Herencia

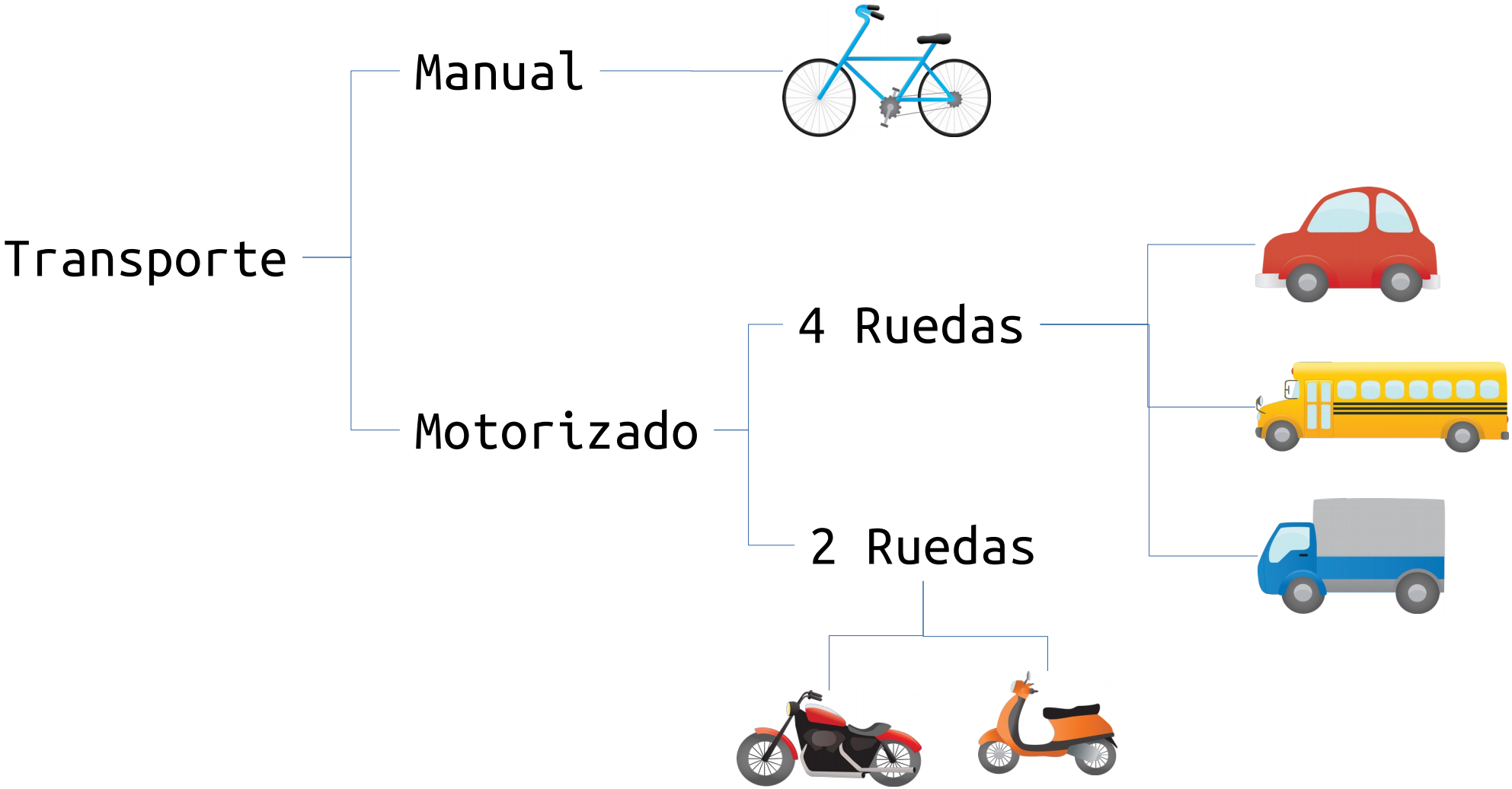
Herencia

- Una de las principales propiedades de las clases es la herencia.
- Esta propiedad nos permite crear nuevas clases a partir de clases existentes, conservando los métodos y atributos de la clase original y añadiendo otros nuevos.
- Esta propiedad nos permite encapsular diferentes partes de cualquier objeto real o imaginario, y vincularlo con objetos más elaborados del mismo tipo básico, que heredarán todas sus características.

