SESIÓN 13 · MÓDULO 1

Funciones

Dar nombre a un trozo de programa, pasarle datos y recibir un resultado, y entender qué ve cada función.

Antonio Chanes Espigares · CC BY-NC-ND 4.0

Hoy

0:00	Corrección de la tarea 12 y repaso del hito 1.
0:15	Situación: una receta que se usa en muchos platos; «haz la bechamel» resume seis pasos.
0:25	**Predice** y visualizador: cada llamada abre su propio marco.
0:40	Construcción: parámetros, `return`, valores por defecto, argumentos con nombre, `*args`, ámbito, `global`,…
1:20	Ejercicios de los dos capítulos.
1:45	Preguntas tipo examen y tarea 13.

Para empezar

Una receta que se usa en muchos platos; «haz la bechamel» resume seis pasos.

Predice

Escribe en papel qué crees que pasará. Después, lo ejecutamos.

```python
def doble(x):
    x = x * 2
    return x

n = 5
print(doble(n), n)

def saluda(nombre, saludo="Hola"):
    print(saludo, nombre)

saluda("Ana")
saluda("Luis", saludo="Buenas")
```

Predice

Escribe en papel qué crees que pasará. Después, lo ejecutamos.

```python
def doble(x):
    x = x * 2
    return x

n = 5
print(doble(n), n)

def saluda(nombre, saludo="Hola"):
    print(saludo, nombre)

saluda("Ana")
saluda("Luis", saludo="Buenas")
```

LO QUE SALE

`10 5` · `Hola Ana` · `Buenas Luis`

La `x` de la función es otra variable: `n` sigue valiendo 5.

El área del rectángulo

Funciones · 1

```python
def area_rectangulo(base, altura):
    area = base * altura
    return area

a = area_rectangulo(3, 4)
b = area_rectangulo(2, 5)
print(a + b)
```

`print` no es `return`

Funciones · 2

```
def suma_mal(a, b):
    print(a + b)

r = suma_mal(2, 3)
print(r)
```

Funciones que usan funciones

Funciones · 3

```
def cuadrado(x):
    return x * x

def hipotenusa(a, b):
    suma = cuadrado(a) + cuadrado(b)
    return suma ** 0.5

h = hipotenusa(3, 4)
print(h)
```

Valores por defecto y argumentos por nombre

Funciones a fondo · 1

```
for llamada in ["precio_final()", "precio_final(100, 10, 5, 3)", "precio_final(100, precio=50)"]:
    try:
        eval(llamada)
    except TypeError as e:
        print(llamada, "→", e)
```

Cualquier número de argumentos: `*args`

Funciones a fondo · 2

```
def media(*notas):
    total = 0
    for n in notas:
        total += n
    return total / len(notas)

print(media(6, 8, 10))
print(media(5))
```

Manos a la obra

Abre el cuaderno de la sesión en Colab, con los ejemplos de hoy y estos ejercicios:

1. Escribe una función celsius_a_fahrenheit(c) que devuelva c * 9 / 5 +…
2. ¿Qué imprime?
3. Corrige suma_mal para que r valga 5.
4. ¿Válido o no? a) def f(a, b=1, c):; b) f(1, c=3) para def f(a, b=1,…
5. ¿Qué imprime?
6. Suma flexible. Escribe suma_todo(*numeros) que devuelva la suma de…



Cuaderno en Colab

Tarea 13 · Refactorizar en funciones

1. `segundos_a_hms(total)` devuelve una tupla `(h, m, s)` (tarea 5).
2. `calificacion(nota)` devuelve el texto de la calificación (tarea 6).
3. `media_de(*numeros)` acepta cualquier cantidad de números y devuelve su media, o `None` si no recibe ninguno.
4. Debajo, prueba cada función con los casos de las tareas originales.

COMPROBACIÓN `segundos_a_hms(4000)` devuelve `(1, 6, 40)` y `media_de()` devuelve `None`.

El proyecto

Hito 2 del proyecto. Las funciones del gestor de notas, probadas con datos escritos a mano.

Cuaderno del proyecto

Antes de irte

Las **preguntas tipo examen** del capítulo, en directo: se corrigen al momento.



Preguntas del capítulo

LA PRÓXIMA SESIÓN

Sesión 14 · Errores, ficheros, CSV y módulos