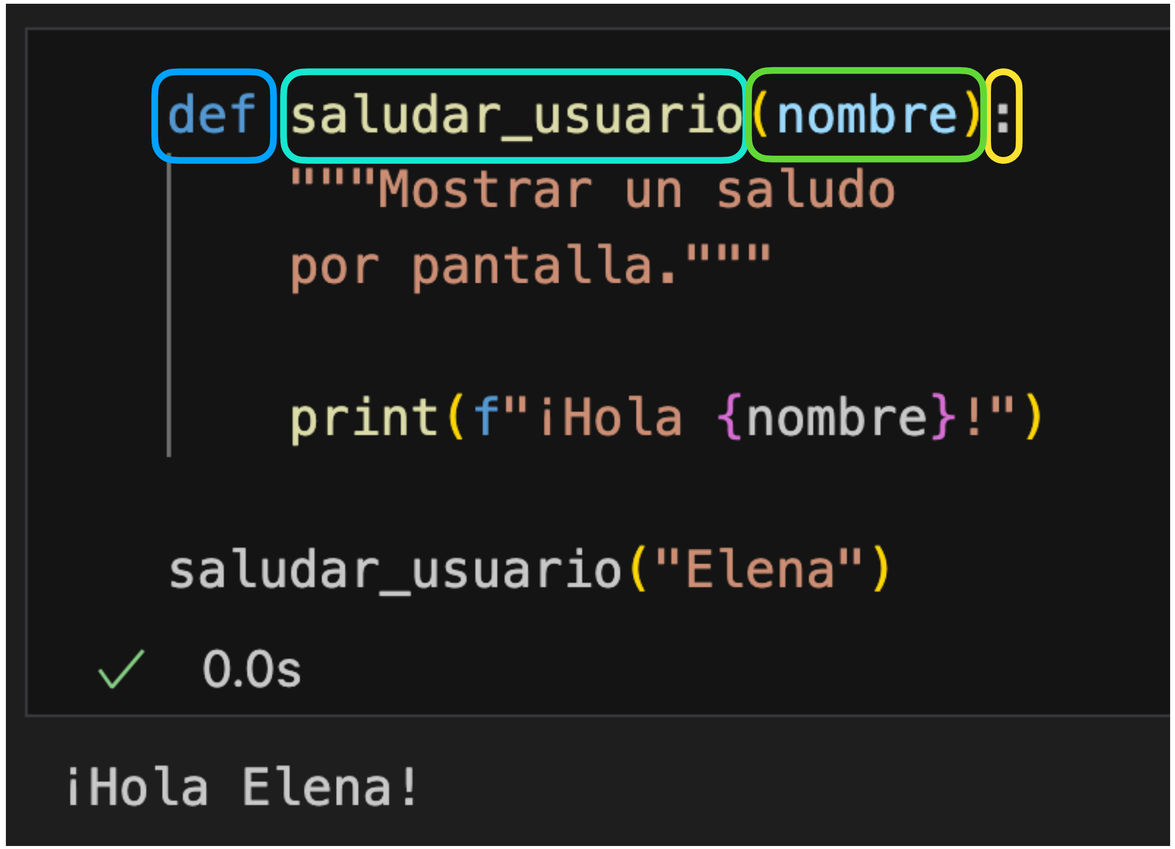
¿Cómo creamos una función?



