

Python:

Orientação a Objetos



Prof. Bruno Gomes



@profbrunogomes



Aula de Hoje

2

- » **Introdução**
- » **Classe**
- » **Atributos**
- » **Métodos**
- » **Object**
- » **Relacionamento**
- » **Herança**



1.

Introdução

Introdução

4

- » **Programação Orientada a Objetos** É um Paradigma de Programação;
- » Fornece um mapeamento direto entre o mundo real e as unidades de organização utilizadas no projeto;
- » Diversas unidades de software, chamadas de objetos, que interagem entre si:
 - » Separa claramente a noção de o que é feito de como é feito.

Conceito de Objeto

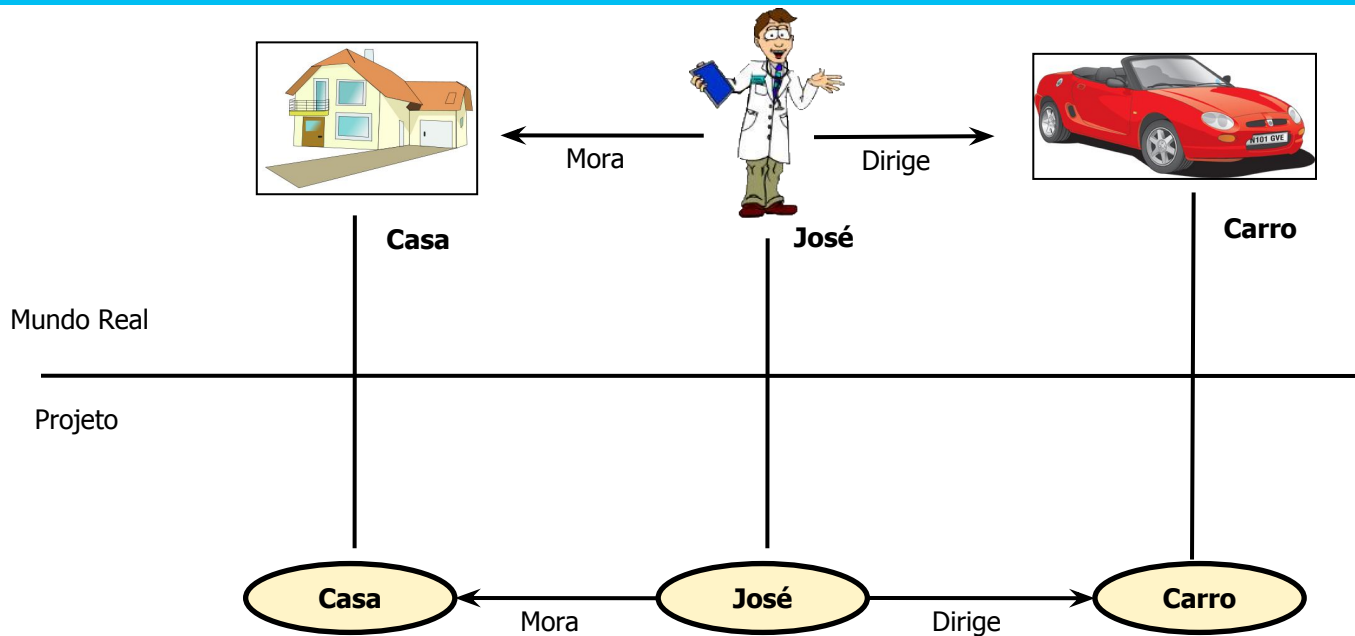
5

- » No mundo real, pensamos em conceitos e em entidades **concretas** e **abstratas**;
- » Tudo é objeto: carro, computador, música, camisa, cliente, conta bancária, etc.



Representação

6



» Vantagens:

- » Flexibilidade;
- » Reusabilidade;
- » Robustez;
- » Modularidade.

» Elementos básicos:

- » Objetos;
- » Classes;
- » Instâncias.



Dúvidas?



2.

