ESCUELA DE INGENIERÍA

# Clases y objetos

PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE


# Cristian Ruz Ruz

Profesor Asistente Adjunto, Departamento de Ciencia de la Computación

Facultad de Ingeniería

Pontificia Universidad Católica de Chile

# Índice

- Programación orientada a objetos
- Clase, objetos, atributos y métodos
- Clase, objetos, atributos y métodos en Python

# Introducción

- ¿Qué es la programación orientada a objetos?
- ¿Qué son las clases, objetos y atributos?
- ¿Cómo se implementa en Python?

# Programación orientada a objetos

# ¿Qué es la Programación Orientada a Objetos?

- **Paradigma de programación.**
- Objetos en la vida real: elementos que poseen **características** y **se pueden manipular.**
- Objetos de software: colecciones de **datos** que poseen **comportamiento.**

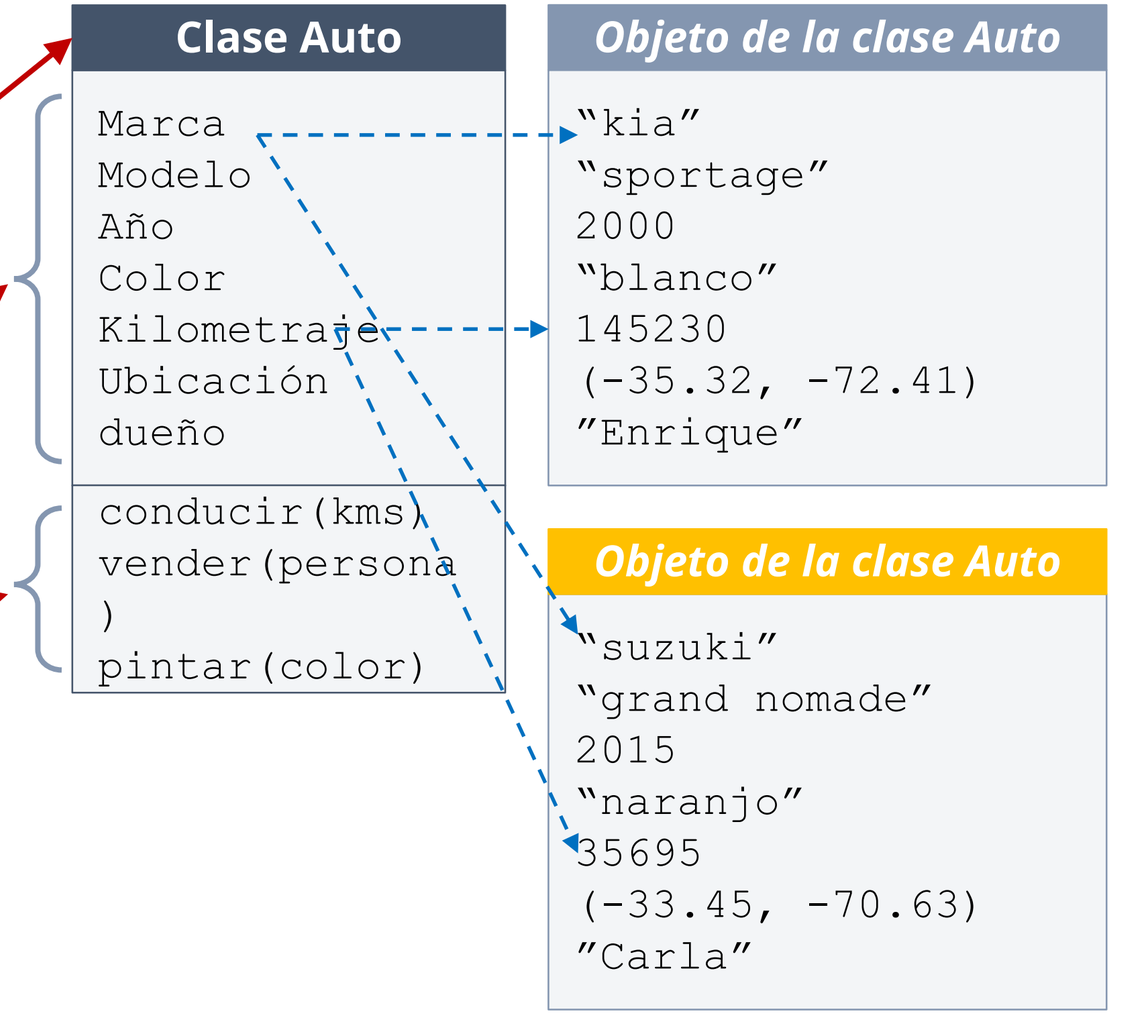
```
marca
modelo
año
color
kilometraje
ubicación
dueño
```

```
conducir(kms)
vender(persona)
pintar(color)
```

# Clases, atributos, objetos y métodos

# Clases, atributos, métodos y objetos

- Paradigma de programación basado en la interacción de **objetos.**
- **Clases**: plantillas que definen una lista de **atributos** y **métodos** que posee un **objeto.**
- **Atributos:** campos o características del **objeto.**
- **Métodos:** acciones que se pueden aplicar sobre un **objeto.**
- **Objeto:** instancia de una clase.



# Clases, atributos, objetos y métodos en Python

# Definiendo clases: `class`

**Definición de clase**

**Definición de atributos**
**Método *inicializador***

**Métodos**

**Creación de instancias de Auto (objetos de clase Auto)**

```
class Auto:
    def __init__(self, ma, mo, a, c, k):
        self.marca = ma
        self.modelo = mo
        self.año = a
        self.color = c
        self.kilometraje = k
        self.ubicacion = (-33.45, -70.63)
        self.dueño = None

    def conducir(self, kms):
        self.kilometraje += kms

    def vender(self, nuevo_dueño):
        self.dueño = nuevo_dueño

    def leer_odometro(self):
        return self.kilometraje
```

```
a = Auto("kia", "sportage", 2000, "blanco", 145230)
b = Auto("suzuki", "grand nomade", 2015, "naranjo", 35695)
b.conducir(1450)
a.vender("Enrique")
print(b.leer_odometro())
```

Invocación de métodos

# Referencia al objeto: `self`

```python
class Auto:
    def __init__(self, ma, mo, a, c, k):
        self.marca = ma
        self.modelo = mo
        self.año = a
        self.color = c
        self.kilometraje = k
        self.ubicacion = (-33.45, -70.63)
        self.dueño = None

    def conducir(self, kms):
        self.kilometraje += kms

    def vender(self, nuevo_dueño):
        self.dueño = nuevo_dueño

    def leer_odometro(self):
        return self.kilometraje
```

```python
a = Auto("kia", "sportage", 2000, "blanco", 145230)
b = Auto("suzuki", "grand nomade", 2015, "naranjo", 35695)
print(a.marca)
print(b.color)
print(a.año)
```

- Al invocar el método inicializador, se de métodos usan un parámetro menos que en su definición.
- El primer argumento, `self`, es una **auto-referencia.**
- Se usa en los métodos para referirse al atributo o método del objeto actual sobre el que estamos trabajando.
- `self` es una convención. Puede ser cualquier nombre.

# Síntesis

- Programación orientada a objetos.
- Clases, objetos, atributo y métodos.
- Clases, objetos, atributo y métodos en Python.

# Referencias bibliográficas


- Documentación Python 3.8. Classes. https://docs.python.org/3/tutorial/classes.html


PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE