Actividad Autónoma - Programación II

Unidad 2: Clasificación del Data set Iris utilizando Programación Orientada a Objetos (POO) y Python

Tema 2: Herencia y Abstracción

🎯 Objetivo

Aplicar los conceptos de herencia, polimorfismo, métodos mágicos y clases abstractas aprendidos en las semanas 3 y 4, mediante la implementación de un proyecto que utiliza los principios de la Programación Orientada a Objetos para resolver un problema de clasificación con el data set Iris, enfocándose en la modularidad, extensibilidad y reutilización de código.

🛠️ Actividad planteada

1. Definir una clase Flor que representa las características principales de una flor del conjunto de datos Iris.
2. Implementar una clase abstracta Clasificador con un método abstracto clasificar ().
3. Cree una subclase ClasificadorIris que herede de Clasificador y que implemente el método clasificar () para clasificar flores según la longitud del pétalo.
4. Agregue un método calcular_distancia() a la clase Flor para calcular la distancia euclidiana entre dos instancias de Flor.
5. Utilizar este método para comparar dos flores y mostrar cuál de ellas está más cerca de una flor de referencia.
6. Sobrecargar el operador + para combinar dos flores y obtener una nueva flor con los atributos promedio.
7. Crear al menos dos objetos Flor y utilizar la sobrecarga del operador + para obtener una nueva flor promedio.
8. Agregue un método mágico **mul** a la clase Flor que permita multiplicar los valores numéricos de una flor por un escalar (número). Implementar un ejercicio que muestre cómo utilizar este método mágico para incrementar en un cierto porcentaje el tamaño de los sépalos y los pétalos de una flor.

🧠 Calculo de Distancias Adicionales

Modificar la clase Flor para incluir métodos que permitan calcular diferentes tipos de distancias que admiten los cuatro atributos (longitud_sepalo, ancho_sepalo, longitud_petalo, ancho_petalo). Además de la distancia euclidiana, implemente al menos otros dos tipos de distancias como la distancia de Manhattan y la distancia de Minkowski.

