Estrutura de dados
Orientação a Objetos

Profª. Drª. Alana Morais

Planejamento OO

1. Identifique uma classe que deve existir no seu problema
    a. Pense nos atributos
    b. Implemente o construtor
    c. Implemente os métodos da classe
    d. Implemente os métodos gets e sets / properties
    e. Implemente o def __str__(self)
2. Verifique se há outras classes
    a. Repita o processo
3. Implemente o arquivo teste.py

Problema Inicial

1- Implemente a classe **Aluno.class**

- Atributos: nome, matrícula, endereço, turma, sexo.

Problema Inicial

1- Implemente a classe **Aluno.class**

- Atributos: nome, matrícula, endereço, turma, sexo.
- Implemente o método:

matricularAlunoTurma("turma")

Problema Inicial

1- Implemente a classe **Aluno.class**

- Atributos: nome, matrícula, endereço, turma, sexo.
- Implemente o método:

matricularAlunoTurma("turma") .

2- Crie o arquivo **teste.py**

Problema Inicial

1- Implemente a classe **Aluno.class**

- Atributos: nome, matrícula, endereço, turma, sexo.
- Implemente o método:

matricularAlunoTurma("turma")

2- Crie o arquivo **teste.py**

- Adicione uma lista de alunos.

Problema Inicial

1- Implemente a classe **Aluno.class**

- Atributos: nome, matrícula, endereço, turma, sexo.
- Implemente o método:

matricularAlunoTurma("turma")

2- Crie o arquivo **teste.py**

- Adicione uma lista de alunos.
- Matricule cada aluno em uma turma.

Problema Inicial

1- Implemente a classe **Aluno.class**

- Atributos: nome, matrícula, endereço, turma, sexo.
- Implemente o método:

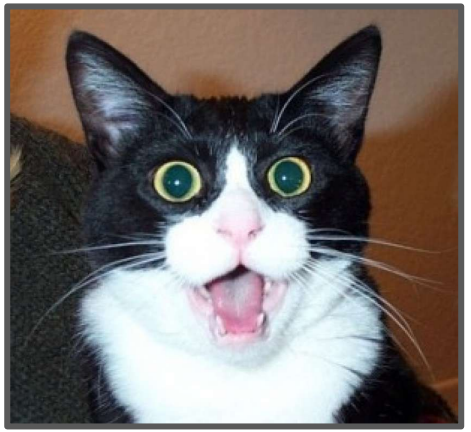
matricularAlunoTurma("turma")

2- Crie o arquivo **teste.py**

- Adicione uma lista de alunos.
- Matricule cada aluno em uma turma.
- Adicione um **aluno de intercâmbio** e imprima a **faculdade de origem** deste aluno.

O QUE DEVEMOS FAZER???

Que tal voltar para metodologia de Orientação a Objetos?
Use os fundamentos da **HERANÇA** em Python.



Herança?? Mas na minha cabeça isso é outra coisa?? Não vejo como usar isso em OO.

O que você lembra com a palavra herança??

- Dinheiro
- Receber algo
- Relacionamento entre as partes
- Posse
- Legado, domínio

Mas será que a Herança em Python não tem nenhuma relação com a herança no mundo real ?

O que você lembra com a palavra herança??

- ~~Dinheiro~~
- Receber algo
- Relacionamento entre as partes
- Posse
- Legado, domínio

Herança em java

- Herança é uma outra forma de **reutilizar** código.
  - Qual já conhecemos?

- Permite a criação de objetos que são parecidos, em comportamento, com outros objetos ancestrais.

- Em orientação a objetos, as relações de **generalização/especializaçã**o são implementadas por meio do mecanismo de herança.

Herança em Python 3

Conceito:

- Herança é a capacidade de especializar tipos de objetos (classes), de forma que os tipos especializados contenham, além de características estruturais e comportamentais já definidas pelos seus "ancestrais", outras definidas para eles próprios.

Forma:

Herança em Python

SuperClasse

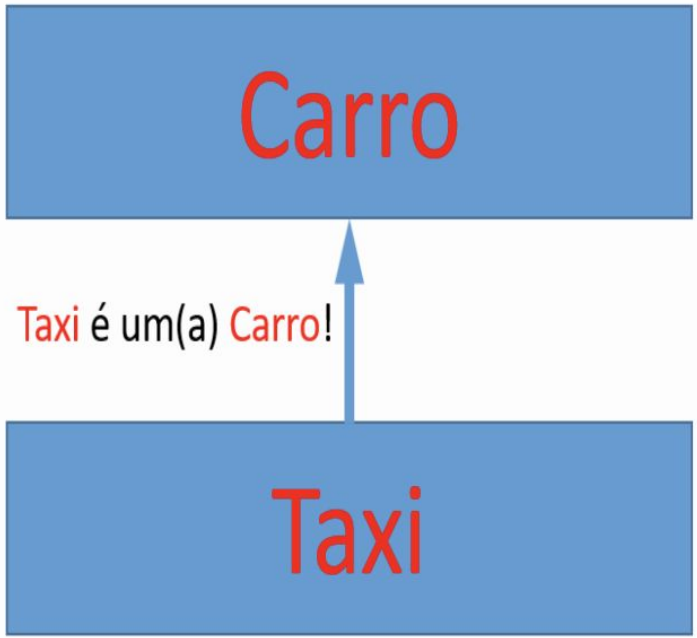
- Numa relação de herança, é a classe **mais genérica**, cuja estrutura e comportamento são herdados pela outra classe.
- Não referencia os filhos

SubClasse:

- Numa relação de herança, é a classe **mais específica**, que herda a estrutura e comportamento da outra classe.
- Classe que faz a incorporação a classe mais genérica.

Exemplo

Carro é uma superclasse, classe pai, classe base de Taxi!



Taxi é uma subclasse, classe filha, classe derivada de Carro!

Herança em Python 3

A herança pode ser feita:

- Com classes já construídas pelo próprio programador.
- Com classes de terceiros ou classes-padrão da linguagem Python.

Em Python, não há palavras-chave para identificar a herança.

- Em Java, utiliza-se a palavra extends, por exemplo.

Exemplo

```
class Carro():
    def __init__(self):
        print("Criou um carro")

    def getMotor(self):
        print("tenho um motor a combustao")
```

Exemplo

```
class Carro():
    def __init__(self):
        print("Criou um carro")

    def getMotor(self):
        print("tenho um motor a combustao")

class Taxi ( Carro ):
    def __init__(self):
        Carro.__init__(self) # Chamando o construtor da classe geral - Carro
        self.getMotor()
```

1o. Sinalizando quem é a classe pai.

Exemplo

```
class Carro():
    def __init__(self):
        print("Criou um carro")

    def getMotor(self):
        print("tenho um motor a combustao")

class Taxi ( Carro ):
    def __init__(self):
        Carro.__init__(self)
        self.getMotor()
```

2o. Chamando o construtor da classe pai no construtor da minha classe

Exemplo

```
class Carro():
    def __init__(self):
        print("Criou um carro")

    def getMotor(self):
        print("tenho um motor a combustao")

class Taxi ( Carro ):
    def __init__(self):
        super().__init__( )
        self.getMotor()
```

2o. Chamando o construtor da classe pai no construtor da minha classe

Exemplo

```
class Carro():
    def __init__(self):
        print("Criou um carro")

    def getMotor(self):
        print("tenho um motor a combustao")

class Taxi( Carro ):
    def __init__(self):
        super().__init__( ) # Chamando o construtor da classe geral - Carro
        self.getMotor()


car = Taxi()
```

Método super

Importante para o acesso aos métodos da classe pai

O super() é utilizado entre heranças de classes, ele nos proporciona extender/subscrever métodos de uma super classe (classe pai) para uma sub classe (classe filha)

Vamos voltar para o nosso exemplo dos Alunos

Sobrecarga de Métodos

- Imagine que agora você precisa matricular o aluno em uma turma e cadastrar por meio da sua cpf ou passaporte.
- É possível sobrescrever qualquer método da classe pai que não se adeque a classe filha.
- Para fazer isso é necessário definir o método na classe derivada com o mesmo nome do método indesejável.
- O Python irá desprezar o método da classe pai e irá só dar atenção ao método definido na classe filha.

Herança Múltipla

- Capacidade de uma classe ter mais de uma classe pai
- Dependendo da rede de classes e de como foram criadas, podem ocorrer comportamentos sem controle do programador.
- As melhores práticas ditam para evitar o uso de Herança Múltipla.

Conclusões

- A herança permite o reuso de códigos mesmo que alguns métodos necessitam ser implementados levemente diferentes nas classe filhas.
- O uso de herança aumenta o acoplamento entre as classes, isto é, o quanto uma classe depende de outra.
- Devemos ter cuidado ao usá-la.

Exercício - Sua vez ...

Implemente uma estrutura de classes que representam tipos de Programas em stream. De um lado temos Filmes (com nome, ano, duração e numeroLikes) e do outro temos as Séries (com nome, ano, temporadas, numeroLikes).

Todo filme precisa ter um método para informar se haverá pré-estreia ou não e sua data.

Toda série precisa de um método que informe se há e quais são os spinoffs dela.

Utilize herança, organize o projeto em pacotes e teste seu projeto.

Dúvidas?

alanamm.prof@gmail.com