Guía de Laboratorio – Semana 4

Curso: Fundamentos de Algoritmos (COMP1201W)

Tema: Integración de Estructuras y Funciones Definidas por el Usuario

🧭 1. Propósito del Laboratorio

Fortalecer las habilidades de los estudiantes en la **modularización del código**, utilizando **funciones propias** para dividir problemas complejos en partes más simples, facilitando la lectura, el mantenimiento y la reutilización del código en **C#** y **Python**.

🎯 2. Objetivos Específicos

- Comprender la **definición y uso de funciones** en los lenguajes C# y Python.
- Aplicar **parámetros** y **valores de retorno** en funciones personalizadas.
- Emplear **variables locales y globales** dentro de una función.
- Desarrollar programas que integren **estructuras de control** y **funciones**.

⚙️ 3. Materiales y Herramientas

- Computadora con sistema operativo Windows o Linux.
- Visual Studio Code.
- Compilador **C# (dotnet SDK)** y **Python 3.x**.
- Cuenta de GitHub activa.

📓 4. Procedimiento del Laboratorio

Actividad 1: Función con retorno de valor

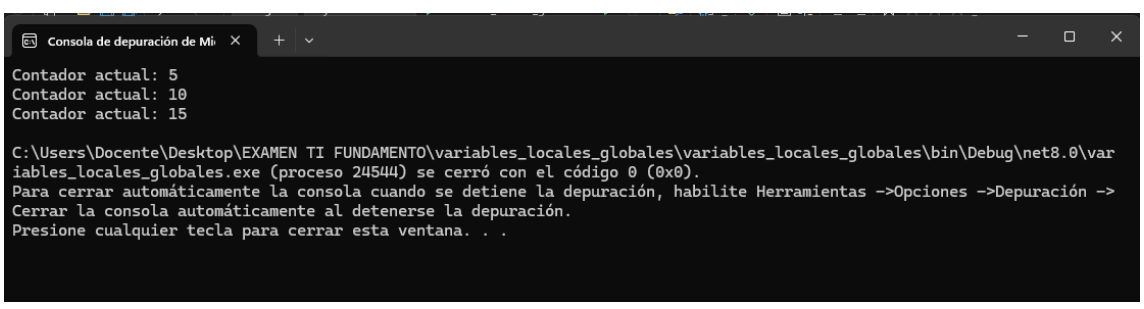
Crea una función que calcule el **área de un rectángulo** a partir de la base y la altura.

C#:

```
using System;
class Program {
    static double CalcularArea(double baseR, double altura) {
        return baseR * altura;
    }

    static void Main() {
        Console.Write("Ingrese la base: ");
        double b = double.Parse(Console.ReadLine());
        Console.Write("Ingrese la altura: ");
        double h = double.Parse(Console.ReadLine());

        double area = CalcularArea(b, h);
        Console.WriteLine($"El área del rectángulo es: {area}");
    }
}
```

