

L U M B R E



Atributos, métodos y self

Grado: 2

Materia: Programación Orientada a Objetos · Programación

11/10/2026

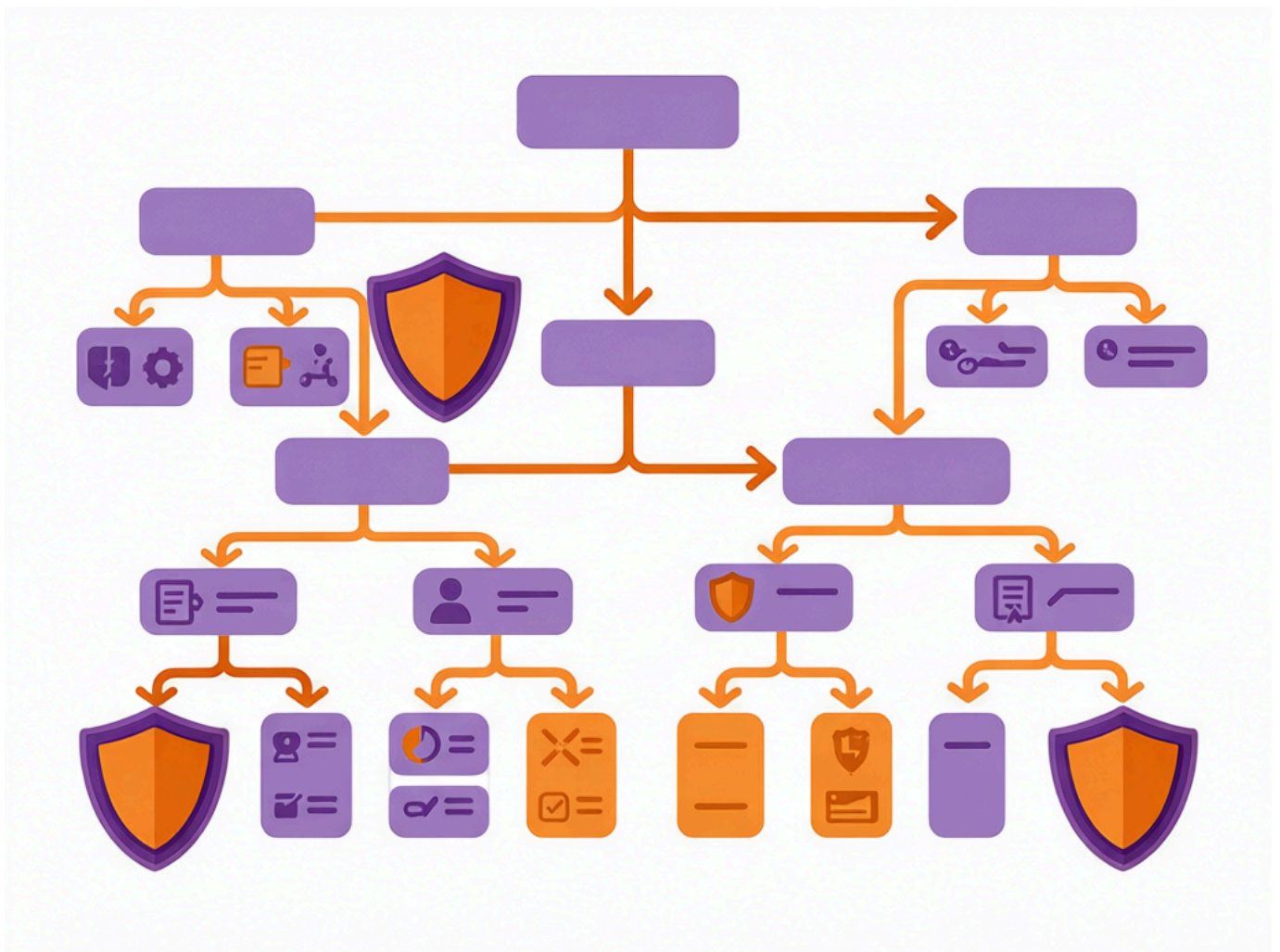
Antes de empezar

El parámetro self es la llave que hace que un método sepa a qué objeto pertenece. Entenderlo bien elimina la mayor parte de la confusión inicial de la POO.

QUÉ APRENDERÁS

- Explicar el rol de self como referencia a la instancia, y distinguir atributos y métodos de objeto frente a atributos y métodos de clase.

Atributos, métodos y self



Principios de la programación orientada a objetos: clases, herencia y polimorfismo.