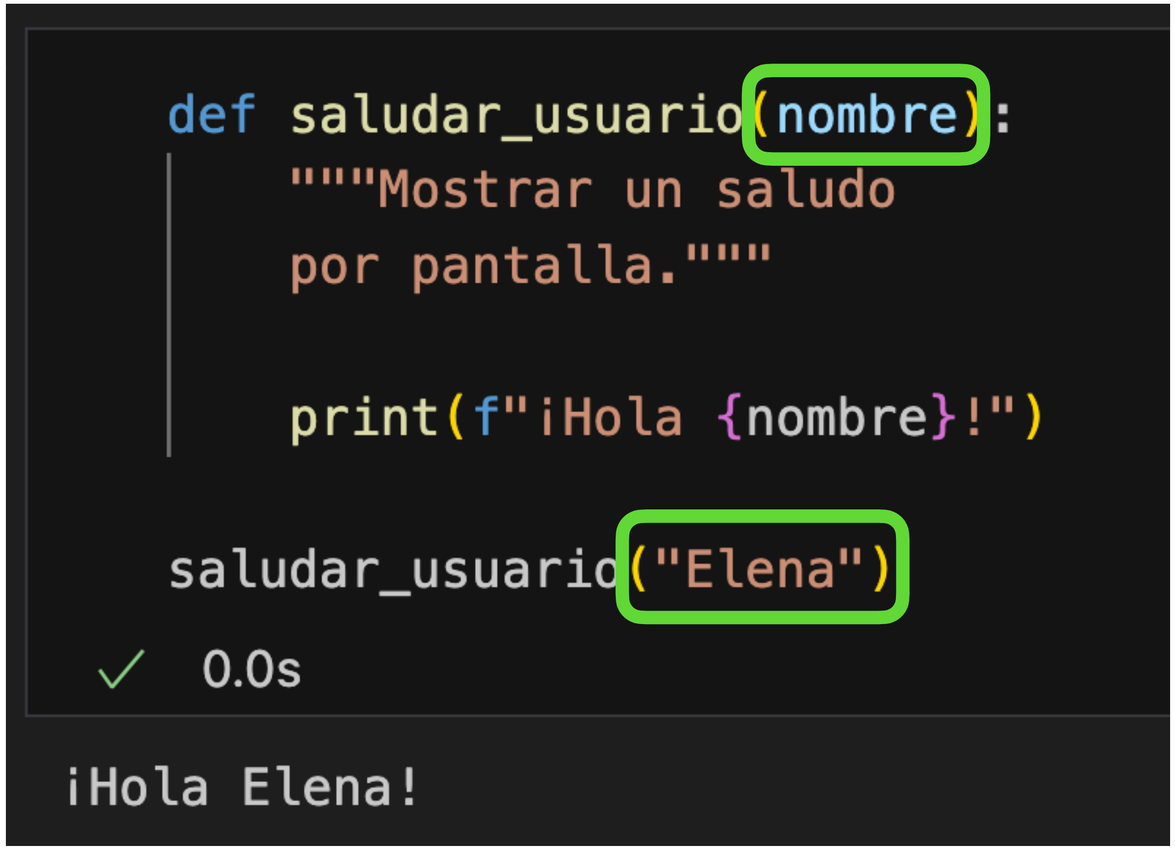
¿Cómo creamos una función?



¿Cómo pasamos información a una función?



¿Cómo pasamos información a una función?



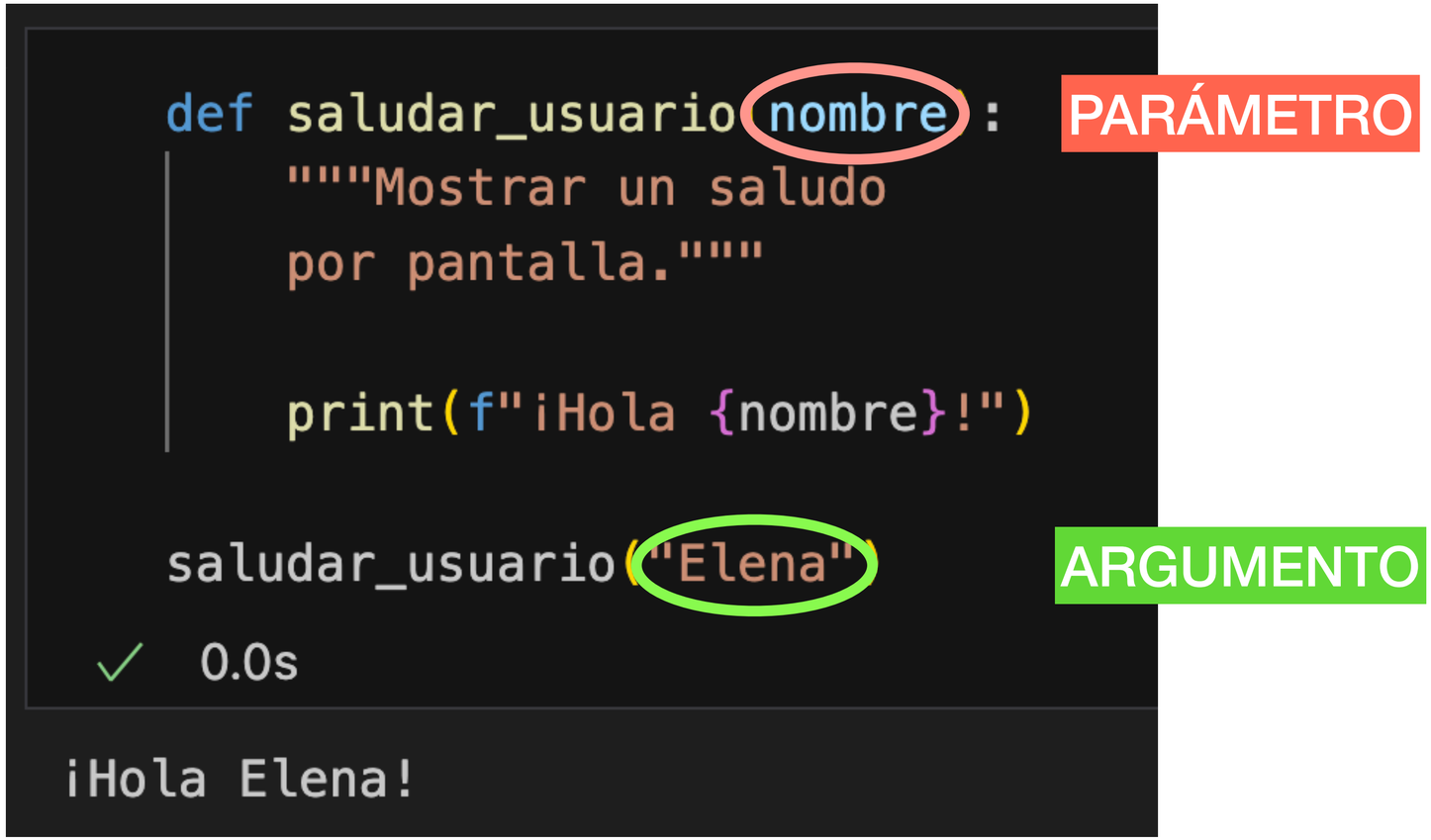
¿Cómo pasamos información a una función?

```python
def saludar_usuario(nombre):
    """Mostrar un saludo
    por pantalla."""

    print(f"¡Hola {nombre}!")

saludar_usuario("Elena")
saludar_usuario("Maria")
saludar_usuario("Enrique")
```

