Actividad 3

Atributos y Métodos

Explicación de los atributos y su papel en la representación de datos en una clase

Los atributos en una clase de programación orientada a objetos son variables que almacenan datos asociados a un objeto especí co creado a partir de esa clase. Cada objeto tiene su propio conjunto de valores de atributos, lo que le da su identidad y de ne su estado. El papel de los atributos y su importancia en la representación de datos en una clase es.

De nición de Atributos:

Los atributos son variables que se de nen dentro de una clase y representan las características o propiedades de los objetos creados a partir de esa clase.
Cada atributo tiene un nombre único dentro de la clase y puede tener un tipo de dato especí co, como entero, cadena, otante, etc.

Explicación de los atributos y su papel en la representación de datos en una clase

Características de los Atributos:

Unicidad: **Cada atributo tiene un nombre único que lo identi ca dentro de la clase.**

Visibilidad: Puede haber atributos públicos, privados o protegidos, según las convenciones de nomenclatura y el uso de guiones bajos o doble guión bajo en el nombre. Roles de los Atributos:

Representación del Estado: **Los atributos representan el estado interno de un objeto. Por ejemplo, en una clase "Coche", los atributos pueden incluir "color", "modelo", "velocidad", etc.**

Almacenamiento de Datos: **Los atributos almacenan datos especí cos para cada objeto creado a partir de la clase. Cada objeto tiene su propio conjunto de valores para estos atributos.**

Explicación de los atributos y su papel en la representación de datos en una clase

Ejemplo Práctico:

Considera una clase "Perro" con atributos como "nombre" y "edad". Cada objeto creado a partir de esta clase representará un perro especí co con su propio nombre y edad.

En este ejemplo, "nombre" y "edad" son atributos de la clase "Perro". Cada objeto (perro1 y perro2) tiene su propio conjunto de valores para estos atributos, lo que permite representar de manera única a cada perro.

```python
class Perro:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

# Creación de objetos de tipo Perro
perro1 = Perro("Buddy", 3)
perro2 = Perro("Max", 5)

# Acceso a los atributos
print(f"{perro1.nombre} tiene {perro1.edad} años.")
print(f"{perro2.nombre} tiene {perro2.edad} años.")
```

Explicación de los atributos y su papel en la representación de datos en una clase

Importancia en la Representación de Datos: **Los atributos son fundamentales para representar datos y características especí cas de los objetos en la programación orientada a objetos.**

Permiten modelar entidades del mundo real de manera más el, ya que los objetos pueden re ejar propiedades únicas a través de sus atributos.

Facilitan la encapsulación al agrupar datos y métodos relacionados en una unidad cohesiva, mejorando la organización y la modularidad del código.

Los atributos en una clase juegan un papel esencial al representar datos especí cos para cada objeto creado a partir de esa clase, contribuyendo a la modelación y representación el de entidades del mundo real en la programación orientada a objetos.

Definición de métodos y su relación con las funciones en la POO

En la Programación Orientada a Objetos, los métodos son funciones que están asociadas a objetos especí cos y se de nen dentro de las clases. La relación entre métodos y funciones es que ambos representan bloques de código que realizan tareas especí cas, pero mientras que las funciones son independientes, los métodos están vinculados a objetos y actúan sobre ellos. una de nición más detallada de los métodos y su relación con las funciones en la POO.

De nición de Métodos: **Los métodos son funciones que están de nidas dentro de una clase y están asociadas a los objetos creados a partir de esa clase. Los métodos tienen acceso a los datos (atributos) de la instancia particular de la clase y pueden manipular esos datos según sea necesario.**

Definición de métodos y su relación con las funciones en la POO

Características de los Métodos:

Pertenencia a una Clase: Cada método está asociado a una clase especí ca y es invocado en instancias de esa clase.

Acceso a Atributos: Los métodos pueden acceder y manipular los atributos de la instancia de la clase a la que pertenecen.

