

 Skillnest



# Asociación, Colaboración y Composición entre Clases en Python



Curso Full Stack Python – Chile 

Contextualizado con ejemplos chilenos reales



# Objetivos

1

Entender cómo se relacionan las clases en Python.

3

Aprender a componer y colaborar entre objetos.

## ¿Qué es una Asociación?

Una clase contiene otra como atributo.

 Un Usuario tiene una Tarjeta de Crédito

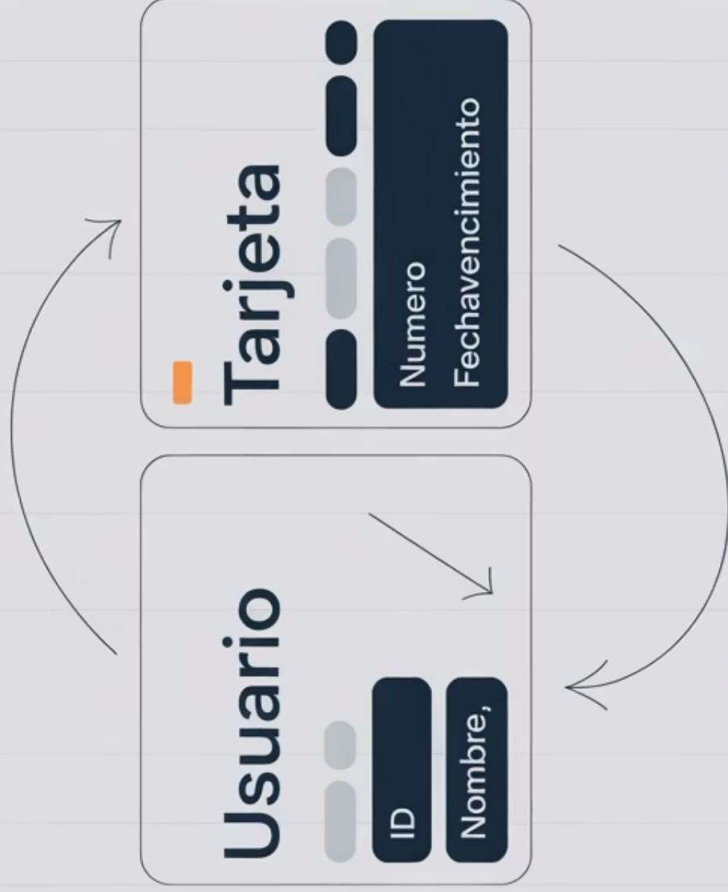
Es decir, existe una relación entre ambas clases.

 Piensa en cómo cada persona en Chile puede tener su tarjeta Visa o

Aplicar estos conceptos en contextos reales (como usuarios bancarios o empresas).

Reconocer la diferencia entre herencia y composición.





# Ejemplo de Asociación

En vez de tener saldo y límite en el Usuario, se  
TarjetaCredito.

```
# Usuario.pyfrom TarjetaCredito import
Usuario: def __init__(self, nombre,
self.nombre = nombre      self.apellido
self.email = email        self.tarjeta
0.015)
```

✓ Asociación permite separar responsabilidad

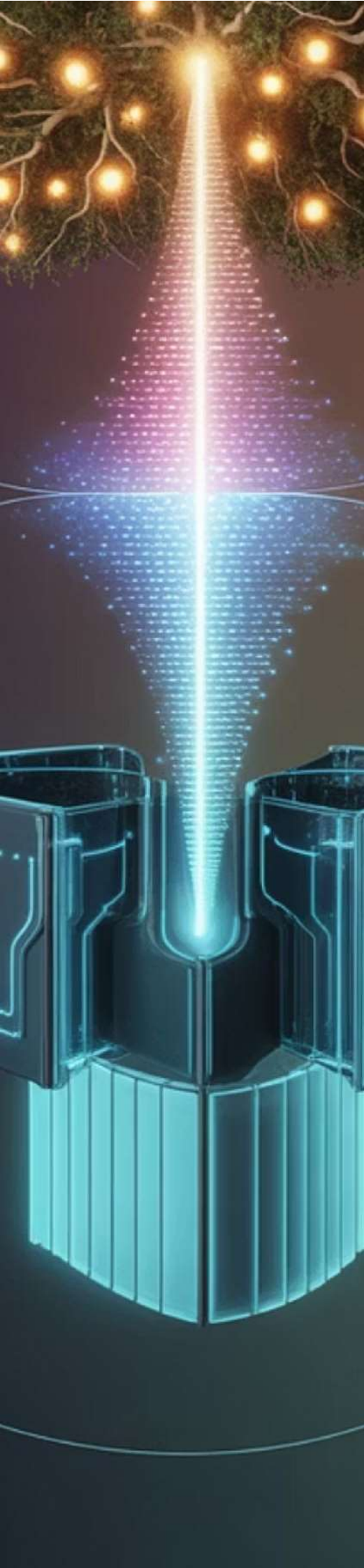
# ¿Qué es Colaboración?