Classe

Sintaxe - Criando uma Classe

10



```
class NomeDaClasse:
```

» Utiliza a palavra reservada class

Classe Vazia

11



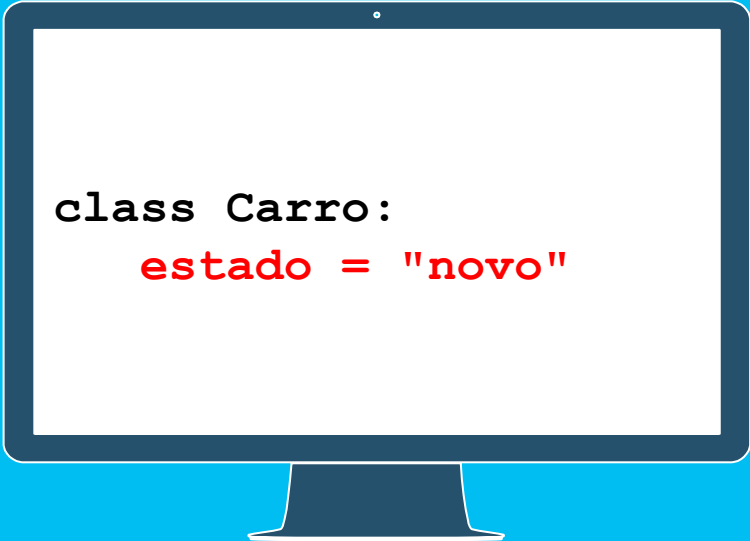
```
class NomeDaClasse:  
    pass
```

- » **Sintaxe mais simples:**
 - » **Classe vazia**
- » **Qualquer bloco vazio, deve ter um pass**
 - » **Classe, método.....**

Atributos da Classe

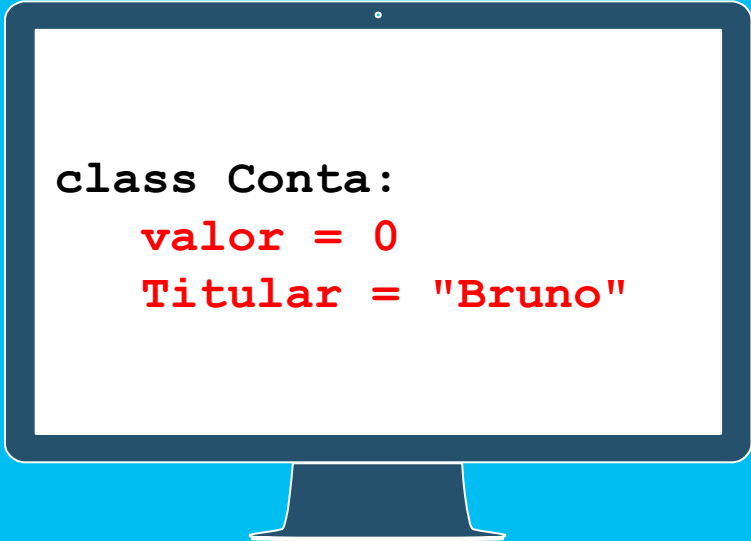
12

Exemplo 1



```
class Carro:  
    estado = "novo"
```

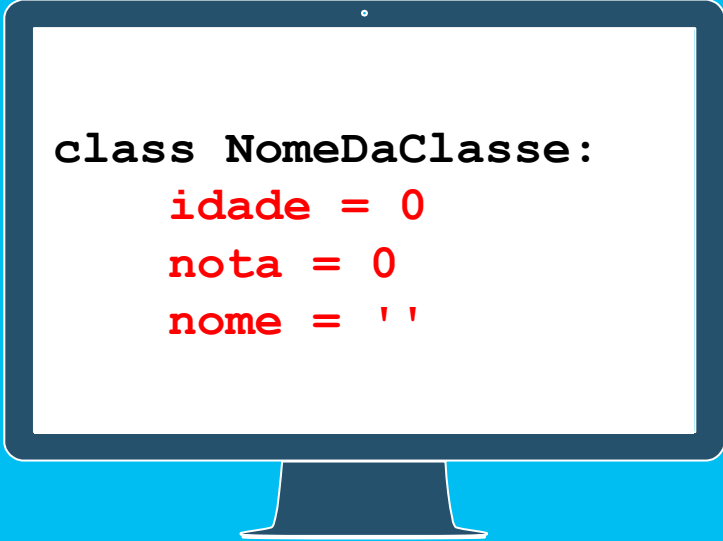
Exemplo 2



```
class Conta:  
    valor = 0  
    Titular = "Bruno"
```