Jerarquía

- Cada nueva clase obtenida mediante herencia se conoce como clase derivada, y las clases a partir de las cuales se deriva, clases base.
- Cada clase derivada puede usarse también como clase base para obtener una nueva clase derivada.
- Además cada clase derivada puede serlo de una o más clases base. En este último caso hablaremos de derivación múltiple.
- Esto nos permite crear una jerarquía de clases tan compleja como sea necesario.

Herencia



Definición de clases derivadas

- Para crear una clase derivada solo hay que indicarlo en la definición entre paréntesis:

```
class A:
    def print(self, a):
        print('OUT:' + str(a))
class B(A): ← clase derivada de A
    pass
x = B() ← Objeto de la clase A.
x.print(25) ← Podemos acceder a los métodos de la clase A.
```

Herencia de métodos.

- A diferencia de otros lenguajes en Python no existe la sobrecarga. Es decir, no pueden existir dos métodos que se llamen igual con diferentes características.
- Eso significa que los métodos que hereda la clase derivada son sobrescritos son redefinidos.

Herencia de métodos.

```
class A:
    def metodo(self):
        print('Método de A')
class B(A):
    def metodo(self):
        print('Método de B')
x = B()
x.metodo() → Método de B
```

Herencia de métodos.

- Sin embargo estos métodos sobrescritos no desaparecen. Podemos acceder a ellos dentro de la clase derivada a través de la función super().

Herencia de métodos.

```
class A:
    def metodo(self):
        print('Método de A')
class B(A):
    def metodo(self):
        super().metodo()
        print('Método de B')
x = B()
x.metodo() → Método de B
```

Herencia de métodos.

- Sin embargo estos métodos sobrescritos no desaparecen. Podemos acceder a ellos dentro de la clase derivada a través de la función super().
- Esto es muy útil cuando queremos llamar al constructor de la clase base (lo que nos puede ahorra mucho código e investigar qué hacía el constructor de la clase base).

Herencia de métodos.

```
class A:
    def __init__(self, a, b):
        self.a, self.b = a, b
class B(A):
    def __init__(self, a, b, c):
        self.c = c
        super().__init__(a, b)
x = B(1, 2, 3)
print(x.a, x.b, x.c) → 1 2 3
```

Herencia de métodos.

- Sin embargo estos métodos sobrescritos no desaparecen. Podemos acceder a ellos dentro de la clase derivada a través de la función super().
- Esto es muy útil cuando queremos llamar al constructor de la clase base (lo que nos puede ahorra mucho código e investigar qué hacía el constructor de la clase base).
- Con este mismo método podemos cambiar el comportamiento de clases predefinidas.

Herencia de métodos.

```
class DictCaseInsensitive (dict):

    def __getitem__(self, a, b):
        if isnstance(key, str):
            return super().__getitem__(key.lower())
        else:
            return super().__getitem__(key)

    def __setitem__(self, key, value):
        if isinstance(key, str):
            super().__setitem__(key.lower(), value)
        else:
            super().__setitem__(key, value)
```

```
x = DictCaseInsensitive()

x['Test'] = 10

x['test'] → 10

x['TEST'] → 10

x['tEsT'] → 10
```

Herencia múltiple.

- Una clase derivada puede tener varias clases base.

```
class A:
    def metodoA(self):
        print('Método A')
class B:
    def metodoB(self):
        print('Método B')
class C:
    def metodoC(self):
        print('Método C')
x = C()
x.metodoA()
x.metodoB()
x.metodoC()
```

Ejemplo herencia.

- Construiremos una clase que contenga la estructura de datos de una persona y sus derivadas:

  Estudiante (Persona)

  Empleado (Persona)

  Becario (Estudiante, Empleado)