📌 Una clase utiliza otra para realizar tareas, pero no depende de ella para existir.

🚗 Ejemplo: un Auto puede usar un Motor, pero no lo "posee" completamente.

Es como si alguien en Chile arrendara un auto por el día. Lo usa, pero no lo posee.





# Código de Colaboración

```
class Motor: def encender(self): return "Motor encendido"
```

# ¿Qué es Composición?

📌 Una clase contiene otra como parte esencial. Si la clase principal muere, el componente también.

🧠 Ejemplo: una Computadora contiene un CPU, si se destruye la computadora, el CPU se inutiliza.









# Código de Composición

```
class CPU: def especificaciones(self): return "Procesador Intel i7"
```

# Diferencias clave

Característica	 Colaboración	 Composición
Relación	Usa otro objeto	Posee otro objeto
Independencia	Los objetos son autónomos	El objeto depende de otros
Vida útil	Sobreviven separados	Si uno muere, los otros mueren
Ejemplo	Auto usa Motor	PC contiene procesador



# Ejemplo contextualizado: BancoEstado

🎯 Supongamos que Camila Rojas, de Temuco, tiene una tarjeta de crédito del BancoEstado.

```
usuario1 = Usuario("Camila", "Rojas",  
"camila@gmail.com")
```

Saldo inicial: \$0

Límite de crédito: \$20.000

Tasa de interés: 1.5%



## Clase TarjetaCredito.py

```
class TarjetaCredito: def __init__(self, saldo_pagar,
limite_credito, tasa_interes): self.saldo_pagar = saldo_pagar
self.limite_credito = limite_credito self.tasa_interes =
tasa_interes def compra(self, monto): if self.saldo_pagar + monto
<= self.limite_credito: self.saldo_pagar += monto print("Compra
realizada.") else: print("Límite excedido.") def resumen(self):
return f"Saldo: ${self.saldo_pagar}"
```

# Clase Usuario.py (asociación)

```
class Usuario:
    def __init__(self, nombre, apellido, email):
        self.nombre = nombre
        self.apellido = apellido
        self.email = email
        self.tarjeta = TarjetaCredito(0, 20000, 0.015)
```

