# Guía de Laboratorio – Sesión 08
Funciones y Modularidad en Python

Curso de Inteligencia Artificial

## Objetivo de la sesión

Al finalizar la sesión, el alumno será capaz de:

- Definir y utilizar funciones en Python.
- Implementar parámetros y valores de retorno.
- Organizar el código en módulos reutilizables.
- Aplicar buenas prácticas en la creación de funciones.

## Instrucciones

Resuelva los siguientes ejercicios en Python. Guarde cada uno en un archivo independiente. Cuando un ejercicio indique modularidad, cree un módulo `utilidades.py` y utilícelo en su script principal.

## Ejercicios

**Ejercicio 1:** **Saludo personalizado:** Cree una función que reciba un nombre y muestre un saludo.

**Ejercicio 2:** **Área de un círculo:** Defina una función que reciba el radio y devuelva el área. Use el módulo `math`.

**Ejercicio 3:** **Número par o impar:** Cree una función que determine si un número es par o impar.

**Ejercicio 4:** **Conversión de temperatura:** Escriba funciones para convertir de Celsius a Fahrenheit y viceversa.

**Ejercicio 5:** **Factorial con función:** Implemente el cálculo del factorial usando una función.

**Ejercicio 6:** **Módulo de utilidades matemáticas:** Cree un módulo `utilidades.py` con funciones para:

- Calcular el cuadrado de un número.
- Calcular el cubo de un número.

- Calcular la raíz cuadrada.

Importe y utilice el módulo en un script principal.

**Ejercicio 7:** **Calculadora modular:** Cree un módulo `calculadora.py` con funciones de suma, resta, multiplicación y división. Implemente un menú en un script principal para usarlas.

**Ejercicio 8:** **Contar vocales:** Defina una función que cuente el número de vocales en una cadena.

**Ejercicio 9:** **Máximo común divisor y mínimo común múltiplo:** Implemente ambas funciones y pruébelas con varios pares de números.

**Ejercicio 10:** **Agenda de contactos modular:** Cree un módulo con funciones para:

- Agregar un contacto (nombre y teléfono).
- Buscar un contacto por nombre.
- Mostrar todos los contactos.

Use un diccionario como estructura de almacenamiento.

## Recomendaciones

- Pruebe sus funciones con distintos valores de entrada.
- Comente el código para documentar el propósito de cada función.
- Mantenga el código limpio y organizado en módulos reutilizables.

## Ejercicios con soluciones

**Ejercicio 1:** **Saludo personalizado:**

```python
def saludar(nombre):
    print(f"Hola, {nombre}!")

saludar("Carlos")
```

**Ejercicio 2:** **Área de un círculo:**

```python
import math

def area_circulo(radio):
    return math.pi * radio**2

print(area_circulo(5))
```

**Ejercicio 3:** **Número par o impar:**

```python
def es_par(num):
    return num % 2 == 0

print(es_par(4))  # True
print(es_par(7))  # False
```

**Ejercicio 4: Conversión de temperatura:**

```python
def celsius_a_fahrenheit(c):
    return c * 9/5 + 32

def fahrenheit_a_celsius(f):
    return (f - 32) * 5/9

print(celsius_a_fahrenheit(0))
print(fahrenheit_a_celsius(32))
```

**Ejercicio 5: Factorial con función:**

```python
def factorial(n):
    resultado = 1
    for i in range(1, n+1):
        resultado *= i
    return resultado

print(factorial(5))
```

**Ejercicio 6: Módulo de utilidades matemáticas:** `utilidades.py`:

```python
import math

def cuadrado(n):
    return n ** 2

def cubo(n):
    return n ** 3

def raiz_cuadrada(n):
    return math.sqrt(n)
```

Script principal:

```python
import utilidades

print(utilidades.cuadrado(4))
print(utilidades.cubo(3))
print(utilidades.raiz_cuadrada(16))
```

**Ejercicio 7: Calculadora modular:** `calculadora.py`:

```python
def sumar(a,b): return a+b
def restar(a,b): return a-b
def multiplicar(a,b): return a*b
def dividir(a,b): return a/b if b != 0 else "Error: división por cero"
```

Script principal:

```python
import calculadora

print(calculadora.sumar(3,4))
print(calculadora.dividir(10,2))
```

**Ejercicio 8: Contar vocales:**

```python
def contar_vocales(texto):
    vocales = "aeiouAEIOU"
    return sum(1 for letra in texto if letra in vocales)

print(contar_vocales("Inteligencia Artificial"))
```

**Ejercicio 9: MCD y MCM:**

```python
import math

def mcd(a,b):
    return math.gcd(a,b)

def mcm(a,b):
    return abs(a*b) // math.gcd(a,b)

print(mcd(12,18))
print(mcm(12,18))
```

**Ejercicio 10: Agenda de contactos modular:** agenda.py:

```python
contactos = {}

def agregar_contacto(nombre, telefono):
    contactos[nombre] = telefono

def buscar_contacto(nombre):
    return contactos.get(nombre, "No encontrado")

def mostrar_contactos():
    return contactos
```

Script principal:

```python
import agenda

agenda.agregar_contacto("Juan", "12345")
print(agenda.buscar_contacto("Juan"))
print(agenda.mostrar_contactos())
```