Atributos da Classe

13

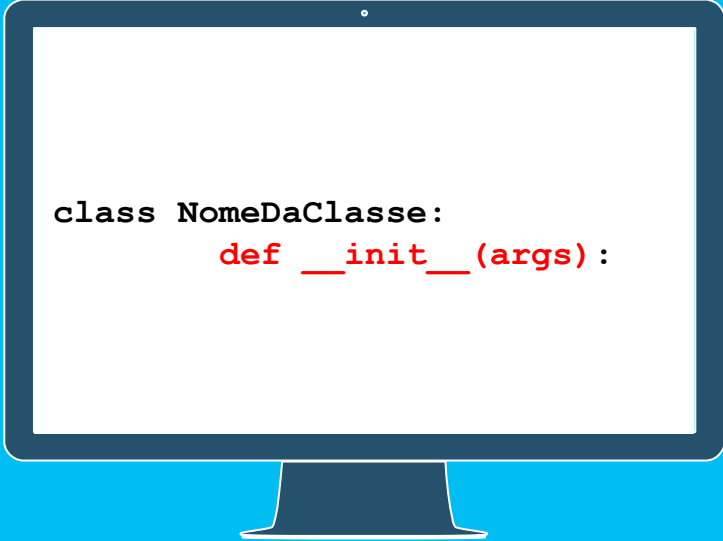


```
class NomeDaClasse:  
    idade = 0  
    nota = 0  
    nome = ''
```

- » Devem ser inicializados, mesmo que valores nulos.

Método de Inicialização

14



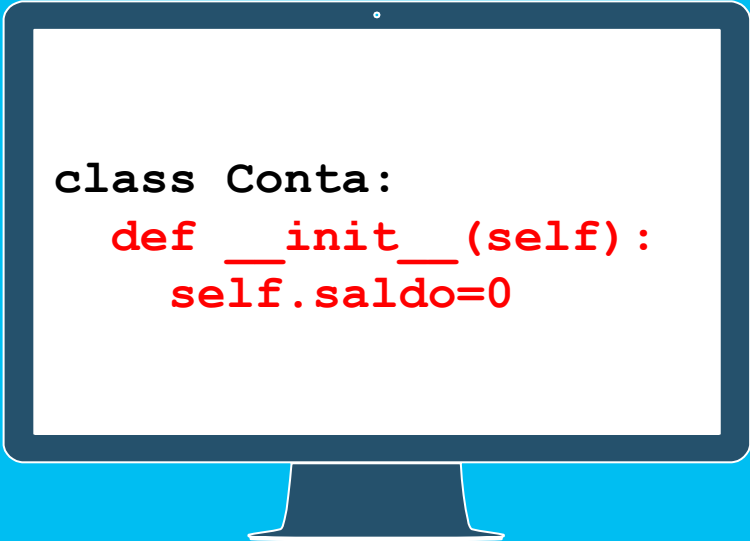
```
class NomeDaClasse:  
    def __init__(args):
```

- » __init__
 - » Método de Inicialização
 - » Conceito semelhante ao construtor em Java
- » Deve-se usar self como primeiro argumento de qualquer método da classe:
 - » É definido automaticamente para referenciar o objeto recém-criado que precisa ser inicializado
 - » self é apenas convenção, pode utilizar qualquer outro nome...

Criando uma Classe

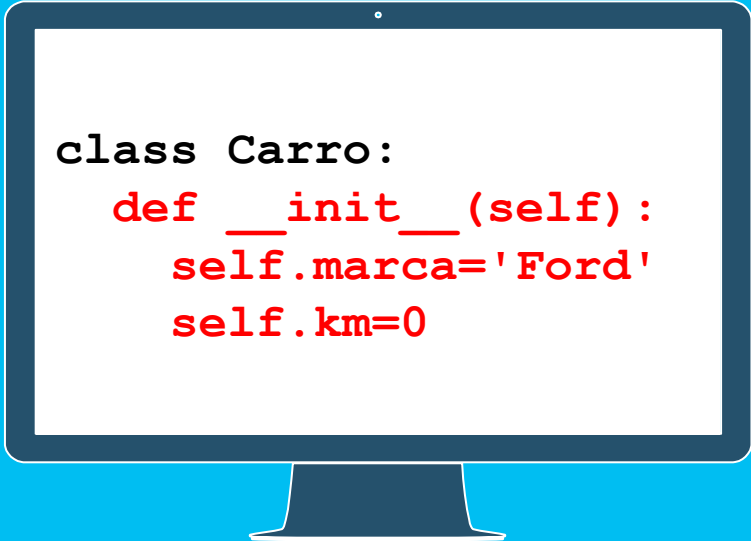
15

Exemplo 1



```
class Conta:  
    def __init__(self):  
        self.saldo=0
```

Exemplo 2



```
class Carro:  
    def __init__(self):  
        self.marca='Ford'  
        self.km=0
```

Dúvidas?