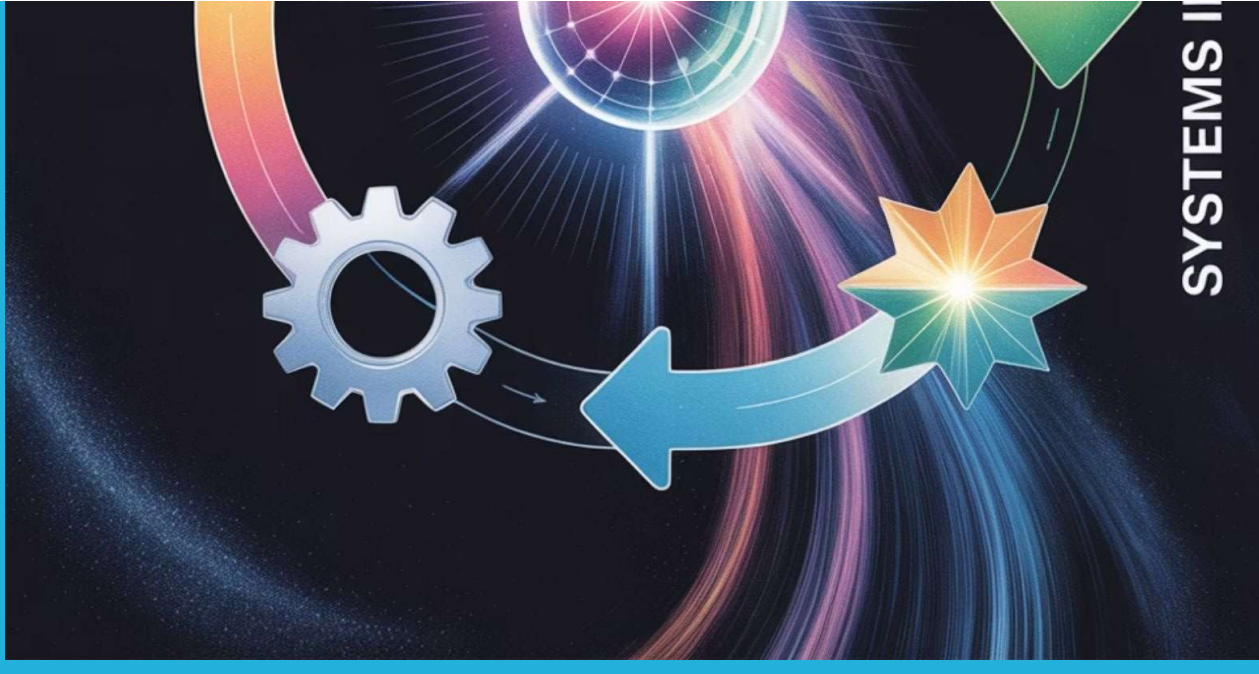


# Interacción entre clases

```
usuario1.comprar(5000)usuario1.ver_resumen()
```

Se llama a métodos del objeto tarjeta dentro del usuario.

Esto demuestra cómo las clases trabajan juntas.





# Colaboración: Auto y Motor

```
mi_auto = Auto()print(mi_auto.arrancar()) # Motor  
encendido
```

Auto necesita el motor, pero no está "casado" con él.

Puede cambiarlo por otro motor fácilmente.

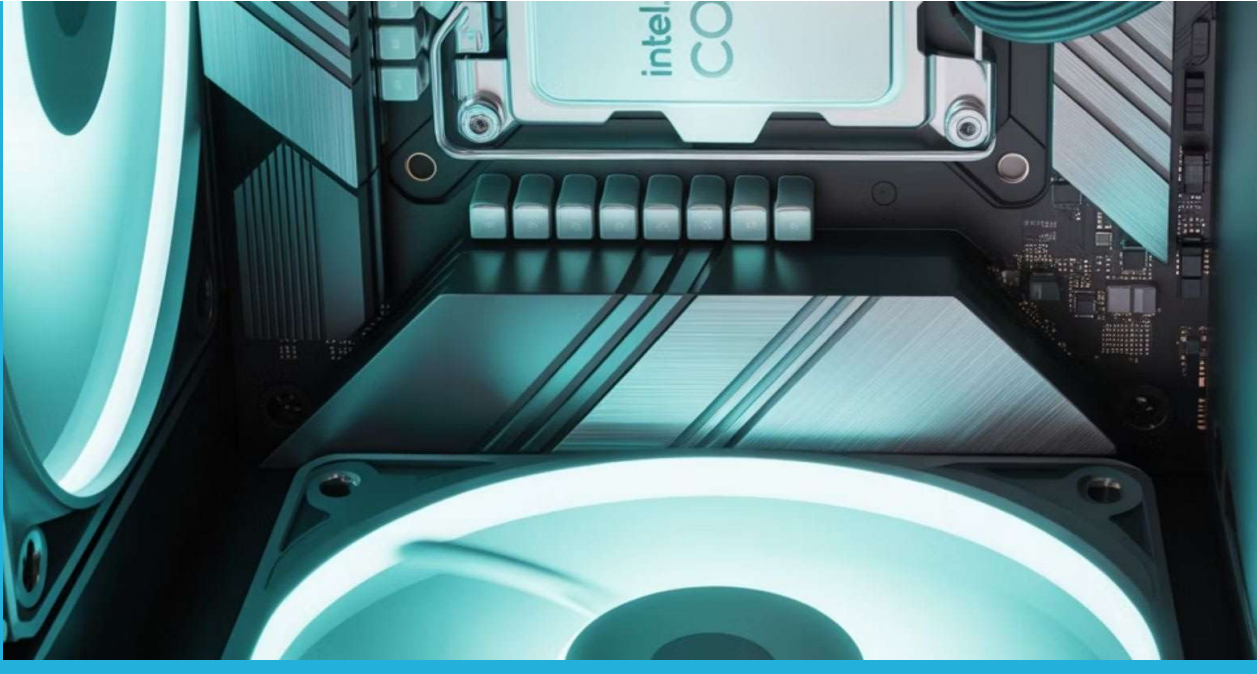


# Composición: Computadora y CPU

```
mi_pc = Computadora()print(mi_pc.mostrar_info())
```

CPU es parte esencial de la computadora.