En [ ]:
```
de abc importar ABC , método abstracto importar matemáticas


# Clase flor, representa una flor del dataset Iris
class Flor : def __init__ ( self , longitud_sepalo , ancho_sepalo , longitud_pet









    # USAMOS GETTER PARA ACCEDER A LOS ATRIBUTOS
    @property
    def longitud_sepalo ( self ): return self . __longitud_sepalo


    @property
    def ancho_sepalo ( self ): devuelve self . __ancho_sepalo


    @property
    def longitud_petalo ( self ): devuelve self . __longitud_pétalo


    @property
    def ancho_petalo ( self ): devuelve self . __ancho_petalo


    @property
    def especie ( self ): return self .__ especie



    # USAMOS SETTER PARA PODER MODIFICAR LOS ATRIBUTOS
    @longitud_sepalo . setter
    def longitud_sepalo ( self , valor ): self . __longitud_sepalo = valor


    @ancho_sepalo . setter
    def ancho_sepalo ( self , valor ): self . __ancho_sepalo = valor


    @longitud_petalo . setter
    def longitud_petalo ( self , valor ): self . __longitud_petalo = valor


    @ancho_petalo . setter
    def ancho_petalo ( self , valor ): self . __ancho_petalo = valor


    @especie . setter
    def especie ( self , valor ): self . __especie = valor


    # STR NOS PERMITE VER LOS DETALLES DE LA FLOR EN UN FORMATO LEGIBLE
    def __str__ ( self ): return ( f " La Flor { self . __especie } : \n " f "SE
```

```
    # Nos va a permitir calcular la distacia euclidiana entre 2 flores
    def distancia ( self , segunda_flor ): return math . sqrt ( ( self . __longi









    # Nos permite calcular la distancia Mnhattan
    def dist_manhatan ( self , otra ): return abs ( self . __longitud_sepalo - o
                abs ( self . __ancho_sepalo - otra . __ancho_sepalo ) + \
                abs ( self . __longitud_petalo - otra . __longitud_petalo ) + \
                abs ( self . __ancho_petalo - otra __ancho_petalo )


    #Nos permite calcular las distacia MINKOWSKI
    def distan_min ( self , otra , p ): return ( abs ( self . __longitud_sepalo 









     # SObrecarga el operador + : Combina 2 flores y obtiene una nueva flor
    def __add__(self, nueva_flor):
        nuevo_long_sepalo = (self.__longitud_sepalo + nueva_flor.__longitud_sepa
        nuevo_ancho_sepalo = (self.__ancho_sepalo + nueva_flor.__ancho_sepalo)/2
        nuevo_long_petalo = (self.__longitud_petalo + nueva_flor.__longitud_peta
        nuevo_ancho_petalo = (self.__ancho_petalo + nueva_flor.__ancho_petalo )/
        return Flor(nuevo_long_sepalo, nuevo_ancho_sepalo, nuevo_long_petalo, nu

    # Nos permite multiplicar los valores numerucos de una flor por un escalae
    def __mul__(self, escala):
        return Flor(
            self.__longitud_sepalo * escala,
            self.__ancho_sepalo * escala,
            self.__longitud_petalo * escala,
            self.__ancho_petalo * escala,
            self.__especie
        )

# Implementacion de una clase abstracta
class Clasificador(ABC):
    @abstractmethod
    def clasificar (self, flor):
        pass

# Subclases heredadas del clasificador (implementa el metodo clasificar flores s
class ClassIiris(Clasificador):
    def clasificar(self, flor):
        if flor.longitud_petalo < 2.5 :
            return " SETOSA "
        elif 2.5 <= flor.longitud_petalo < 5 :
            return " VERSICOLOR "
        else :
```

```
            return " VIRGINICA "

# nos permite comprarar dos flores con respecto a una flor de referencia
def compa_distan(flor_re, flor_1, flor_2):
    dis_1 = flor_re.distancia(flor_1)
    dis_2 = flor_re.distancia(flor_2)
    print(f"Distania a la Flor 1: {dis_1: .2f}")
    print(f"Distancia a la Flor 2: {dis_2: .2f}")
    if dis_1 < dis_2 :
        print ("La Flor 1 esta mas cerca de la flor referencial")
    else:
        print("La flor 2 esta mas cerca de la flor referencial")


# EJEMPLO PARA PRUEBA DEL CODIGO
# Primero se creara 3 flores
flor1= Flor(4.2, 3.0, 2.5, 0.1, "Setosa")
flor2 = Flor(5.1, 3.5, 1.4, 0.2, "Versicolor")
flor3 = Flor(6.2, 2.2, 6.0, 1.5, "Virginica")

#Nos muestra sus detalles
print(flor1)
print(flor2)

#Nos modifica sus atributos mediante el Setter
flor1.ancho_petalo= 0.3
print("Flor 1 modificada")
print(flor1)

# Va a clasificar las flores
clasificador = ClassIiris()
print("Clasificaion de flores: ")
print(f"Flor 1: {clasificador.clasificar(flor1)}")
print(f"Flor 2: {clasificador.clasificar(flor2)}")
print(f"Flor 3: {clasificador.clasificar(flor3)}")

# Ahora va,os a comparar las distancias con nuestra flor de referencia
print("Comparacion de distancias: ")
compa_distan(flor1, flor2, flor3)

# Combinar las flores
flor_promedio = flor1 + flor2
print("El promedio entre la flor 1 y 2 es : ")
print(flor_promedio)

# Aumenta el tamaño
flor_escala = flor1 * 1.5
print("La flor 1 fue escalada al 150 %: ")
print(flor_escala)

# Calculo final de distancia entre flores
print("Calculo de la distancia entre Flor 1 y Flor 2: ")
print(f"Eucliciana: {flor1.distancia(flor2): .2f}")
print(f"Manhattan: {flor1.dist_manhatan(flor2): .2f}")
print(f"Minkowski (p=3): {flor1.distan_min(flor2, 3): .2f}")
```

In [ ]: