Rutinas y Subrutinas en Python - Parte 2

Materia: Algoritmos y Programación

Prof. D.Sc. BARSEKH-ONJI Aboud

Facultad de Ingeniería
Universidad Anáhuac México

20 de octubre de 2025

Agenda

1. Funciones y Valor de Retorno

Recordatorio: ¿Qué es una Subrutina?

Lo que vimos la clase pasada

Una **subrutina** (o función) es un bloque de código que **hace una tarea**. Por lo general, su resultado es una acción visible, como imprimir en pantalla.

```
def imprimir_tabla(numero_tabla):
    print(f"\n--- Tabla del {numero_tabla} ---")
    for i in range(1, 11):
        # ... lineas de codigo ...
        print(f"{numero_tabla} x {i} = ...")

# La llamamos y "hace algo" (imprime)
imprimir_tabla(7)

```

Recordatorio: ¿Qué es una Subrutina?

La Limitación

¿Pero qué pasa si no quiero imprimir la tabla, sino que quiero **guardar** el resultado de `7 x 8` en una variable para usarlo después? La subrutina `imprimir_tabla` no me "entrega" ningún dato.

¿Qué es una Función?

Definición: Función

Una **función** es un bloque de código que, además de recibir argumentos, está diseñada para **calcular un valor** y **devolverlo** al programa principal.

La Analogía del Trabajador

- Una **Subrutina** (clase pasada) es como un trabajador al que le das una orden ("¡imprime la tabla del 5!") y él la cumple.

¿Qué es una Función?

Definición: Función

Una **función** es un bloque de código que, además de recibir argumentos, está diseñada para **calcular un valor** y **devolverlo** al programa principal.

La Analogía del Trabajador

- Una **Subrutina** (clase pasada) es como un trabajador al que le das una orden ("¡imprime la tabla del 5!") y él la cumple.
- Una **Función** (clase de hoy) es como un trabajador al que le das unos datos ("Calcula 5 + 3") y él te **entrega un papel** con el resultado ("8"), para que tú decidas qué hacer con él.

La Palabra Clave: 'return'

Para "entregar ese papel" y devolver el valor, las funciones usan la palabra clave `return`.

Sintaxis: ‘return‘

¿Cómo funciona ‘return‘?

Cuando Python llega a una línea con `return` dentro de una función, hace dos cosas:

1. **Termina la función inmediatamente** (ignora cualquier código que venga después).
2. **Envía de vuelta** el valor que está a la derecha de la palabra `return`.

```
# Definicion de una FUNCION
def sumar(a, b):
    resultado = a + b
    return resultado  # Devuelve el valor de 'resultado'

# Definicion de una SUBRUTINA (para comparar)
def sumar_e_imprimir(a, b):
    resultado = a + b
    print(f"El resultado es: {resultado}") # No devuelve nada

```

¡La Clave! Capturando el Valor de Retorno

¿Cómo usamos el valor devuelto?

El valor que `return` envía de vuelta puede ser "atrapado" en una variable usando el signo de asignación (=).

Código en Python

```
1  # Definimos la funcion
2  def sumar(a, b):
3      print("...calculando...")
4      return a + b
5      print("Esto NUNCA se ejecuta")
6
7  # --- Programa Principal ---
8  mi_suma = sumar(10, 5)
9  print(f"El valor guardado es: {mi_suma}")
10 print(f"El doble es: {mi_suma * 2}")
11
```

¡La Clave! Capturando el Valor de Retorno

Salida en Consola

```
...calculando...
El valor guardado es: 15
El doble es: 30
```

Análisis

- La línea `mi_suma = sumar(10, 5)` ejecuta la función.
- La función `return` 15.
- La línea original se convierte en `mi_suma = 15`.
- El `print` después del `return` fue ignorado.

Ejemplo 1: Función de Cálculo Simple

Objetivo

Escribe un programa para crear una función que reciba la base y la altura de un triángulo y **devuelva** su área.

Salida en Consola

```
El area del primer triangulo es: 25.0
El area del segundo triangulo es: 10.5
La suma de las areas es: 35.5
```

Ventaja

¡Podemos llamar a la función tantas veces como queramos con diferentes argumentos y reutilizar los resultados!

Ejemplo 1: Función de Cálculo Simple

Código en Python

```python
# 1. Definimos la funcion
def calcular_area_triangulo(base, altura):
    area = (base * altura) / 2
    return area

# --- Programa Principal ---

# 2. Llamamos a la funcion y guardamos resultados
area1 = calcular_area_triangulo(10, 5)
area2 = calcular_area_triangulo(7, 3)

# 3. Usamos los resultados
print(f"El area del primer triangulo es: {area1}")
print(f"El area del segundo triangulo es: {area2}")
print(f"La suma de las areas es: {area1 + area2}")

```

Ejemplo 2: Refactorizando Código (Avanzado)

Objetivo

Vamos a mejorar nuestro código de promedios de la clase pasada. Crearemos una función que **reciba una lista** y **devuelva su promedio** usando una función que llamaremos `calcular_promedio`.

Ejemplo 2: Refactorizando Código (Avanzado)

```python
# --- Definicion de la Funciona ---
def calcular_promedio(lista_de_notas):
    suma = 0
    for nota in lista_de_notas:
        suma += nota

    promedio = suma / len(lista_de_notas)
    return promedio # Devuelve el numero calculado

# --- Programa Principal (MAS LIMPIO) ---
calif_estudiante1 = [10, 9, 10]
calif_estudiante2 = [8, 7, 8]

# Usamos nuestra funcion
prom1 = calcular_promedio(calif_estudiante1)
prom2 = calcular_promedio(calif_estudiante2)

print(f"El promedio del Estudiante 1 es: {prom1:.2f}")
print(f"El promedio del Estudiante 2 es: {prom2:.2f}")
```

Resumen y Próxima Clase

Resumen: Funciones y ‘return‘

- **Función:** Es un bloque de código (definido con `def`) que **calcula** y **devuelve** un valor.
- **‘return‘:** Es la palabra clave que **finaliza** la función y **envía un valor de vuelta**.
- **Capturar el Valor:** Usamos una variable y el signo = para .ªtrapar.el valor que la función devuelve (ej. `mi_variable = mi_funcion()`).
- **Ventaja:** Esto nos permite usar el resultado en cálculos futuros, guardarlo, o pasarlo como argumento a *otra* función.