Si se elimina `mi_pc`, el CPU también deja de existir.





## Actividad Lúdica: Kahoot

💡 Lanza un Kahoot con estas preguntas:

- ¿Qué tipo de relación es más fuerte: colaboración o composición?
- ¿Qué clase contiene otra como atributo?
- ¿En qué tipo de relación los objetos son inseparables?
- ¿Qué representa una tarjeta dentro de un usuario?



# Ejercicio Final: Estudiante y Mochila

 Crear clase Mochila con útiles escolares

 Clase Estudiante que contiene una mochila



# Código del Ejercicio

```
class Mochila:
    def __init__(self):
        self.utiles = []
        def agregar_util(self, item):
            self.utiles.append(item)
    def __init__(self, nombre):
        self.nombre = nombre
        self.mochila = Mochila()
        def guardar_util(self, item):
            self.mochila.agregar_util(item)
    def ver_mochila(self):
        print(f"{self.nombre} lleva: {self.mochila.utiles}")
nico = Estudiante("Nicolás")
nico.guardar_util("Cuaderno")
nico.ver_mochila()
```

# Actividad práctica en parejas

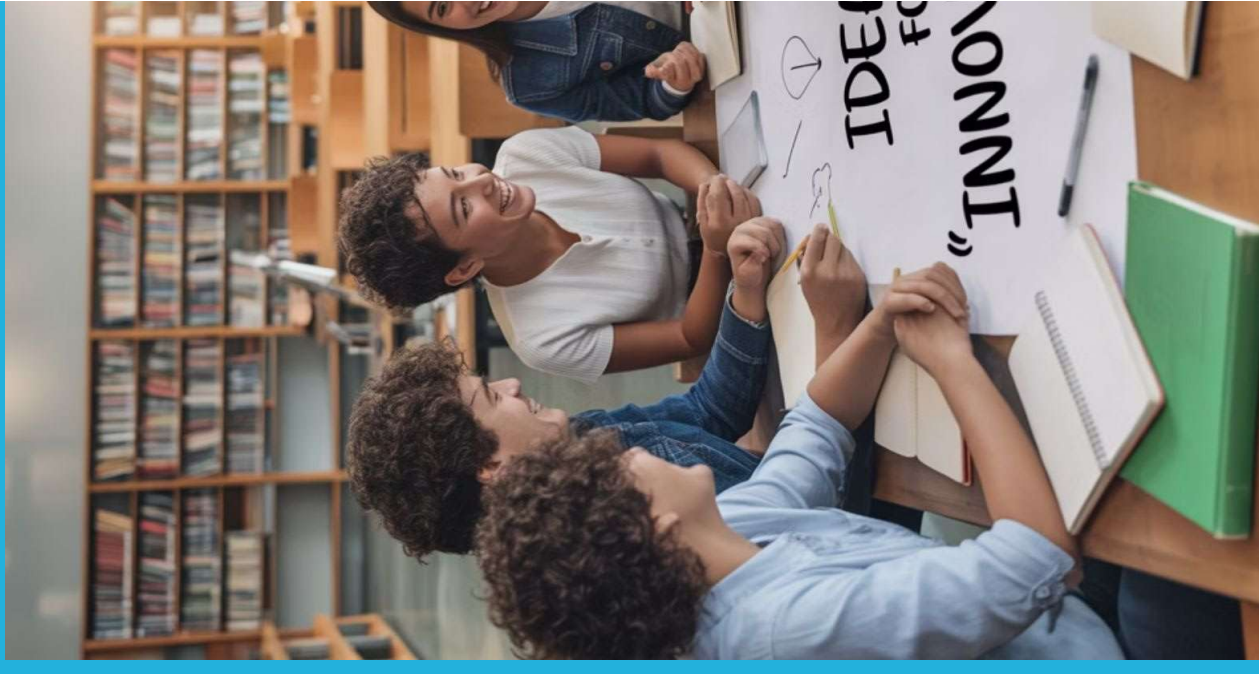
## Instrucción:

En parejas, creen una relación de composición entre dos clases.

Ejemplo sugerido:

- Profesor tiene un Computador
- Estudiante tiene un Estuche

**Objetivo:** Aplicar composición en su propio ejemplo.



CONNECTIONS



# Un colegio tiene estudiantes