SESIÓN PROGRAMACIÓN ORIENTADA A OBJETOS POO

CONTENIDOS:

- ¿En qué consiste el paradigma de orientación a objetos en la programación?
- Principios básicos de la programación a objetos: abstracción y encapsulación.
- Clases y objeto, comportamiento y métodos, constructores e instanciación.
- El objeto String y sus métodos principales.

¿EN QUÉ CONSISTE EL PARADIGMA DE ORIENTACIÓN A OBJETOS EN LA PROGRAMACIÓN?

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. Python es un lenguaje que soporta POO, permitiendo la creación y manipulación de clases y objetos de manera intuitiva.

En POO, trabajamos con clases como plantillas para crear objetos, y cada objeto es una instancia de una clase específica.

```python
# Ejemplo básico de clase y objeto
class Perro:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

    def ladrar(self):
        print(f"{self.nombre} está ladrando.")

# Creación de un objeto de la clase Perro
mi_perro = Perro("Fido", 3)
mi_perro.ladrar()  # Salida: Fido está ladrando.
```

Ilustración 1 Ejemplo clase y objeto

PRINCIPIOS BÁSICOS DE LA PROGRAMACIÓN A OBJETOS: ABSTRACCIÓN Y ENCAPSULACIÓN

- **Abstracción:** Este principio permite enfocarse en los detalles relevantes de un objeto, ocultando los detalles internos y exponiendo solo lo esencial. En Python, creamos abstracciones al definir clases y métodos que representan comportamientos específicos y simplifican el acceso a la funcionalidad necesaria sin exponer la implementación.
- **Encapsulación:** Este principio implica agrupar datos y métodos que manipulan esos datos en una sola unidad (una clase) y controlar su acceso. En Python, el encapsulamiento se logra utilizando convenciones de nombres: el guion bajo _nombre o __nombre indica que un atributo o método es privado y no debe ser accedido directamente desde fuera de la clase.

```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
```

Ilustración 2 Encapsulación

CLASES Y OBJETOS, COMPORTAMIENTO Y MÉTODOS, CONSTRUCTORES E INSTANCIACIÓN

- **Clases y objetos:** En Python, una clase es un modelo o plantilla que define los atributos y comportamientos de los 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:** Un constructor es un método especial llamado __init__ en Python. Este método se ejecuta automáticamente cuando se crea un nuevo objeto de 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__.

```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.
```

Ilustración 3 Clase con constructor e instancia de objetos

EL OBJETO STRING Y SUS MÉTODOS PRINCIPALES

En Python, las cadenas de caracteres (Strings) son objetos de la clase **str** que poseen diversos métodos incorporados que facilitan la manipulación de texto. Estos métodos permiten, por ejemplo, convertir el texto a mayúsculas, buscar elementos, y dividir el texto, entre otros.

Algunos de los métodos principales del objeto String son:

- **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.

```
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']
```

Ilustración 4 Métodos del objeto String

En síntesis, la Programación Orientada a Objetos (POO) es un paradigma que organiza el código en torno a clases y objetos, y constituye un enfoque fundamental en la programación moderna. La abstracción y encapsulamiento, como pilares de la POO, ayudan a gestionar la complejidad de los programas, permitiendo modelar el mundo real en estructuras de datos claras y proteger el acceso a la información.

Con las clases como plantillas que definen atributos y métodos, y el uso de constructores que permiten crear instancias con valores específicos, la POO facilita la reutilización y expansión del código.