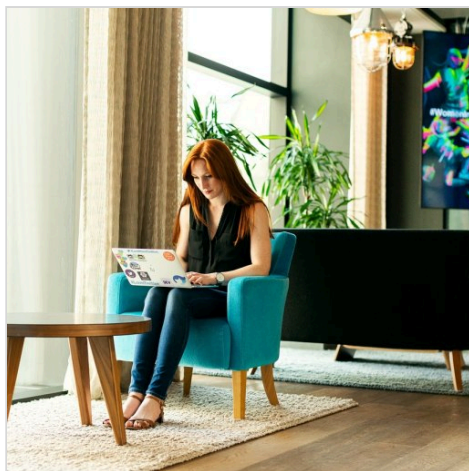


L U M B R E



Clases abstractas e interfaces

Grado: 2

Materia: Programación Orientada a Objetos · Programación

11/10/2026

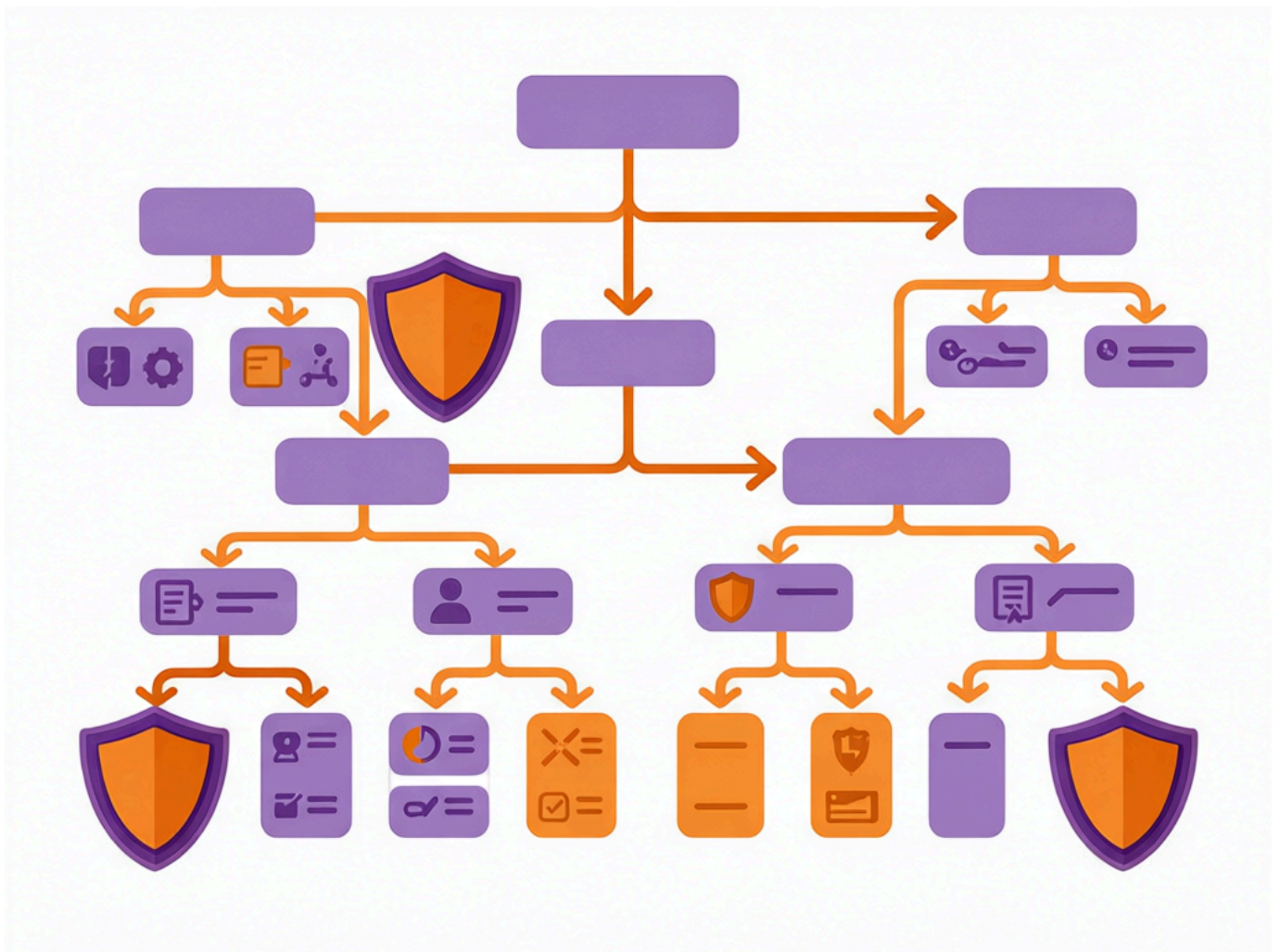
Antes de empezar

Una clase abstracta es un contrato: dice «todo lo que me herede DEBE saber hacer esto». Evita implementaciones a medias y documenta la intención del diseño.