✓ 0.0s

```
¡Hola Elena!
¡Hola Maria!
¡Hola Enrique!
```

¿Cómo pasamos información a una función?



ARGUMENTOS POSICIONALES

```
def describir_mascota(tipo_animal, nombre_mascota):
    """ Mostrar informacion de mascota."""

    print("Tengo un " + tipo_animal + "." )
    print("Mi " + tipo_animal + " se llama " \
        + nombre_mascota.title() + ".")

describir_mascota("hamster", "harry")
```

✓ 0.0s

```
Tengo un hamster.
Mi hamster se llama Harry.
```

ARGUMENTOS POSICIONALES

```
def describir_mascota(tipo_animal, nombre_mascota):
    """ Mostrar informacion de mascota."""

    print("Tengo un " + tipo_animal + "." )
    print("Mi " + tipo_animal + " se llama " \
        + nombre_mascota.title() + ".")

describir_mascota( "harry", "hamster",)
```

✓ 0.0s

```
Tengo un harry.
Mi harry se llama Hamster.
```

ARGUMENTOS DE PALABRA CLAVE

```
def describir_mascota(tipo_animal, nombre_mascota):
    """ Mostrar informacion de mascota."""

    print("Tengo un " + tipo_animal + "." )
    print("Mi " + tipo_animal + " se llama " \
        + nombre_mascota.title() + ".")

describir_mascota(tipo_animal="hamster", nombre_mascota="harry")
```

✓ 0.0s

```
Tengo un hamster.
Mi hamster se llama Harry.
```

ARGUMENTOS DE PALABRA CLAVE

```
def describir_mascota(tipo_animal, nombre_mascota):
    """ Mostrar informacion de mascota."""

    print("Tengo un " + tipo_animal + "." )
    print("Mi " + tipo_animal + " se llama " \
        + nombre_mascota.title() + ".")

describir_mascota(nombre_mascota="harry", tipo_animal="hamster")
```

✓ 0.0s

```
Tengo un hamster.
Mi hamster se llama Harry.
```

VALORES POR DEFECTO

```
def describir_mascota(nombre_mascota, tipo_animal = "perro"):
    """ Mostrar informacion de mascota."""

    print("Tengo un " + tipo_animal + "." )
    print("Mi " + tipo_animal + " se llama " \
        + nombre_mascota.title() + ".")

describir_mascota(nombre_mascota="Roc")
```