Ejemplo Práctico: **Considera una clase "Coche" con un método llamado "acelerar". Este método puede modi car el atributo "velocidad" del coche.**

+ info

```python
class Coche:
    def __init__(self, marca, modelo, velocidad):
        self.marca = marca
        self.modelo = modelo
        self.velocidad = velocidad

    def acelerar(self, aumento):
        self.velocidad += aumento
        print(f"El coche {self.marca} {self.modelo} aceleró. Nueva velocidad: {self.velocidad} km/h.")

# Creación de un objeto de tipo Coche
mi_coche = Coche("Toyota", "Camry", 50)

# Llamada al método acelerar
mi_coche.acelerar(20)
```

En este ejemplo, "acelerar" es un método de la clase "Coche". Cuando se llama a este método en la instancia mi_coche, afecta directamente al atributo velocidad de ese objeto especí co.

Definición de métodos y su relación con las funciones en la POO

Relación con Funciones: **Mientras que las funciones son bloques de código independientes que realizan tareas especí cas, los métodos están vinculados a objetos y forman parte integral de la programación orientada a objetos.**

Las funciones pueden llamarse desde cualquier parte del código, mientras que los métodos son invocados en instancias particulares de una clase.

Importancia en la POO: 1 2 3 4 5

Pueden tener acceso a los atributos de la clase mediante self.

```
class Rectangulo:
    def __init__(self, base, altura):
        self.base = base
        self.altura = altura

    def calcular_area(self):
        return self.base * self.altura

rectangulo = Rectangulo(5, 10)
print(rectangulo.calcular_area())
```

Los métodos son funciones asociadas a objetos en la POO y juegan un papel crucial al permitir que los objetos realicen acciones especí cas y manipulen sus propios datos. Su integración con la encapsulación y la modularidad es fundamental para el diseño efectivo de programas orientados a objetos.

Mejoran la modularidad del código al organizar la funcionalidad en bloques lógicos y reutilizables.

Facilitan la encapsulación al agrupar el comportamiento (código) y los datos (atributos) relacionados en una unidad cohesiva (el objeto).

Los métodos son esenciales en la POO ya que permiten que los objetos realicen acciones especí cas y respondan a solicitudes o eventos.

Definición de métodos y su relación con las funciones en la POO

Ejercicio : Métodos de la Clase

De ne una clase llamada Persona con atributos nombre y edad.
Crea dos objetos de tipo Persona e imprime sus atributos.
Modi ca la clase Persona del ejercicio anterior.
Agrega un método llamado saludar que imprima un saludo con el nombre de la persona.
Llama al método saludar para ambos objetos creados. Solucion:

```python
class Persona:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

# Crea dos objetos de tipo Persona
persona1 = Persona("Juan", 30)
persona2 = Persona("María", 25)

# Imprime los atributos de las personas
print(f"{persona1.nombre} tiene {persona1.edad} años.")
print(f"{persona2.nombre} tiene {persona2.edad} años.")
```

Definición de métodos y su relación con las funciones en la POO

Ejercicio : Actualización de Atributos **Modi ca la clase Persona del ejercicio anterior. Agrega un método llamado envejecer que tome un parámetro años y actualice la edad de la persona. Llama al método envejecer para uno de los objetos e imprime la nueva edad.**

```python
class Persona:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

    def envejecer(self, anos):
        self.edad += anos

# Crea un objeto de tipo Persona
persona1 = Persona("Juan", 30)

# Llama al método envejecer e imprime la nueva edad
persona1.envejecer(5)
print(f"Después de envejecer, {persona1.nombre} tiene{persona1.edad} años.")
```

+ info

Estos ejercicios te permitirán practicar la de nición de clases, la creación de objetos y la implementación de métodos en Python. Puedes ajustar la complejidad según el nivel de los participantes.

TIC

TALENTO TECH

Universidad Tecnológica de Pereira

faceIT