QUÉ APRENDERÁS

- Definir contratos con clases abstractas e interfaces para forzar a las subclasses a implementar ciertos comportamientos.

Clases abstractas e interfaces



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.



Plantillas que no se usan solas

Hay conceptos que no tienen instancia lógica: no existe «un Animal» genérico suelto, siempre es un perro o un gato. Una clase abstracta formaliza esa idea: se hereda, pero no se instancia.

Clase abstracta

Una clase abstracta define métodos que la subclase está obligada a implementar. No se puede crear un objeto de ella directamente: funciona como base incompleta que fija una estructura común. En Python se construye con el módulo `abc` y el decorador `abstractmethod`.

El método abstracto

Un método abstracto se declara sin cuerpo (o con pass): es una promesa. Cualquier subclase concreta que no lo reemplace quedará ella misma como abstracta y tampoco podrá instanciarse. Así el compilador (en otros lenguajes) o la creación de objeto (en Python) avisa del incumplimiento.

Interfaz vs clase abstracta

- Interfaz: solo contratos, ningún estado ni implementación. «Qué sabes hacer».
- Clase abstracta: contratos además de algo de código e estado compartido. «Qué eres y qué haces».

Para qué sirven

Documentan el diseño: quien lee la base sabe qué debe ofrecer toda implementación. Y obligan: si añades un método abstracto, toda subclase que lo omita fallará al instanciarse, no cuando ya esté en producción.

La base abstracta mezcla contrato y reutilización; la interfaz solo contrato.

Concepto	Se instancia	Contiene	Uso
Clase concreta	Si	todo implementado	objetos reales
Clase abstracta	No	métodos abstractos y tal vez código	base para heredar
Interfaz	No	solo firmas de métodos	contrato de comportamiento

1. ¿Qué ocurre al intentar crear un objeto de una clase abstracta en Python?

- ☐ Se crea normalmente
- ☐ Se lanza un error: no se puede instanciar
- ☐ Se crea con los métodos vacíos
- ☐ Python implementa los abstractos por ti

El patrón de la estrategia

Una base abstracta permite escribir código que dependa del contrato («algo que puede procesar()») y no de implementaciones concretas. Puedes intercambiar estrategias (un normalizador, un validador) siempre que cumplan el contrato: bajo acoplamiento con un esqueleto común.

ABC y mixin

En Python, heredar de ABC marca la clase como abstracta. Un mixin, en cambio, es una clase pequeña que aporta comportamiento para componerse, no para ser una jerarquía. Ambos ayudan a organizar sin sobrecargar la herencia.

2. Una clase con un método abstracto puede instanciarse si el resto de métodos están implementados.

- ☐ Verdadero
- ☐ Falso

Interfaces implícitas: protocolo

Python no fuerza tipos, así que a menudo una interfaz es informal: se espera que el objeto tenga ciertos métodos (duck typing). Los Protocol de tipado estructurado permiten declarar esas interfaces de forma explícita para el chequeo estático, sin forzar la herencia.

Cuándo usarlas

- Hay un concepto general que nunca se instancia solo: usa una base abstracta.
- Solo quieres un contrato, sin compartir código: usa una interfaz (o Protocol).
- Necesitas que el error aparezca al instanciar, no al ejecutar: abstracta.

Errores frecuentes

- Hacer abstracta una clase que sí tendría instancias lógicas.
- Dejar métodos concretos que en realidad todo hijo debería redefinir.
- Confundir interfaz con jerarquía profunda: a veces basta un Protocol y nada de herencia.

3. Clasifica cada elemento:

Elementos:

- class Forma(ABC) con area() abstracto
- class Circulo(Forma) con area()
- Protocol Comparable solo con firmas
- class Rectangulo(Forma) con area()

Abstracta	. . .
Concreta	. . .
Interfaz	. . .

Python Docs <https://docs.python.org/es/3/library/abc.html> Módulo abc: clases y métodos abstractos.

VIDEO

Video complementario

Recurso audiovisual para reforzar los conceptos.



Escanea para verlo en Lumbre

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

4. Diseña una base abstracta Exportador con métodos abstractos escribir() y soporta_formato(). Describe dos subclases concretas (CSV, JSON) y explica qué ganarías al dejar la base abstracta.

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-006-clases-abstractas-e-interfaces>

Material educativo generado desde Lumbre · lumbre.lat