Mapa conceptual

- Programación OO
 - Clases
 - Objetos

- Atributos
- Metodos self
- Encapsulamiento
 - Public
 - Private
 - Properties
- Herencia
 - Polimorfismo
 - Abstractas
 - Interfaces
- Composicion
 - Sobre herencia
 - Delegacion
 - Acoplamiento
- Proyecto
 - Paquetes
 - Modulos
 - Convenciones

El mapa mental interactivo es interactivo en la versión online.



Self: el objeto se reconoce a sí mismo

Todo método recibe, sin que lo pidas, una referencia al objeto sobre el que opera: `self`. Es lo que permite que «`ladrar()`» sepa qué perro está ladrando.

¿Qué es self?

En Python, `self` es el nombre por convención del primer parámetro de un método de instancia: representa AL objeto que invoca el método. Cuando llamas `mi_perro.ladRAR()`, Python traduce eso a `Perro.ladRAR(mi_perro)` y pasa el objeto como `self`. No es mágico, es azúcar de paso de argumento.

Acceder al estado

Dentro de un método, `self.nombre` lee el atributo de ese objeto concreto, y `self.edad = 5` lo escribe. Sin `self`, no hay forma de distinguir el estado de un objeto del de otro de la

misma clase.

`__init__` y el constructor

El método `__init__` se ejecuta al instanciar: inicializa los atributos del objeto nuevo. Su primer parámetro también es `self`; los demás son los argumentos con que creaste el objeto. Ahí se «siembra» el estado inicial de cada instancia.

Atributo de instancia vs de clase

- De instancia: `self.x`, pertenece a cada objeto; un cambio no afecta a los demás.
- De clase: definido en el cuerpo de la clase, compartido por todas las instancias salvo que una lo redefina. Útil para contadores globales o valores por defecto.

La diferencia `self` vs `cls` es la de objeto vs plantilla.

Elemento	Dónde vive	Alcance	Ejemplo
Atributo de instancia	en <code>self</code>	solo ese objeto	<code>self.nombre</code>
Atributo de clase	en la clase	compartido	<code>Perro.especie</code>
Método de instancia	recibe <code>self</code>	opera sobre un objeto	<code>def ladrar(self)</code>
Método de clase	recibe <code>cls</code>	opera sobre la clase	<code>@classmethod def contar(cls)</code>

1. ¿Qué representa `self` dentro de un método de instancia?

- ☐ La clase entera
- ☐ El objeto concreto que invocó el método
- ☐ El módulo actual
- ☐ Nada, es opcional

El precio de olvidar `self`

Un error clásico es escribir una variable dentro de un método sin `self`. entonces: nace una variable local que muere al terminar, no un atributo del objeto. Si quieres guardar algo en el objeto, debe vivir colgado de `self`.

Mutabilidad compartida

Cuidado con atributos de clase que son listas o diccionarios: todas las instancias comparten el MISMO objeto mutable, y modificarlo desde una lo cambia para todas. Es un bug elegante que atrapa a los principiantes.

2. `mi_perro.ladRAR()` y `Perro.ladRAR(mi_perro)` son equivalentes en Python.

- ☐ Verdadero
- ☐ Falso

Encadenar

Como los métodos reciben y pueden devolver el objeto, se pueden encadenar: `p.set_nombre(A).set_edad(3)`. Devolver `self` permite esta fluidez. No es necesario, pero aparece en librerías de diseño de interfaces.

Errores frecuentes

- Olvidar `self` como primer parámetro de un método de instancia.
- Asignar un atributo nuevo fuera de `__init__` de forma dispersa, dificultando leer el estado del objeto.
- Confundir atributo de clase mutable compartido con atributo de instancia.

3. Asigna cada definición a su tipo:

Elementos:

- `self.nombre`
- `Perro.especie` compartida
- `def ladRAR(self)`
- `def crear_desde_dict(cls)`

Instancia	. . .
Clase	. . .

Documentación de Python <https://docs.python.org/es/3/tutorial/classes.html> El tutorial oficial sobre clases y el papel de `self`.

VIDEO

Video complementario

Recurso audiovisual para reforzar los conceptos.



Escanea para verlo en Lumbre

<https://www.youtube.com/watch?v=Zbe5K6uEJYw>

4. Explica con tus palabras por qué hace falta self y qué pasaría si un método accediera a «nombre» sin self. Muestra un pequeño ejemplo donde la falta de self cause un bug.

Python: Classes <https://docs.python.org/3/tutorial/classes.html> Tutorial oficial de clases en Python 3.

Referencia curricular

Grado:	2
Materia:	Programación Orientada a Objetos · Programación

Versión interactiva completa (con actividades, videos y autoevaluación):



<https://univ.lumbre.lat/object/cdia-poo-002-atributos-metodos-y-self>

Material educativo generado desde Lumbre · lumbre.lat