✓ 0.0s

```
Tengo un perro.
Mi perro se llama Roc.
```

VALORES POR DEFECTO

```
def describir_mascota(nombre_mascota, tipo_animal = "perro"):
    """ Mostrar informacion de mascota."""

    print("Tengo un " + tipo_animal + "." )
    print("Mi " + tipo_animal + " se llama " \
        + nombre_mascota.title() + ".")

describir_mascota(tipo_animal = "hamster", nombre_mascota="harry")
```

✓ 0.0s

```
Tengo un hamster.
Mi hamster se llama Harry.
```

VALORES POR DEFECTO

```
def describir_mascota(tipo_animal = "perro", nombre_mascota):
    """ Mostrar informacion de mascota."""

    print("Tengo un " + tipo_animal + "." )
    print("Mi " + tipo_animal + " se llama " \
        + nombre_mascota.title() + ".")

describir_mascota(nombre_mascota="Roc")
```

⊗ 0.0s

```
  Cell In[24], line 1
    def describir_mascota(tipo_animal = "perro", nombre_mascota):
                                                                 ^
SyntaxError: non-default argument follows default argument
```

LLAMADAS EQUIVALENTES

```
def describir_mascota(nombre_mascota, tipo_animal = "perro"):
    """ Mostrar informacion de mascota."""

    print("Tengo un " + tipo_animal + "." )
    print("Mi " + tipo_animal + " se llama " \
        + nombre_mascota.title() + ".")
```

```
# un perro llamado Roc
describir_mascota(nombre_mascota="roc")
print("----")
describir_mascota("roc")

# un hamster llamado harry
print("======")
describir_mascota("harry", "hamster")
print("----")
describir_mascota(nombre_mascota = "harry", tipo_animal = "hamster")
print("----")
describir_mascota(tipo_animal = "hamster", nombre_mascota = "harry")
```

```
Tengo un perro.
Mi perro se llama Roc.
----
Tengo un perro.
Mi perro se llama Roc.
======
Tengo un hamster.
Mi hamster se llama Harry.
----
Tengo un hamster.
Mi hamster se llama Harry.
----
Tengo un hamster.
Mi hamster se llama Harry.
```

ERRORES FRECUENTES

```
    def describir_mascota(tipo_animal = "perro", nombre_mascota):
        """ Mostrar informacion de mascota."""

        print("Tengo un " + tipo_animal + "." )
        print("Mi " + tipo_animal + " se llama " \
            + nombre_mascota.title() + ".")

    describir_mascota(nombre_mascota="Roc")
```

⊗ 0.0s

```
  Cell In[24], line 1
    def describir_mascota(tipo_animal = "perro", nombre_mascota):
                                                                 ^
SyntaxError: non-default argument follows default argument
```

ERRORES FRECUENTES

```
    def describir_mascota(nombre_mascota, tipo_animal = "perro"):
        """ Mostrar informacion de mascota."""

        print("Tengo un " + tipo_animal + "." )
        print("Mi " + tipo_animal + " se llama " \
            + nombre_mascota.title() + ".")

    describir_mascota()
```

⊗ 0.0s

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[31], line 8
      4     print("Tengo un " + tipo_animal + "." )
      5     print("Mi " + tipo_animal + " se llama " \
      6         + nombre_mascota.title() + ".")
----> 8 describir_mascota()

TypeError: describir_mascota() missing 1 required positional argument: 'nombre_mascota'
```

ERRORES FRECUENTES

```
    def describir_mascota(nombre_mascota = "willy", tipo_animal = "perro"):
        """ Mostrar informacion de mascota."""

        print("Tengo un " + tipo_animal + "." )
        print("Mi " + tipo_animal + " se llama " \
            + nombre_mascota.title() + ".")

    describir_mascota()
```

✓ 0.0s

```
Tengo un perro.
Mi perro se llama Willy.
```

```
def describir_mascota(nombre_mascota, tipo_animal = "perro"):
    """ Mostrar informacion de mascota."""

    print("Tengo un " + tipo_animal + "." )
    print("Mi " + tipo_animal + " se llama " \
        + nombre_mascota.title() + ".")

describir_mascota("willy")
```

✓ 0.0s

```
Tengo un perro.
Mi perro se llama Willy.
```

REPASO

1. Paradigmas de programación

2. Qué es una función y como crearla

3. Trabajar con parámetros y argumentos

CONQUER BLOCKS