

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:

```
1 def saludar(nombre):  
2     print(f"Hola, {nombre}!")  
3  
4 saludar("Carlos")
```

Ejercicio 2: Área de un círculo:

```
1 import math  
2  
3 def area_circulo(radio):  
4     return math.pi * radio**2  
5  
6 print(area_circulo(5))
```

Ejercicio 3: Número par o impar:

```

1 def es_par(num):
2     return num % 2 == 0
3
4 print(es_par(4))    # True
5 print(es_par(7))    # False

```

Ejercicio 4: Conversión de temperatura:

```

1 def celsius_a_fahrenheit(c):
2     return c * 9/5 + 32
3
4 def fahrenheit_a_celsius(f):
5     return (f - 32) * 5/9
6
7 print(celsius_a_fahrenheit(0))
8 print(fahrenheit_a_celsius(32))

```

Ejercicio 5: Factorial con función:

```

1 def factorial(n):
2     resultado = 1
3     for i in range(1, n+1):
4         resultado *= i
5     return resultado
6
7 print(factorial(5))

```

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

```

1 import math
2
3 def cuadrado(n):
4     return n ** 2
5
6 def cubo(n):
7     return n ** 3
8
9 def raiz_cuadrada(n):
10    return math.sqrt(n)

```

Script principal:

```

1 import utilidades
2
3 print(utilidades.cuadrado(4))
4 print(utilidades.cubo(3))
5 print(utilidades.raiz_cuadrada(16))

```

Ejercicio 7: Calculadora modular: calculadora.py:

```

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

```

Script principal:

```

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

```

Ejercicio 8: Contar vocales:

```

1 def contar_vocales(texto):
2     vocales = "aeiouAEIOU"
3     return sum(1 for letra in texto if letra in vocales)
4
5 print(contar_vocales("Inteligencia Artificial"))

```

Ejercicio 9: MCD y MCM:

```

1 import math
2
3 def mcd(a,b):
4     return math.gcd(a,b)
5
6 def mcm(a,b):
7     return abs(a*b) // math.gcd(a,b)
8
9 print(mcd(12,18))
10 print(mcm(12,18))

```

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

```

1 contactos = {}
2
3 def agregar_contacto(nombre, telefono):
4     contactos[nombre] = telefono
5
6 def buscar_contacto(nombre):
7     return contactos.get(nombre, "No encontrado")
8
9 def mostrar_contactos():
10    return contactos

```

Script principal:

```

1 import agenda
2
3 agenda.agregar_contacto("Juan", "12345")
4 print(agenda.buscar_contacto("Juan"))
5 print(agenda.mostrar_contactos())

```