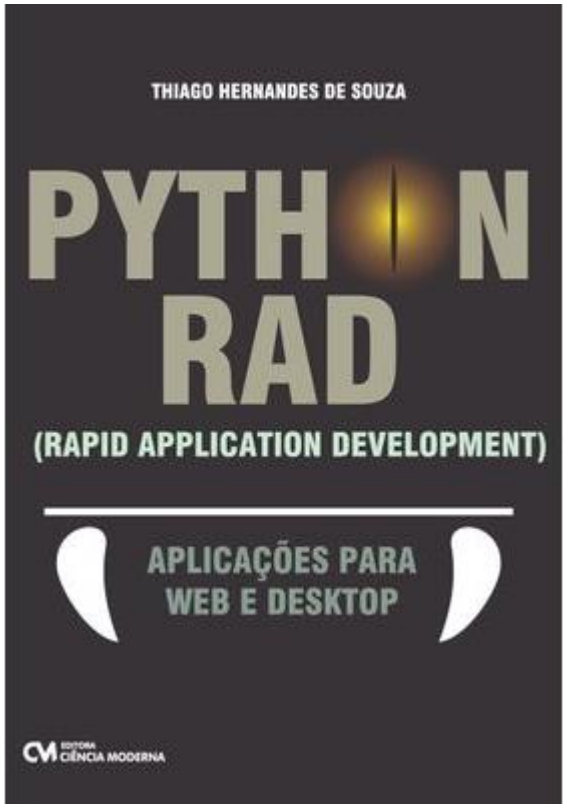
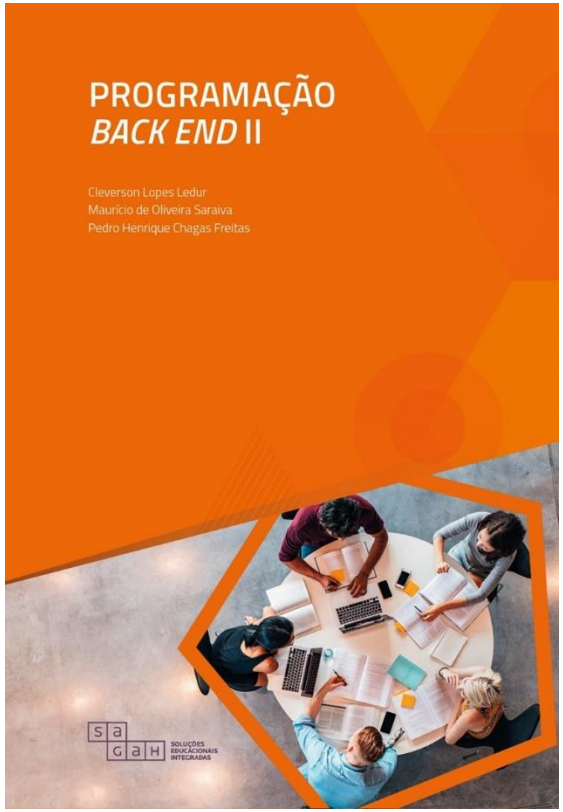
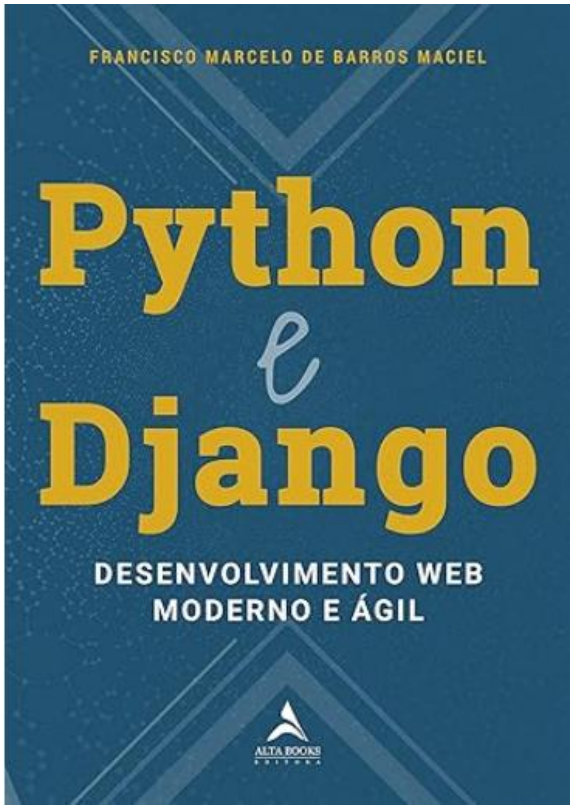
PROGRAMAÇÃO BACK-END







Programação Back-End

Python

Herança

Herança Python

- A herança nos permite definir uma classe que herda todos os métodos e propriedades de outra classe.
- Classe pai é a classe da qual está sendo herdada, também chamada de **classe base**.
- Classe filha é a classe que herda de outra classe, também chamada de classe derivada.

Classe pai

Qualquer classe pode ser uma classe pai, então a sintaxe é a mesma da criação de qualquer outra classe.