3.

Parâmetros

Criando uma Classe com parâmetros

18

Exemplo 1

```
class Conta:  
    def __init__(self, saldo):  
        self.saldo=saldo
```

Exemplo 2

```
class Carro:  
    def __init__(self, km):  
        self.marca='Ford'  
        self.km=km
```

Dúvidas?



4.

Métodos

Método

21



```
class NomeDaClasse:
```

```
    def nome(self, args):
```

- » Métodos são funções definidas dentro da classe
- » São criados da mesma forma que funções
- » Podem ou não retornar valores
- » Primeiro argumento deve ser self

Exemplo de Métodos

22

Exemplo 1

```
class Conta:
    def __init__(self, saldo):
        self.saldo=saldo

    def sacar(self, valor):
        self.saldo -= valor
```

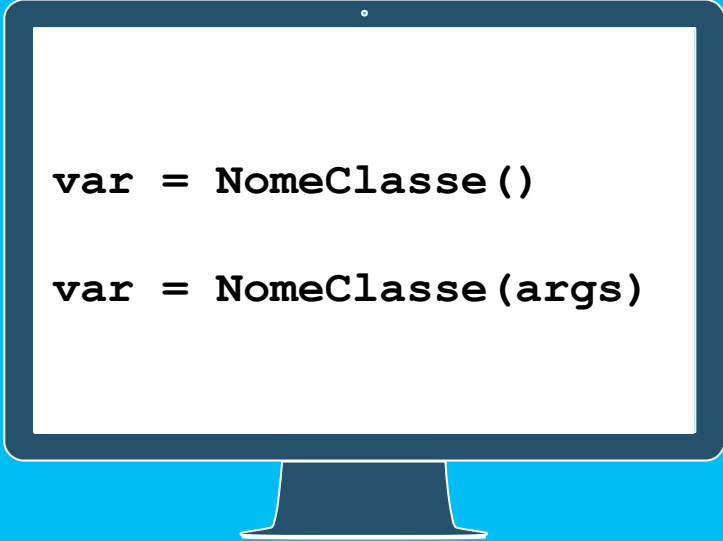
Exemplo 2

```
class Carro:
    def __init__(self):
        self.marca='Ford'
        self.km=0

    def consultar_marca(self):
        return self.marca
```

Criando o objeto

23



```
var = NomeClasse()  
  
var = NomeClasse(args)
```

- » O objeto é criado e salvo em uma variável.

Exemplos

24

```
class Conta(object):  
    def __init__(self, saldo):  
        self.saldo=saldo  
  
    def sacar(self, valor):  
        self.saldo -= valor
```

```
co = Conta(540)  
co.sacar(20)  
print(co.saldo)
```

```
class Carro:  
    def __init__(self, km):  
        self.marca='Ford'  
        self.km=km  
  
    def consultar_marca(self):  
        return self.marca
```

```
carrol = Carro(1000)  
print(carrol.consultar_marca())  
print(carrol.km)
```

Dúvidas?



Atividade

26

- » **Desenvolva um Algoritmo em Python que:**
 - » **Crie uma classe Calculadora;**
 - » **Crie 4 métodos, cada um recebe 2 atributos, e respectivamente retornam:**
 - » **Soma**
 - » **Subtração**
 - » **Multiplicação**
 - » **Divisão**

