Ejercicio 1: Crear una clase
Consigna: Define una clase llamada Animal con un atributo nombre. Luego, crea una instancia de la clase y muestra el nombre.

Ejercicio 2: Añadir un método
Consigna: Añade un método llamado hacer_sonido a la clase Animal que imprima un sonido.

Ejercicio 3: Atributos adicionales
Consigna: Añade un atributo edad a la clase Animal y muestra tanto el nombre como la edad del animal.

Ejercicio 4: Método que usa atributos
Consigna: Añade un método llamado descripcion que devuelva una cadena con el nombre y la edad del animal.

Ejercicio 5: Modificar atributos
Consigna: Añade un método llamado cumplir_años que incremente la edad del animal en 1.

```
#defino la clase
class Animal:
   def __init__(self, nombre, edad):
       self.nombre = nombre
       self.edad = edad

#agrego un metodo que imprime un sonido
   def hacer_sonido(self):
       sonido = ("wau wau wau")
       return sonido

   def descripcion(self):
       return "mi perra se llama {} y tiene {}
años".format(self.nombre, self.edad)

   def cumplir_años(self):
       self.edad = (self.edad + 1)
       return f"hoy es su cumple y su nueva edad es de {self.edad}
años"


#creo la instancia de la clase
mi_animal = Animal("lola", 5)


print(f"mi perrita se llama {mi_animal.nombre} y cuando pasa gente por
la calle hace {mi_animal.hacer_sonido()}")
```

```
print(mi_animal. descripcion())
print(mi_animal.cumplir_años())
```

Ejercicio 6: Clase base y clase derivada
Consigna: Crea una clase base Animal con un método hablar. Luego, crea una clase derivada Perro que herede de Animal y sobrescriba el método hablar.

```
class Animal:
    def __init__(self, nombre):
        self.nombre = nombre

    def hablar(self):
        print(f"{self.nombre} puede hablar")

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

    def hablar(self):
        print(f"mi perro {self.nombre} ¡hace wau wau!")

mi_animal = Animal("lola")
mi_perro = Perro("lola")

mi_animal.hablar()
mi_perro.hablar()
```

Ejercicio 7: Herencia y atributos
Consigna: Crea una clase base Vehiculo con un atributo marca. Luego, crea una clase derivada Coche que herede de Vehiculo y tenga un atributo adicional modelo.

```
class Vehiculo:
    def __init__(self, marca):
        self.marca = marca

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

mi_coche = Coche("toyota", 2004)

print(f"mi auto es un {mi_coche.marca}, del año {mi_coche.modelo}")
```

Ejercicio 8: Métodos adicionales en la clase derivada
Consigna: Crea una clase base Persona con un método saludar. Luego, crea una clase derivada Estudiante que herede de Persona y tenga un método adicional estudiar.

```
class Persona:
    pass

    def saludar(self):
        print("hola, como va?")

class Estudiante(Persona):
    pass

    def estudiar(self):
        print("tengo que estudiar")

alumno = Estudiante()
alumno.saludar()
alumno.estudiar()
```

Ejercicio 9: Herencia y métodos sobrescritos
Consigna: Crea una clase base Figura con un método area. Luego, crea una clase derivada Cuadrado que herede de Figura y sobrescriba el método area.

```
class Figura:

    def area(self):
        pass

class Cuadrado(Figura):
    def __init__(self, lado):
        self.lado = lado

    def area(self):
        return self.lado * self.lado

mi_cuadrado = Cuadrado(4)
print(f"El área del cuadrado es de: {mi_cuadrado.area()}")
```

Ejercicio 10: Herencia y uso de super()
Consigna: Crea una clase base Empleado con un método trabajar. Luego, crea una clase derivada Gerente que herede de Empleado y use super() para llamar al método trabajar.

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


    def trabajar(self):
        print(f"{self.nombre} está trabajando")

class Gerente(Empleado):
    def __init__(self, nombre):
        super().__init__(nombre)

    def trabajar(self):
        super().trabajar()

el_empleado = Empleado("Juan", 100)
el_empleado.trabajar()
el_gerente = Gerente("Diego", 1000)
el_gerente.trabajar()
```

Ejercicio 11: Herencia y inicialización de atributos
Consigna: Crea una clase base Producto con un atributo nombre. Luego, crea una clase derivada Electronico que herede de Producto y tenga un atributo adicional precio.

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

class Electronico(Producto):
    def __init__(self, nombre, precio):
        super().__init__(nombre)
        self.precio = precio

Producto_electronico = Electronico("tele", 100000)
print(f"El producto es un {Producto_electronico.nombre} y su precio es
de {Producto_electronico.precio}")
```

Ejercicio 12: Herencia y métodos adicionales
Consigna: Crea una clase base Instrumento con un método tocar. Luego, crea una clase derivada Guitarra que herede de Instrumento y tenga un método adicional afinar.

```python
class Instrumento:
    def __init__(self, nombre):
```

```
        self.nombre = nombre

    def tocar(self):
        print(f"Mi {self.nombre} está sonando")

class Guitarra(Instrumento):
    def __init__(self, nombre):
        super().__init__(nombre)

    def afinar(self):
        print(f"Mi {self.nombre} se está afinado")

mi_guitarra = Guitarra("guitarra")

mi_guitarra.tocar()
mi_guitarra.afinar()
```

Ejercicio 13: Herencia y atributos adicionales
Consigna: Crea una clase base Libro con un atributo titulo. Luego, crea una clase derivada Ebook que herede de Libro y tenga un atributo adicional formato.

```
class Libro:
    def __init__(self, titulo):
        self.titulo = titulo

class Ebook(Libro):
    def __init__(self, titulo, formato):
        super().__init__(titulo)
        self.formato = formato

mi_ebook = Ebook("El gran libro", "PDF")
print(f"El libro se llama {mi_ebook.titulo} y su formato es
{mi_ebook.formato}")
```

Ejercicio 14: Herencia y métodos sobrescritos con super()
Consigna: Crea una clase base Vehiculo con un método moverse. Luego, crea una clase derivada Bicicleta que herede de Vehiculo y sobrescriba el método moverse usando super().

```
class Vehiculo:
    def moverse(self):
        print("El vehículo esta avanzando")
```

```
class Bicicleta(Vehiculo):
    def moverse(self):
        super().moverse()
        print("La bicicleta se mueve pedaleando")

mi_bicicleta = Bicicleta()
mi_bicicleta.moverse()
```

Ejercicio 15: Herencia y métodos adicionales con super()
Consigna: Crea una clase base Persona con un método presentarse. Luego, crea una clase derivada Profesor que herede de Persona y tenga un método adicional enseñar usando super().

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

    def presentarse(self):
        print(f"Me llamo {self.nombre}")

class Profesor(Persona):
    def __init__(self, nombre, materia):
        super().__init__(nombre)
        self.materia = materia

    def enseñar(self):
        print(f"Mi nombre es {self.nombre} y enseño {self.materia}")

profesor = Profesor("Juan", "Matemáticas")
profesor.presentarse()
profesor.enseñar()
```