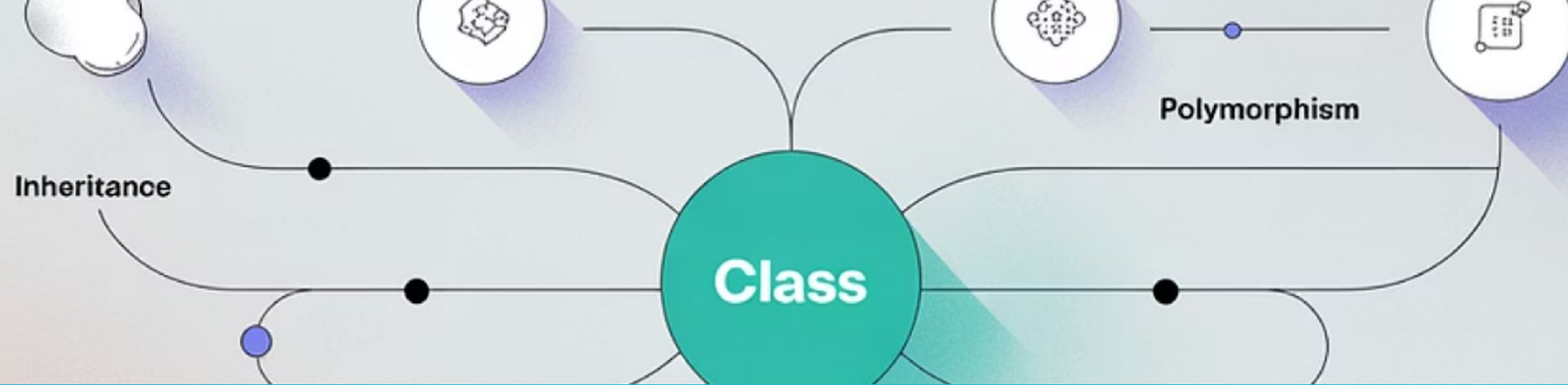


Métodos, Encadenamiento y Métodos Especiales en Python

Bootcamp Full Stack Python – Jueves 24 de julio



Code
your
future



Aprendizaje Esperado

AE N°2:

Comprender y aplicar correctamente el uso de métodos, encadenamiento de métodos, métodos de clase y métodos estáticos en Python.

Objetivos de la Clase

 Al terminar esta clase podrás:

1

Entender qué es un método en Python

Comprenderás la definición y uso de métodos en programación orientada a objetos.

2

Diferenciar entre métodos de instancia, clase y estáticos

Conocerás las características y usos de cada tipo de método.

3

Encadenar métodos (method chaining)

Aprenderás a llamar varios métodos en una sola línea de código.


4

Aplicar todo en ejemplos prácticos

Pondrás en práctica los conceptos aprendidos con ejercicios reales.

Introducción + Pregunta inicial

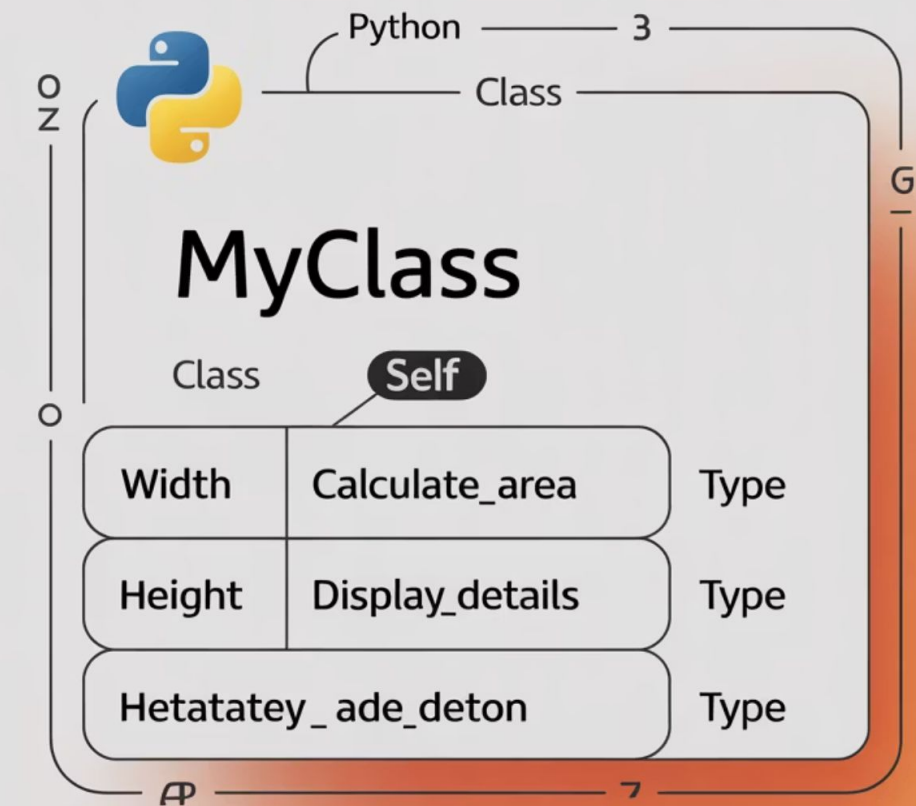
¿Qué diferencia hay entre una función y un método?