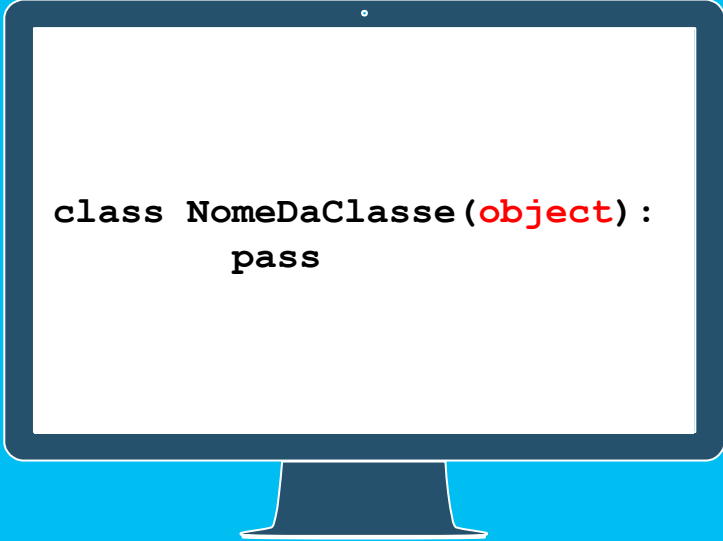


5.

Object

Criando uma Classe

28



```
class NomeDaClasse(object):  
    pass
```

- » Tudo em python deve ser explícito
- » Toda classe herda de Object
 - » Deve ser colocado na declaração da classe
 - » Opcional

Exemplos

29

```
class Conta(object):  
    def __init__(self, saldo):  
        self.saldo=saldo
```

```
    def sacar(self, valor):  
        self.saldo -= valor
```

```
co = Conta(540)  
co.sacar(20)  
print(co.saldo)
```

```
class Carro(object):  
    def __init__(self, km):  
        self.marca='Ford'  
        self.km=km
```

```
    def consultar_marca(self):  
        return self.marca
```

```
carrol = Carro(1000)  
print(carrol.consultar_marca())  
print(carrol.km)
```

Dúvidas?



6.

Relacionamento

Relacionamento

32

```
class Classe1(object):  
    def __init__(self, a):  
        self.a=a  
  
class Classe2(object):  
    def __init__(self, b, classe1):  
        self.b=b  
        self.classe1=classe1  
  
    def retornarA(self):  
        return self.classe1.a
```

- » Uma classe tem referência de outra classe
- » Como python é uma linguagem não tipada, podemos acessar qualquer método e atributo de outra classe (referenciada no código) supondo que ali dentro terá um objeto desta classe.

Exemplo

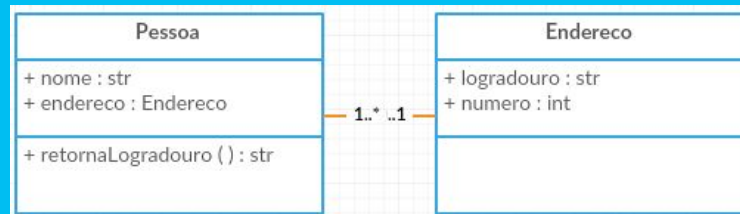
33

```
class Endereco(object):
    def __init__(self, logradouro, numero):
        self.logradouro = logradouro
        self.numero = numero

class Pessoa(object):
    def __init__(self, nome, endereco):
        self.nome = nome
        self.endereco = endereco

    def retornaLogradouro(self):
        return self.endereco.logradouro

e = Endereco("Rua X", "123")
p1 = Pessoa("Bruno", e)
print(p1.retornaLogradouro())
```



Dúvidas?