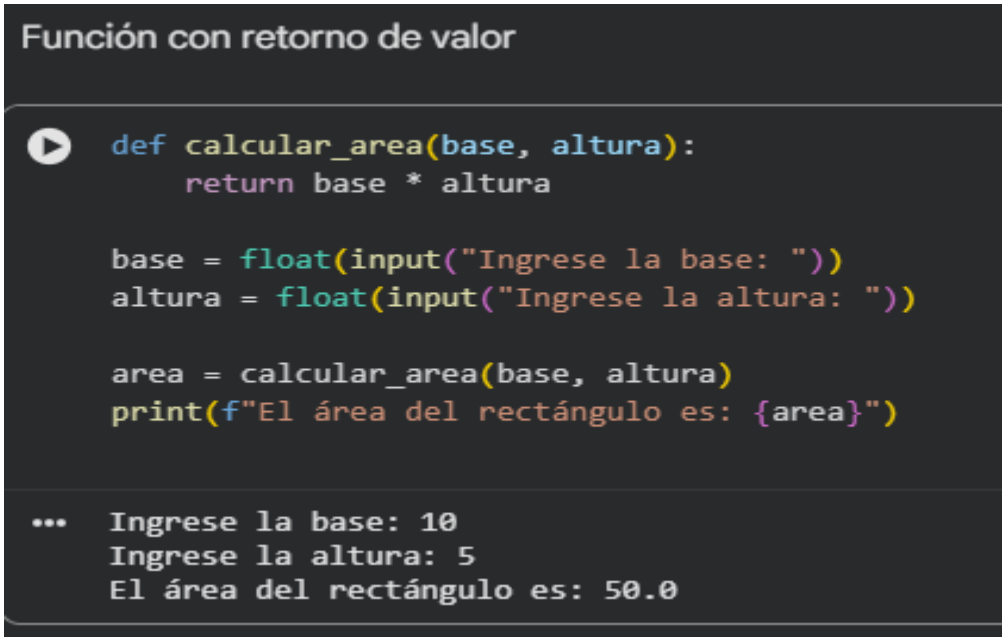


Python:

```
def calcular_area(base, altura):
    return base * altura

base = float(input("Ingrese la base: "))
altura = float(input("Ingrese la altura: "))

area = calcular_area(base, altura)
```



print(f"El área del rectángulo es: {area}")

Actividad 2: Uso de variables locales y globales

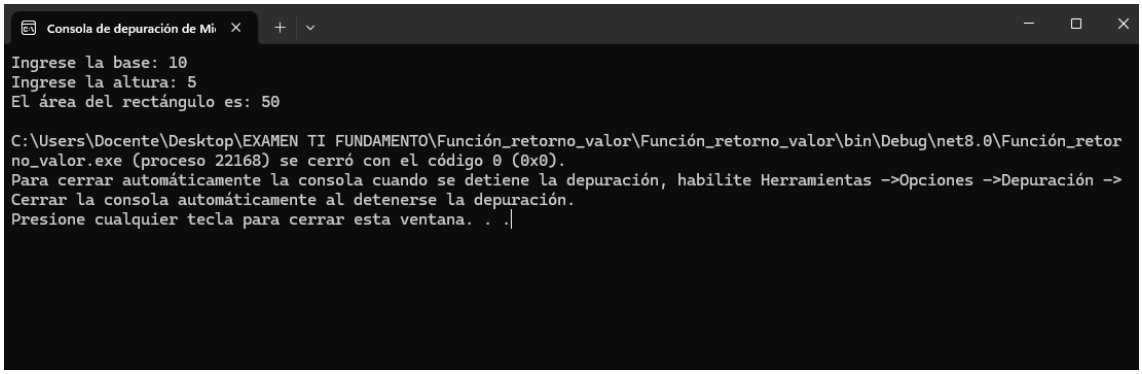
Diseña un programa que incremente un contador global cada vez que se llame a una función.

C#:

```
using System;
class Program {
    static int contador = 0; // variable global

    static void Incrementar() {
        int local = 5; // variable local
        contador += local;
        Console.WriteLine($"Contador actual: {contador}");
    }

    static void Main() {
        Incrementar();
        Incrementar();
        Incrementar();
    }
}
```



Python:

```
contador = 0  # variable global

def incrementar():
    global contador
    local = 5  # variable local
    contador += local
    print(f"Contador actual: {contador}")

incrementar()
incrementar()
incrementar()
```

Actividad 3: Integración de estructuras y funciones

Crea un programa que solicite el sueldo base y calcule el **bono por rendimiento**, usando una función con condicionales.

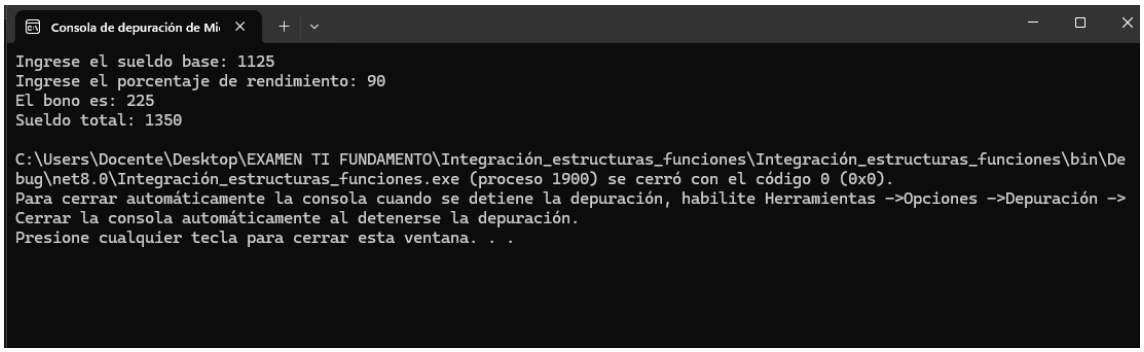
C#:

```
using System;
class Program {
    static double CalcularBono(double sueldo, int rendimiento) {
        if (rendimiento >= 90) return sueldo * 0.20;
        else if (rendimiento >= 75) return sueldo * 0.10;
```

```
        else return 0;
    }

    static void Main() {
        Console.Write("Ingrese el sueldo base: ");
        double sueldo = double.Parse(Console.ReadLine());
        Console.Write("Ingrese el porcentaje de rendimiento: ");
        int rendimiento = int.Parse(Console.ReadLine());

        double bono = CalcularBono(sueldo, rendimiento);
        Console.WriteLine($"El bono es: {bono}");
        Console.WriteLine($"Sueldo total: {sueldo + bono}");
    }
}
```

