Fundamentos de programación

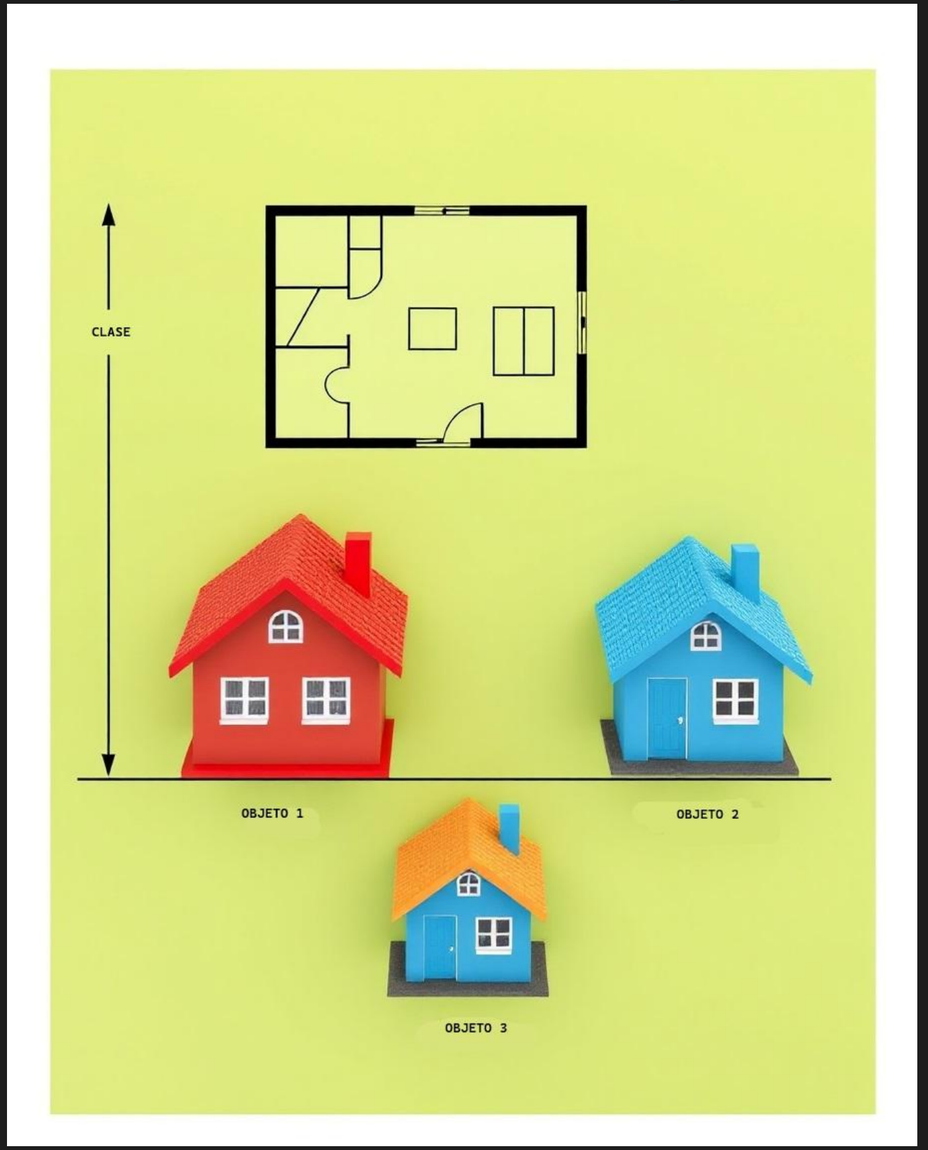
Python

Para el análisis de datos

Sesión 6

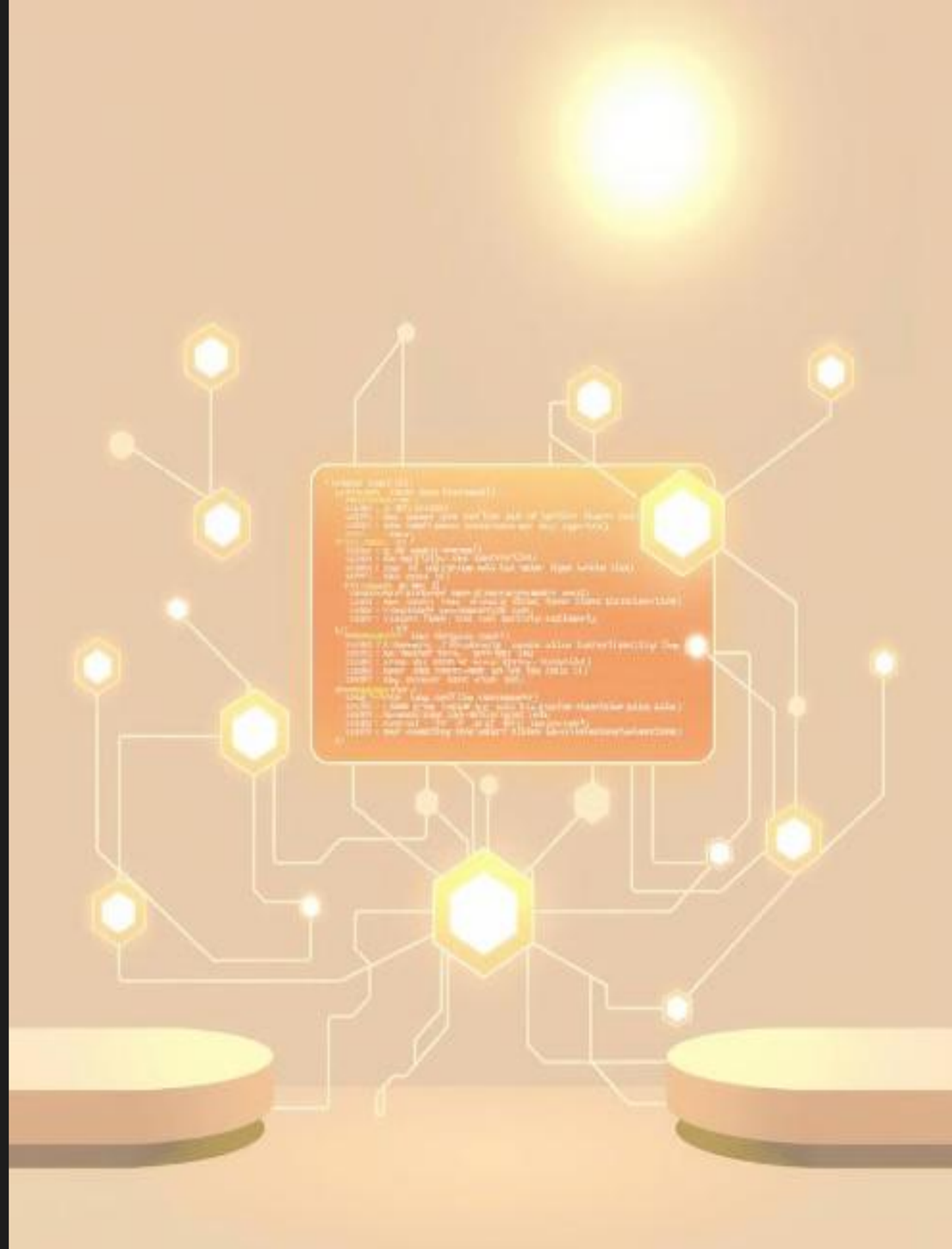
¿Qué es la Programación Orientada a Objetos?

La Programación Orientada a Objetos (POO) es un paradigma que organiza el código en torno a “objetos”, que representan entidades del mundo real con atributos (datos) y comportamientos (métodos o funciones). El objetivo de la POO es simplificar el diseño y hacer que el código sea reutilizable, modular y más fácil de mantener.