 Método = función dentro de una clase, que accede a la instancia (self).

Función	Método
Fuera de clase	Dentro de clase
No usa self	Usa self

¿Qué es un método de instancia?

- Función definida en una clase.
- Accede a datos de la instancia con self.
- ✓ Se usa para comportamiento específico de objetos.



Ejemplo básico de método de instancia

```
class Persona:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad
    def saludar(self):
        return f"Hola, soy {self.nombre} y tengo {self.edad} años."
```

Explicación paso a paso del ejemplo



`__init__`

Constructor que inicializa los atributos nombre y edad



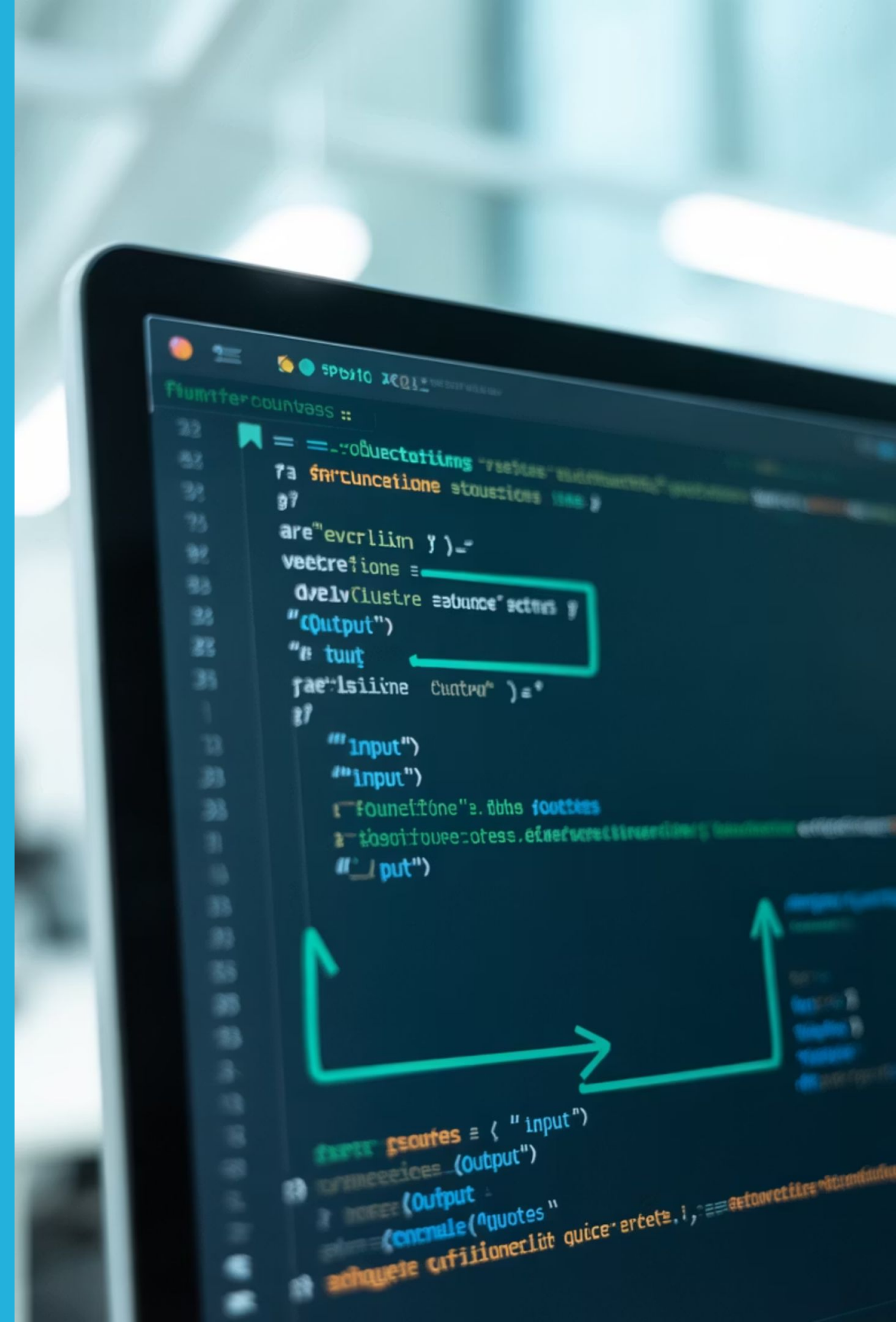
`saludar`

Método que accede a atributos con `self`



`persona1.saludar()`

Ejecuta el método en la instancia creada



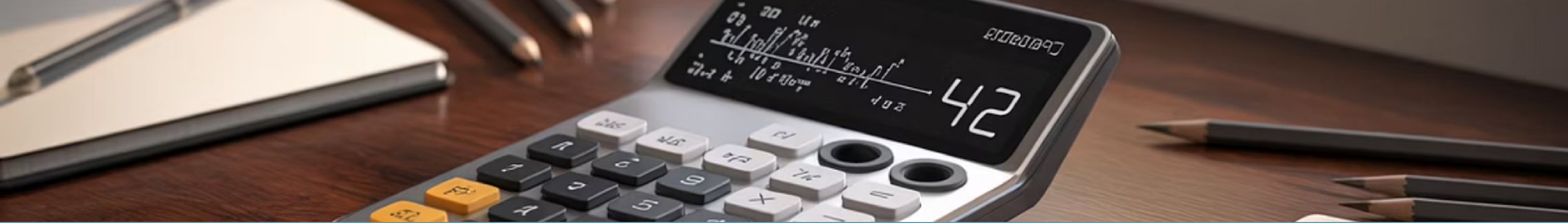
¿Qué es encadenar métodos?

🔗 **Llamar varios métodos en una sola línea**

📌 **Clave:** Retornar self en cada método.

```
obj.metodo1().metodo2().metodo3()
```





