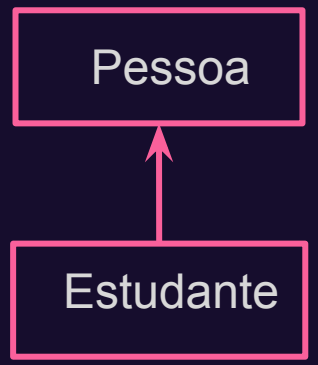
Orientação a Objetos com Python

Herança e *Mixins*

WOMAKERS CODE

Herança

- **Herança** é um conceito comum a todas as linguagens de programação que possuem orientação a objetos
- Com a herança, uma classe pode **herdar** os métodos e atributos de outra classe.
- A herança cria uma relação de uma classe **“é”** do mesmo tipo de outra. Por exemplo: um estudante é uma pessoa.
- A relação de “é” se opõe a outro tipo de relação, que é o **“tem”**. Por exemplo: um estudante tem livros.



Benefícios do uso da Herança

- A herança permite modelar relações do mundo real.
- **Herança** é mais uma ferramenta que permite o **reuso de código**, já que o programador não precisa escrever as mesmas funcionalidades várias vezes em classes diferentes.
- **Herança** permite adicionar novas funcionalidades sem modificar uma classe que já existe, o que pode ser feito criando uma classe **derivada** (ou classe **filha**).
- **Herança** funciona de forma **transitiva**, ou seja, se a classe B herda todas as funcionalidades da classe A, todas as classes que herdam de B também ganham automaticamente todas as funcionalidades da classe A.
- **Exemplo**: um professor é um trabalhador, que por sua vez é uma pessoa. Logo, a classe **Professor** vai herdar todos os métodos e propriedades de **Trabalhador**, que por sua vez também herda de **Pessoa**.
- Em Python, todas as classes herdam implicitamente da classe **object**, que fornece métodos comuns que podem ser sobrescritos como o **__init__**, o **__str__** e outros.≈

Herança Múltipla (*Mixins*)

- Uma classe pode herdar de múltiplas classes em Python.
- Essa funcionalidade não existe em algumas outras linguagens de programação orientadas a objetos (C#, Java) por ser controversa
- Herança múltipla pode fazer o código ficar muito mais complicado do que o necessário.
- O caso de uso mais legítimo é na criação de um framework.
- Ao trabalhar com Django, vocês podem ver casos onde uma classe vai herdar de duas ou mais classes.