Principios básicos de la POO

1. Abstracción
   Este principio permite enfocarse en los detalles relevantes de un objeto, ocultando los detalles internos y exponiendo lo esencial.
2. Encapsulación
   Este principio implica agrupar datos y métodos que manipulan esos datos en una sola unidad (una clase) y controlar su acceso.



Principios básicos de la POO

Abstracción y Encapsulación

```python
# Ejemplo de encapsulación
class CuentaBancaria:
    def __init__(self, titular, saldo=0):
        self.titular = titular
        self.__saldo = saldo  # Atributo privado

    def depositar(self, monto):
        if monto > 0:
            self.__saldo += monto

    def consultar_saldo(self):
        return self.__saldo

cuenta = CuentaBancaria("Juan", 100)
cuenta.depositar(50)
print(cuenta.consultar_saldo())  # Salida: 150
```

Clases y objetos

Clases

En Python, una clase es un modelo o plantilla que define los atributos y comportamientos de los objetos.

Objetos

Un objeto es una instancia de una clase, con sus propios valores de atributos.

Comportamiento y Métodos

Los métodos son funciones definidas dentro de una clase que describen los comportamientos de los objetos. Estos permiten que el objeto realice acciones o interacciones.

Constructores e Instanciación

Constructor

Un constructor es un método especial llamado __init__ en Python. Este método se ejecuta automáticamente cuando se crea un nuevo objeto con la clase y se utiliza para inicializar los atributos del objeto.

Instanciación

Es el proceso de crear un objeto a partir de una clase. Esto se hace llamando a la clase como si fuera una función, lo que invoca el constructor __init__.

Ejemplo de Clase con Constructor e Instanciación

```python
# Ejemplo de clase con constructor e instanciación de objetos
class Persona:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

    def saludar(self):
        print(f"Hola, mi nombre es {self.nombre} y tengo {self.edad} años.")

# Instanciación de objetos
persona1 = Persona("Ana", 25)
persona2 = Persona("Luis", 30)

persona1.saludar()  # Salida: Hola, mi nombre es Ana y tengo 25 años.
persona2.saludar()  # Salida: Hola, mi nombre es Luis y tengo 30 años.
```

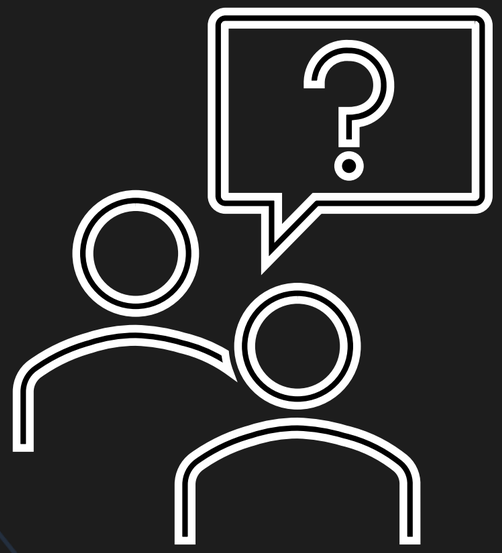
El Objeto String

upper()	Convierte la cadena a mayúsculas.
lower()	Convierte la cadena a minúsculas.
find()	Busca una subcadena y devuelve su índice si la encuentra.
replace()	Reemplaza una subcadena con otra.
split()	Divide la cadena en una lista de palabras.

```python
texto = "Hola Mundo"
print(texto.upper())          # Salida: HOLA MUNDO
print(texto.lower())          # Salida: hola mundo
print(texto.find("Mundo"))    # Salida: 5
print(texto.replace("Mundo", "Python"))  # Salida: Hola Python
palabras = texto.split()      # Salida: ['Hola', 'Mundo']
```

Preguntas

Sección de preguntas



Fundamentos de programación

Python

Para el análisis de datos

Continúe con las actividades