Ejemplo de encadenamiento

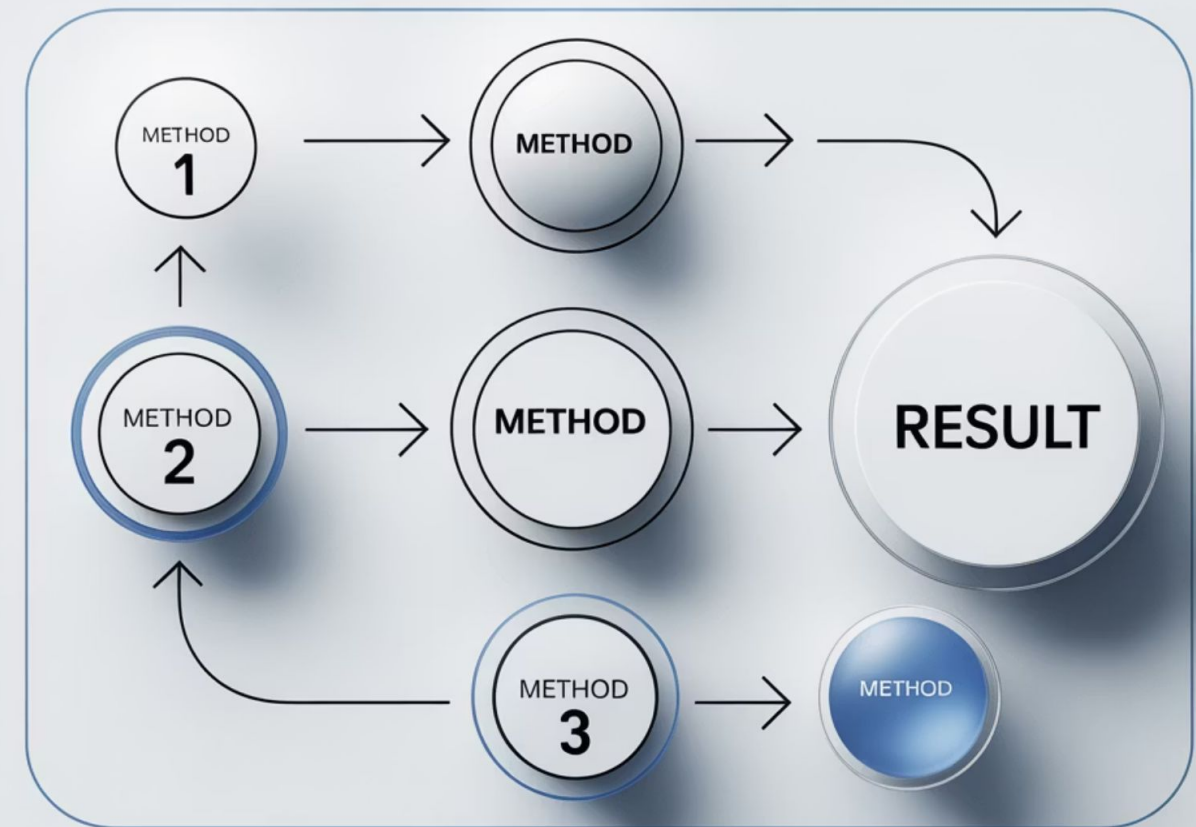
```
class Calculadora: def __init__(self, valor=0): self.valor = valor def sumar(self, numero): self.valor += numero return self def multiplicar(self, numero): self.valor *= numero return self def mostrar(self): print(f"Resultado: {self.valor}") return self
```

Ejecución de método encadenado

```
calc = Calculadora()calc.sumar(5).multiplicar(2).mostrar()#  
Resultado: 10
```

📌 sumar y multiplicar devuelven self

📌 mostrar() imprime y también retorna self



¿Qué es un método de clase?

 Usa el decorador `@classmethod`

 Recibe `cls` en lugar de `self`

 Sirve para crear objetos o modificar la clase



Ejemplo de método de clase

```
class Producto:
    def __init__(self, nombre, precio):
        self.nombre = nombre
        self.precio = precio
    @classmethod
    def desde_string(cls, cadena):
        nombre, precio = cadena.split("-")
        return cls(nombre, float(precio))
```

¿Qué es un método estático?

📌 Usa el decorador `@staticmethod`

✅ No recibe ni `self` ni `cls`

🔧 Función que no depende del estado de la clase



Ejemplo de método estático

```
class Util: @staticmethod def es_par(numero): return numero % 2 == 0
```


Comparación resumen de tipos de métodos

Tipo de método	Decorador	Primer parámetro	Accede a...
Instancia	<i>(ninguno)</i>	self	atributos de objeto
Clase	@classmethod	cls	clase
Estático	@staticmethod	ninguno	nada por defecto

Actividad práctica guiada

 Crear clase Libro con:

Atributos:

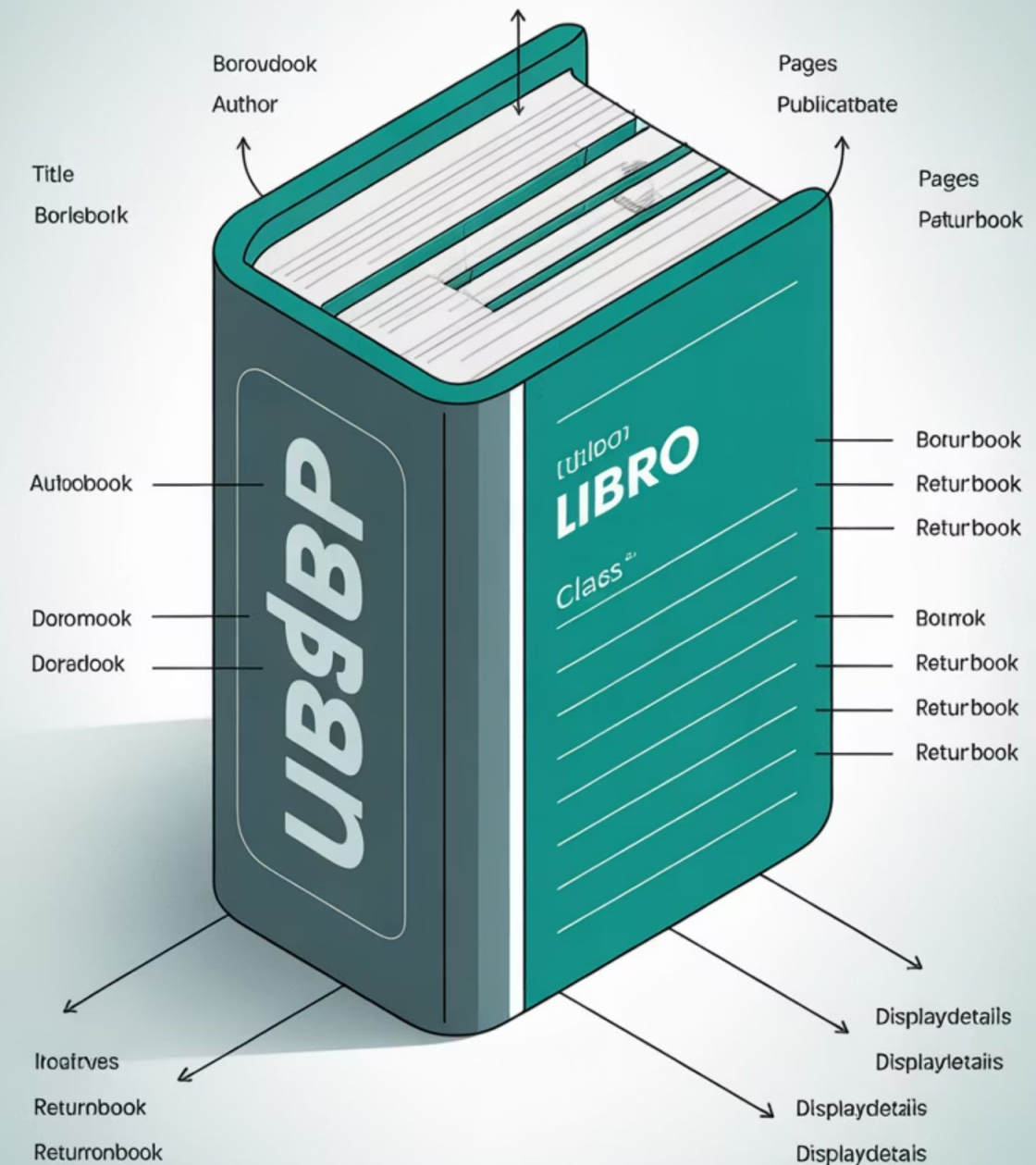
- titulo
- autor
- paginas

Métodos:

- mostrar_info()
- leer(paginas)
- progreso()
- @classmethod
cargar_desde_string(...)
- @staticmethod
es_libro_grande(...)

Bonus: encadenar leer().progreso()

Libro



Trivia en equipos (lúdica)

🧠 Preguntas para repasar:

1

¿Para qué sirve `@classmethod`?

2

¿Qué parámetro usan los métodos de clase?

3

¿Qué retorna un método encadenado?

4

¿Qué hace `@staticmethod`?

🎉 Kahoot

🎁 ¡Premio simbólico al equipo ganador!



Cierre y reflexión

Repaso final:

- Diferencias entre tipos de métodos
- ¿Qué aprendiste hoy?
- ¿Qué te pareció más interesante o difícil?

 "¡Estás pensando cada vez más como programador/a!"



Ejercicios Finales

✚ Ejercicio 1: Clase CuentaBancaria

1. Atributos: titular, saldo
2. Métodos:
 - depositar(monto)
 - retirar(monto)
 - mostrar_saldo()
3. Bonus: permite depositar().mostrar_saldo()



✚ Ejercicio 2: Clase Vehiculo

1. Atributos: marca, modelo, velocidad
2. Métodos:
 - acelerar(km)
 - frenar(km)
 - mostrar_estado()
 - @classmethod crear_desde_texto("Marca-Modelo")
 - @staticmethod es_rapido(velocidad) → True si >120





Recursos extra

 **Para seguir practicando:**



Documentación oficial Python

 docs.python.org/3/



Video recomendado

 **POO en Python** <https://www.youtube.com/watch?v=qiSCMNBIP2g>