Temas

- POO (Programación Orientada a Objetos)
- Clase
- Objeto
- Atributos y Métodos
- Mensajes entre objetos
- Encapsulamiento

POO (Programación orientada a objetos)

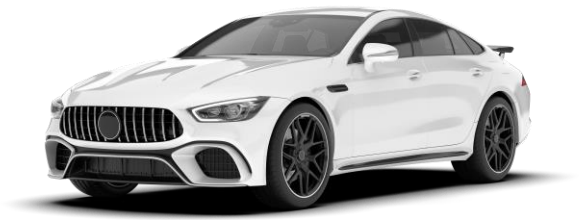
Pensemos en el mundo real: todo lo que nos rodea son **objetos** con **características** y **comportamientos**.

Un auto puede tener las siguientes **características**:

- Marca
- Modelo
- Color

También puede tener los siguientes **comportamientos**:

- Arrancar
- Acelerar
- Frenar

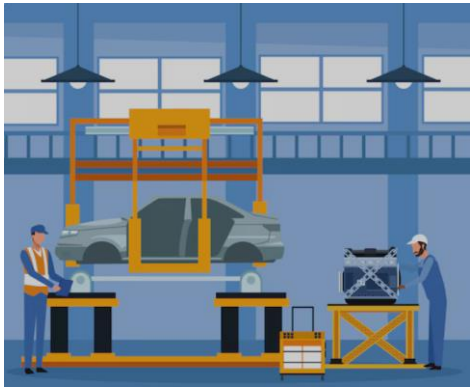


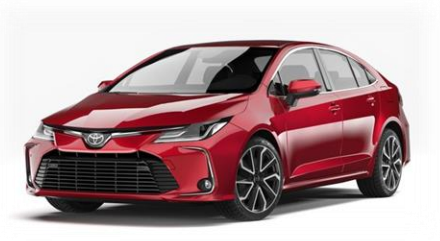
La **POO** intenta simular esto en el software.

Clases

Estas características (**atributos**) y comportamientos (**métodos**) nos permiten definir una plantilla de como debe representarse un objeto especifico. Esta plantilla se conoce como **clase**.

Una clase es
como una fabrica





Atributos:
Marca: Toyota
Modelo: Corolla
Color: Rojo

Atributos:
Marca: Ford
Modelo: Fiesta
Color: Blanco

Clases

Siguiendo con el ejemplo de los autos, podríamos definir nuestra clase Auto de la sig. manera:

```
class Auto:
    #Atributos
    def __init__(self, color, marca, modelo):
        self.color = color
        self.marca = marca
        self.modelo = modelo
    #Metodos
    def arrancar(self):
        return "El auto: " + self.marca + " modelo: " + self.modelo + " ha arrancado."
    def frenar(self):
        return "El auto: " + self.marca + " modelo: " + self.modelo + " ha frenado."
```

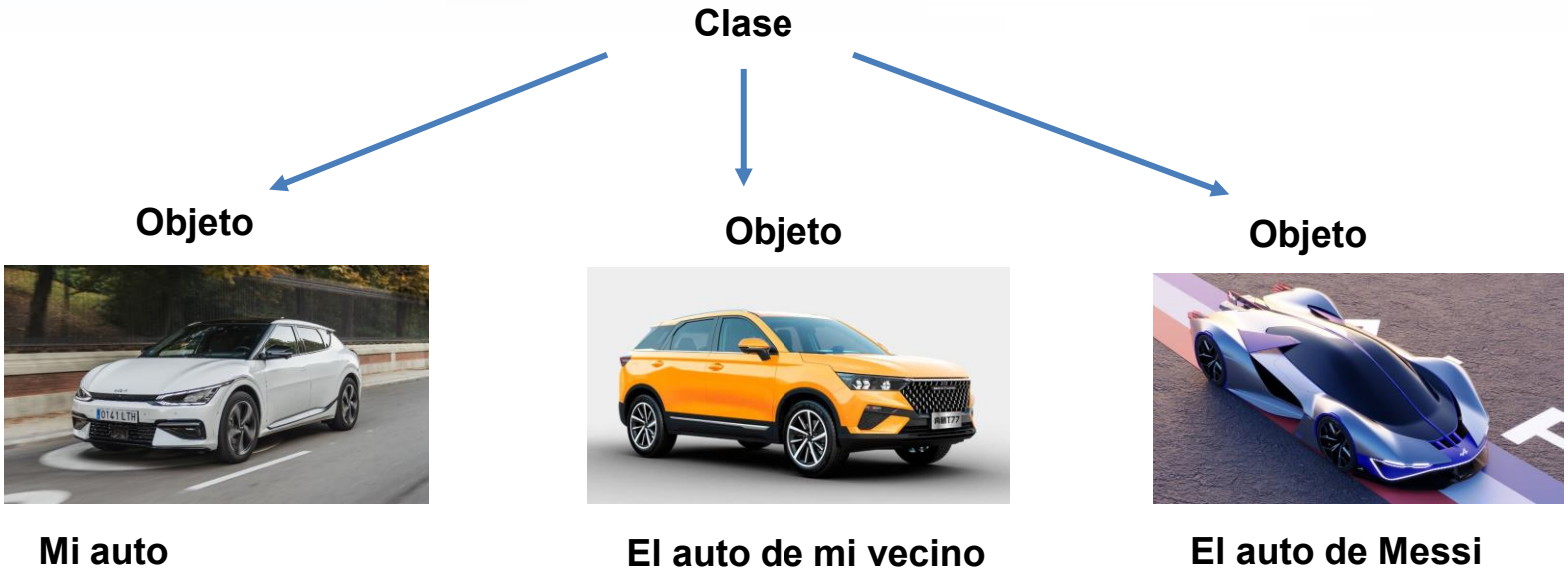
¿Qué es una clase?