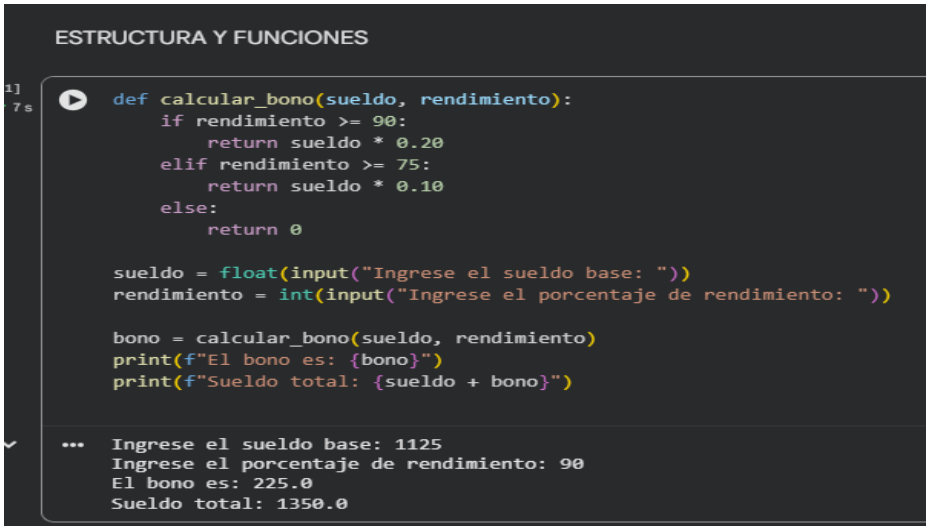


Python:

```
def calcular_bono(sueldo, rendimiento):
    if rendimiento >= 90:
        return sueldo * 0.20
    elif rendimiento >= 75:
        return sueldo * 0.10
    else:
        return 0

sueldo = float(input("Ingrese el sueldo base: "))
rendimiento = int(input("Ingrese el porcentaje de rendimiento: "))

bono = calcular_bono(sueldo, rendimiento)
print(f"El bono es: {bono}")
print(f"Sueldo total: {sueldo + bono}")
```



🧩 5. Evidencias a entregar

- Archivos: laboratorio4.cs y laboratorio4.py.
- Capturas de pantalla de la ejecución de cada programa.
- Enlace del repositorio GitHub con las tres actividades.

  https://github.com/FERMIN931/PRACTICA_DE_LAB_S4.git

- Informe en PDF con respuestas y conclusiones.

💬 6. Preguntas de Reflexión

1. ¿Qué ventajas ofrece modularizar el código mediante funciones?

   Facilita la lectura y el mantenimiento del código, dividiendo problemas complejos en partes más simples

2. ¿Qué diferencias existen entre variables globales y locales?

Las variable globales son accesibles y pueden ser modificadas desde cualquier parte del programa y las locales son accesibles dentro de esa función específica, de manera local.

3. ¿Por qué es importante el uso de parámetros en una función?

   Son fundamentales para la modularización del código, permitiendo que la función realice su tarea con datos específicos sin depender de variables externas.

4. ¿Cómo se puede mejorar la legibilidad y el mantenimiento del código usando funciones?

   El uso de funciones mejora la legibilidad y el mantenimiento de los códigos.

🧠 7. Conclusiones

- Las funciones permiten dividir un problema complejo en partes más pequeñas y manejables.
- El uso de variables locales y globales debe ser controlado para evitar errores de alcance.
- Modularizar el código mejora la reutilización, reduce errores y facilita el mantenimiento.