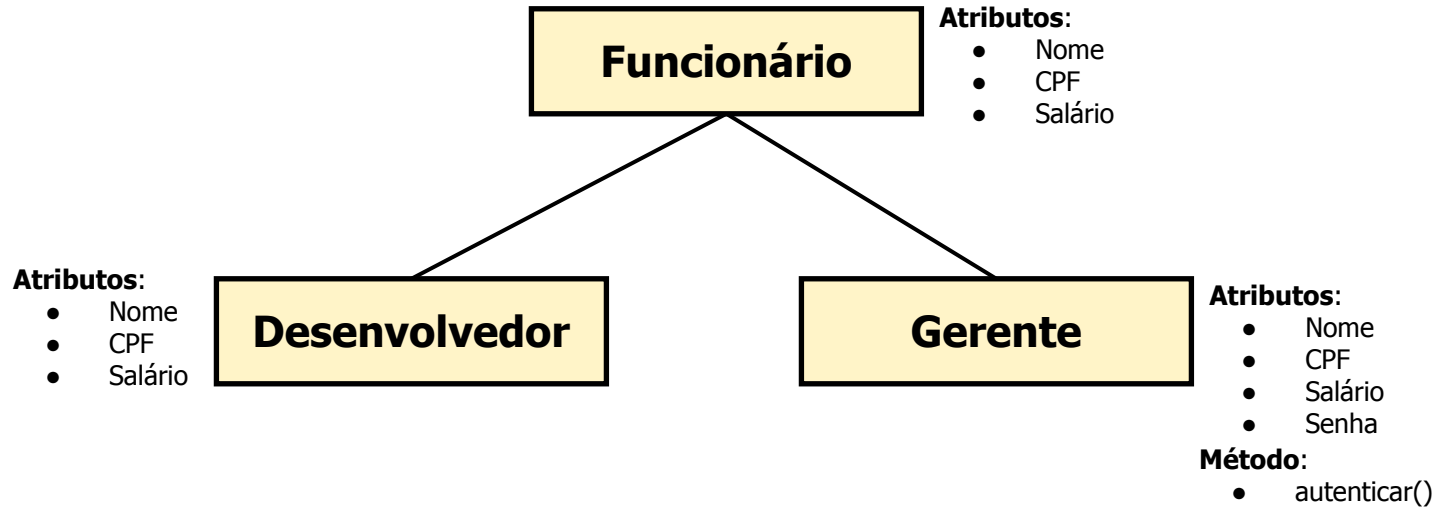


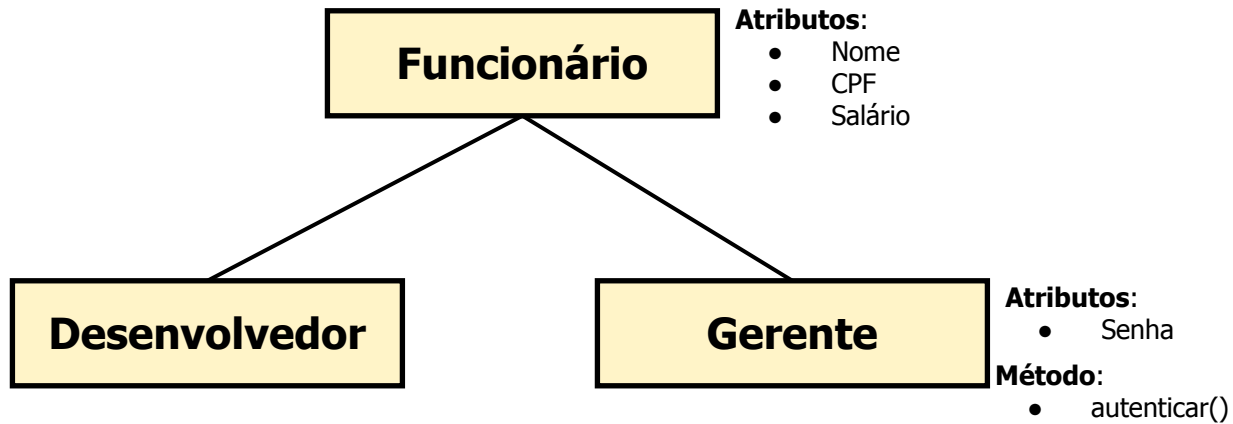
7.

Herança

Problema

36

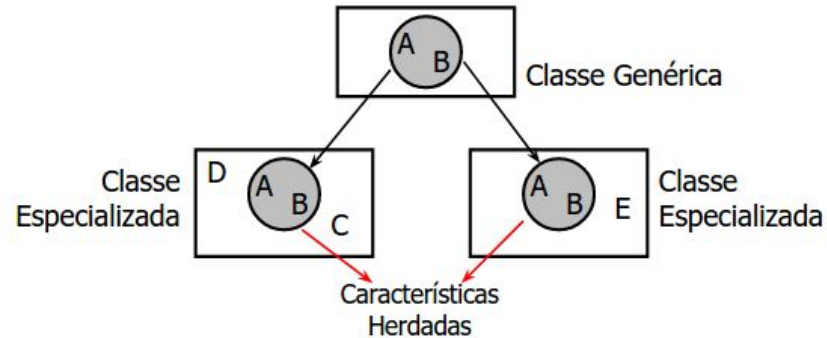




Herança

38

- » Estrutura Hierárquica e modular
- » Classes especializadas reutilizam o código das mais genéricas (herdam)