- Es una **plantilla** o **molde** que define cómo serán los objetos.
- Describe los atributos (características)
- Describe los métodos (comportamientos)
- No ocupa memoria

Tener en cuenta que una clase no es un objeto, sino que define como será un objeto.

Objeto

Un **objeto** es una **instancia concreta** de una clase. Ahora puedo representar cualquier tipo de auto.



Objeto

Ahora tendremos un "ejemplar" real creado a partir del plano de la clase.

Podremos crear múltiples objetos a partir de la misma clase, y cada uno de ellos tendrá sus propios valores para sus atributos.

Por ejemplo, podríamos representar varios autos de la sig. manera:

```
mi_auto = Auto("rojo", "Ford", "Fiesta")
tu_auto = Auto("azul", "Toyota", "Corolla")
```

Atributo

Los **atributos** son las **características o propiedades** de un objeto. Son como variables que pertenecen a un objeto y almacenan información sobre él.

Por ejemplo, en los objetos que creamos anteriormente:

- Color, marca y modelo son los atributos.
- Para mi_auto los atributos son “rojo”, “Ford” y “Fiesta”.
- Para tu_auto los atributos son “azul”, “Toyota” y “Corolla”.

Podremos acceder a los atributos de un objeto usando la notación de punto : objeto.atributo

```
print(mi_auto.color)    # Salida: rojo
print(tu_auto.marca)    # Salida: Toyota
```

Método

Son las **acciones** o **comportamientos** que un **objeto** puede realizar. Son **funciones** que pertenecen a una clase y operan sobre los datos (**atributos**) del objeto.

Por ejemplo, en la **clase** Auto definida anteriormente, arrancar() y frenar() son **métodos**.

Podremos llamar a los **métodos** de un **objeto** utilizando la notación de punto: objeto.metodo()

```
print(mi_auto.arrancar()) # Salida: El Ford Fiesta de color rojo ha arrancado.
print(tu_auto.frenar())   # Salida: El Toyota Corolla de color azul ha frenado.
```

Mensaje entre Objetos

Los objetos pueden interactuar entre sí enviándose **mensajes**. Un mensaje es esencialmente una llamada a un método de otro objeto.

Imaginemos el caso donde se necesita de un conductor que encienda el auto.

- El conductor también tiene características(**atributos**) y comportamientos (**metodos**)
- El conductor puede ser representado como un **objeto**
- Un atributo del conductor puede ser su nombre
- Un método del conductor puede ser encender el auto

Si el conductor es un **objeto** y el auto también, entonces el conductor puede **enviar un mensaje** al **objeto** auto para que arranque.

Mensaje entre Objetos

Siguiendo los ejemplos anteriores, podremos codificar el mensaje de la sig. manera:

```
class Conductor:
    def __init__(self, nombre):
        self.nombre = nombre
    def encender_auto(self, auto):
        print(f"{self.nombre} está intentando encender el auto.")
        # El Conductor envía un mensaje (llama al método arrancar) al objeto Auto
        print(auto.arrancar())

mi_conductor = Conductor("Juan")
mi_conductor.encender_auto(mi_auto)
# Salida:
# Juan está intentando encender el auto.
# El Ford Fiesta de color rojo ha arrancado.
```