1. **Repaso de Objetos**

from abc import ABC, abstractmethod

class Vehiculo(ABC):

    def \_\_init\_\_(self, marca, modelo, color):

        self.marca = marca

        self.modelo = modelo

        self.color = color

        self.encendido = False

    def encender(self):

        self.encendido = True

        print("El vehículo está encendido.")

    def apagar(self):

        self.encendido = False

        print("El vehículo está apagado.")

    @abstractmethod

    def conducir(self):

        pass

class Automovil(Vehiculo):

    def \_\_init\_\_(self, marca, modelo, color, tipo\_combustible):

        super().\_\_init\_\_(marca, modelo, color)

        self.tipo\_combustible = tipo\_combustible

    def conducir(self):

        if self.encendido:

            print(

                f"Conduciendo el {self.marca} {self.modelo} - {self.tipo\_combustible}."

            )

        else:

            print("No se puede conducir, el vehículo está apagado.")

class Camion(Vehiculo):

    def \_\_init\_\_(self, marca, modelo, color, capacidad\_carga):

        super().\_\_init\_\_(marca, modelo, color)

        self.capacidad\_carga = capacidad\_carga

    def conducir(self):

        if self.encendido:

            print(

                f"Conduciendo el {self.marca} {self.modelo} - Capacidad de carga: {self.capacidad\_carga}."

            )

        else:

            print("No se puede conducir, el vehículo está apagado.")

# Crear objetos de las clases derivadas

automovil1 = Automovil("Toyota", "Corolla", "Rojo", "Gasolina")

camion1 = Camion("Ford", "F-150", "Azul", "2 toneladas")

# Encender y conducir un automóvil

automovil1.encender()

automovil1.conducir()

# Apagar el automóvil

automovil1.apagar()

* 1. **Encapsulamiento**

# Atributos privados

class Ejemplo:

    def \_\_init\_\_(self):

        self.\_\_atributo\_privado = 10

    def get\_atributo(self):

        print(f"{ self.\_\_atributo\_privado\*2}")

valor = Ejemplo()

valor.get\_atributo()

valor.\_\_atributo\_privado = 20

# Atributos protegidos

class Ejemplo:

    def \_\_init\_\_(self):

        self.\_atributo\_protegido = 20

    def get\_atributo(self):

        print(self.\_atributo\_protegido)

valor = Ejemplo()

print(valor.\_atributo\_protegido)  # no es correcto / se puede lograr

valor.get\_atributo()

class CuentaBancaria:

    def \_\_init\_\_(self, numero\_cuenta, saldo):

        self.\_numero\_cuenta = numero\_cuenta  # atributo protegido

        self.\_\_saldo = saldo  # atributo privado

    def get\_saldo(self):

        return self.\_\_saldo

    def depositar(self, monto):

        self.\_\_saldo += monto

    def retirar(self, monto):

        if self.\_\_saldo >= monto:

            self.\_\_saldo -= monto

        else:

            print("Saldo insuficiente")

# Modo de uso

cuenta = CuentaBancaria("12345", 1000)

print(cuenta.\_numero\_cuenta)

# No es recomedable

# print(cuenta.\_CuentaBancaria\_\_saldo)

# Correcto -> getter

valor = cuenta.get\_saldo()

# utilizar metodos publicos

cuenta.depositar(500)

cuenta.retirar(200)

# print(cuenta.get\_saldo())

class Ejemplo:

    def \_\_init\_\_(self):

        self.\_\_atributo\_privado = 30

    def getter(self):

        return self.\_\_atributo\_privado

    def setter(self, valor):

        if valor > 0:

            self.\_\_atributo\_privado = valor

        else:

            print("El valor debe ser positivo")

ejemplo = Ejemplo()

print(ejemplo.getter())

ejemplo.setter(2)

print(ejemplo.getter())

ejemplo.\_\_atributo\_privado = 100

print(ejemplo.getter())

* 1. **Abstracción**

class vehiculo:

    def \_\_init\_\_(self, marca, modelo):

        self.marca = marca

        self.modelo = modelo

        self.encendido = False

        self.velocidad = 0

    def mostrar\_vehiculo(self):

        print(f"Este es un {self.marca}  {self.modelo}")

    def encender(self):

        self.encendido = True

        print("El vehiculo esta encendido")

    def apagar(self):

        self.encendido = False

        print("El vehiculo esta apagado")

    def acelerar(self, incremento):

        if self.encendido:

            self.velocidad += incremento

            print(f"Vehiculo acelerado: {self.velocidad} km/h")

        else:

            print(f"No se puede acelerar, encienda el vehiculo")

    def frenar(self, decremento):

        if self.encendido:

            if self.velocidad - decremento >= 0:

                self.velocidad -= decremento

                print(f"El vehiculo desacelero a: {self.velocidad} km/h")

            else:

                self.velocidad = 0

                print(f"El vehiculo se detuvo")

        else:

            print(f"No se puede frenar, encienda el vehiculo")

vehiculo1 = vehiculo("Toyota", "Corolla")

vehiculo1.mostrar\_vehiculo()

vehiculo1.encender()

vehiculo1.acelerar(10)

vehiculo1.frenar(10)

vehiculo1.apagar()

* 1. **Herencia**

class Vehiculo:

    def \_\_init\_\_(self, marca, modelo):

        self.marca = marca

        self.modelo = modelo

    def describir(self):

        return f"Este vehiculo es un {self.marca} {self.modelo}"

class Automovil(Vehiculo):

    def \_\_init\_\_(self, marca, modelo, puertas):

        super().\_\_init\_\_(marca, modelo)

        self.puertas = puertas

    def describir(self):

        return f"Este automovil es un {self.marca} {self.modelo} con { self.puertas} puertas"

class Camion(Automovil):

    def \_\_init\_\_(self, marca, modelo, puertas, carga):

        super().\_\_init\_\_(marca, modelo, puertas)

        self.carga = carga

    def describir(self):

        return f"Este camion es un {self.marca} {self.modelo} con { self.puertas} puertas y una capacidad de carga {self.carga} kg"

vehiculo1 = Automovil("Toyota", "Corolla", 4)

vehiculo2 = Camion("Toyota", "FSHD", 2, 1000)

print(vehiculo1.describir())

print(vehiculo2.describir())

* 1. **Polimorfismo**

class Figura:

    def area(self):

        pass

class Circulo(Figura):

    def \_\_init\_\_(self, radio):

        self.radio = radio

    def area(self):

        return 3.14 \* self.radio\*\*2

class Cuadrado(Figura):

    def \_\_init\_\_(self, lado):

        self.lado = lado

    def area(self):

        return self.lado\*\*2

def calcular\_area(figura):

    return figura.area()

circulo = Circulo(5)

cuadrado = Cuadrado(10)

print(f"Area del circulo: {calcular\_area(circulo)}")

print(f"Area del cuadrado: {calcular\_area(cuadrado)}")