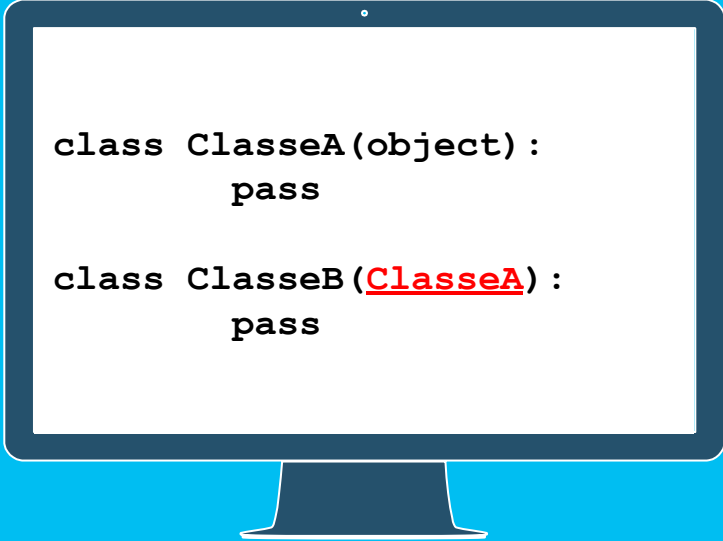
Na Herança:

39

- » Classe genérica, classe base, **superclasse** ou pai:
 - » Define variáveis de instância “genéricas” e métodos.
- » Classe especializada, derivada, **subclasse** ou filha:
 - » Especializa, **estende** ou herda os métodos “genéricos” de uma superclasse;
 - » Define apenas os métodos que são especializados.

Sintaxe da Herança

40

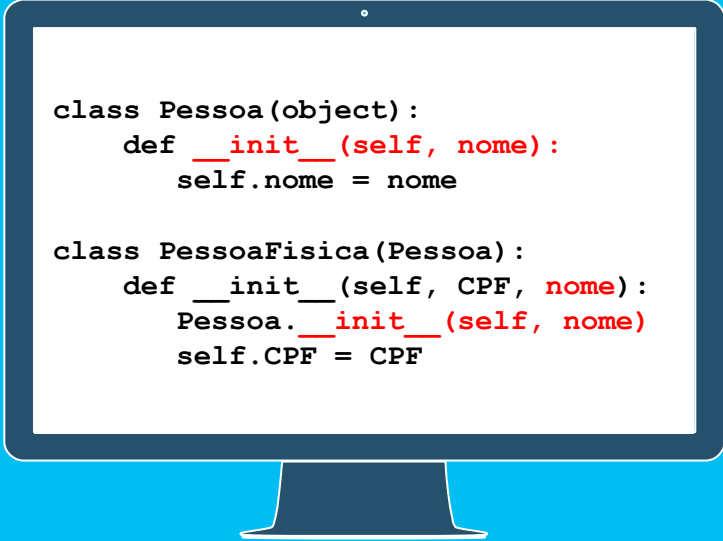


```
class ClasseA(object):  
    pass  
  
class ClasseB(ClasseA):  
    pass
```

- » A herança é informada nos () logo após o nome da classe
- » No exemplo ao lado:
 - » ClasseB herda de ClasseA

Método `__init__`

41



```
class Pessoa(object):  
    def __init__(self, nome):  
        self.nome = nome  
  
class PessoaFisica(Pessoa):  
    def __init__(self, CPF, nome):  
        Pessoa.__init__(self, nome)  
        self.CPF = CPF
```

- » O método `__init__` da classe pai deve ser acessado na classe filha e passado os argumentos.

Exemplo Completo

42

```
class Pessoa(object):
    def __init__(self, nome):
        self.nome = nome

    def getNome(self):
        return self.nome

class PessoaFisica(Pessoa):
    def __init__(self, CPF, nome):
        Pessoa.__init__(self, nome)
        self.CPF = CPF

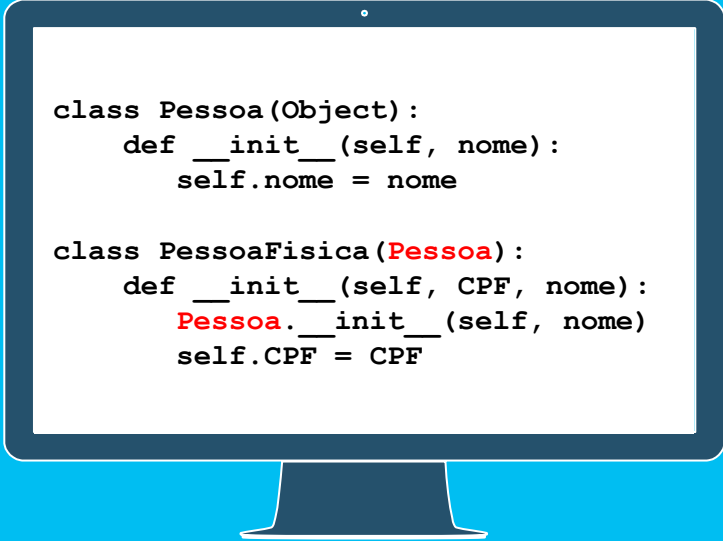
class PessoaJuridica(Pessoa):
    def __init__(self, CNPJ, nome):
        Pessoa.__init__(self, nome)
        self.CNPJ = CNPJ
```

```
pessoa1 = PessoaFisica("123", "Bruno")
print(pessoa1.getNome())
print(pessoa1.CPF)
```

```
pessoa2 = PessoaJuridica("54321", "IFRN")
print(pessoa2.CNPJ)
```

Observação

43

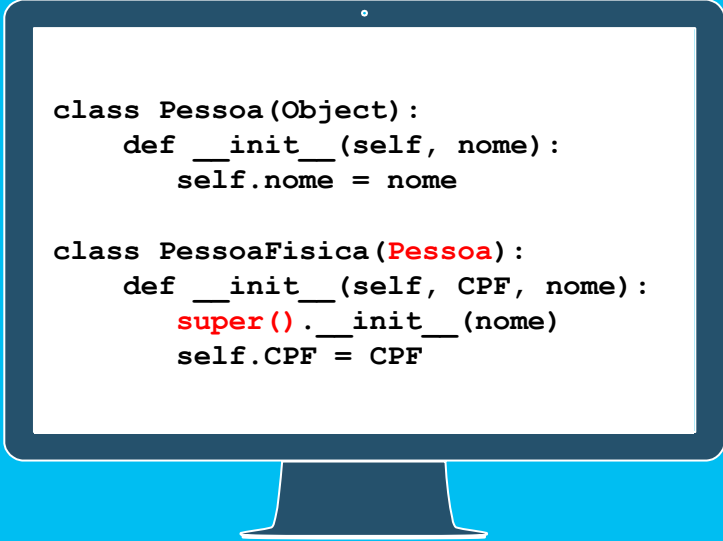


```
class Pessoa(Object):  
    def __init__(self, nome):  
        self.nome = nome  
  
class PessoaFisica(Pessoa):  
    def __init__(self, CPF, nome):  
        Pessoa.__init__(self, nome)  
        self.CPF = CPF
```

- » Se a classe pai mudar, teríamos que mudar também a chamada do `__init__`:
 - » `Pessoa.__init__(self, nome)`

Solução

44



```
class Pessoa(Object):  
    def __init__(self, nome):  
        self.nome = nome  
  
class PessoaFisica(Pessoa):  
    def __init__(self, CPF, nome):  
        super().__init__(nome)  
        self.CPF = CPF
```

- » Utilizar o `super()`
 - » Retirar o `self` dos parâmetros

Exemplo

45

```
class Pessoa:
    def __init__(self, nome):
        self.nome = nome

    def getNome(self):
        return self.nome

class PessoaFisica(Pessoa):
    def __init__(self, CPF, nome):
        super().__init__(nome)
        self.CPF = CPF

class PessoaJuridica(Pessoa):
    def __init__(self, CNPJ, nome):
        super().__init__(nome)
        self.CNPJ = CNPJ
```

```
pessoa1 = PessoaFisica("123", "Bruno")
print(pessoa1.getNome())
print(pessoa1.CPF)
```

```
pessoa2 = PessoaJuridica("54321", "IFRN")
print(pessoa2.CNPJ)
```