A seguir é mostrado um exemplo de uma classe chamada Pessoa, com propriedades nome e sobrenome e um método imprimenome

PROGRAMAÇÃO BACK-END

Exemplo

teste01.py > ...

```python
class Pessoa:
    def __init__(self, fnome, fsobrenome):
        self.nome = fnome
        self.sobrenome = fsobrenome

    def imprimenome(self):
        print(self.nome, self.sobrenome)


p1 = Pessoa("Carlos", "Henrique")
p1.imprimenome()
```

Classe filha

Para criar uma classe que herda a funcionalidade de outra classe, envie a classe pai como parâmetro ao criar a classe filha.

Crie uma classe chamada Estudante, que herdará as propriedades e métodos da classe Pessoa

```
 9   class Estudante(Pessoa):
10       pass
```

Nota: Use a pass palavra-chave quando não desejar adicionar outras propriedades ou métodos à classe.

Exemplo completo

teste02.py > ...

```python
class Pessoa:
    def __init__(self, fnome, fsobrenome):
        self.nome = fnome
        self.sobrenome = fsobrenome

    def imprimenome(self):
        print(self.nome, self.sobrenome)

class Estudante(Pessoa):
    pass

p1 = Pessoa("Carlos", "Henrique")
p1.imprimenome()

p2 = Estudante("Jose", "Candido")
p2.imprimenome()
```

Use a classe Estudante para criar um objeto e depois execute o método imprimenome.

PROGRAMAÇÃO BACK-END

Adicione a função __init__()

Até agora criamos uma classe filha que herda as propriedades e métodos de sua classe pai.

Queremos adicionar a função __init__() à classe filha (em vez da palavra-chave pass).

```
9   class Estudante(Pessoa):
10      def __init__(self, fnome, lsobrenome):
11          # Coloque aqui as propriedades e metodos
```

Quando a função __init__() é adicionado, a classe filha não herdará mais a função __init__() do pai.

Para manter a herança da função __init__() pai, adicione uma chamada à função __init__() pai.

```
 9   class Estudante(Pessoa):
10       def __init__(self, fnome, fsobrenome):
11           Pessoa.__init__(self, fnome, fsobrenome)
```

PROGRAMAÇÃO BACK-END

Função super()

Python também possui uma função super() que fará com que a classe filha herde todos os métodos e propriedades de sua classe pai.

teste03.py > ...

```python
class Pessoa:
    def __init__(self, fnome, fsobrenome):
        self.nome = fnome
        self.sobrenome = fsobrenome

    def imprimenome(self):
        print(self.nome, self.sobrenome)

class Estudante(Pessoa):
    def __init__(self, fnome, fsobrenome):
        super().__init__(fnome, fsobrenome)

p1 = Pessoa("Carlos", "Henrique")
p1.imprimenome()

p2 = Estudante("Jose", "Candido")
p2.imprimenome()
```

Adicionar uma propriedade a Classe Estudante

Adicione uma propriedade chamada anograduacao à classe Estudante

```
 9   class Estudante(Pessoa):
10       def __init__(self, fnome, fsobrenome):
11           super().__init__(fnome, fsobrenome)
12           self.anograduacao = 2024
```

PROGRAMAÇÃO BACK-END

Exemplo completo

teste04.py > ...

```python
class Pessoa:
    def __init__(self, fnome, fsobrenome):
        self.nome = fnome
        self.sobrenome = fsobrenome

    def imprimenome(self):
        print(self.nome, self.sobrenome)

class Estudante(Pessoa):
    def __init__(self, fnome, fsobrenome, fanograduacao):
        super().__init__(fnome, fsobrenome)
        self.anograduacao = fanograduacao

p1 = Pessoa("Carlos", "Henrique")
p1.imprimenome()

p2 = Estudante("Jose", "Candido", 2024)
p2.imprimenome()
```

Adicionando métodos

Pode-se adicionar métodos à classe Estudante.

```
 9  class Estudante(Pessoa):
10      def __init__(self, fnome, fsobrenome, fanograduacao):
11          super().__init__(fnome, fsobrenome)
12          self.anograduacao = fanograduacao
13
14      def mensagem(self):
15          print("Ola", self.nome, self.sobrenome, " do ano de ", self.anograduacao)
16
```

No exemplo está sendo criado o método mensagem.

PROGRAMAÇÃO BACK-END

Exemplo completo

teste05.py > ...

```python
class Pessoa:
  def __init__(self, fnome, fsobrenome):
    self.nome = fnome
    self.sobrenome = fsobrenome

  def imprimenome(self):
    print(self.nome, self.sobrenome)

class Estudante(Pessoa):
    def __init__(self, fnome, fsobrenome, fanograduacao):
        super().__init__(fnome, fsobrenome)
        self.anograduacao = fanograduacao

    def mensagem(self):
        print("Ola", self.nome, self.sobrenome, " do ano de ", self.anograduacao)

p1 = Pessoa("Carlos", "Henrique")
p1.imprimenome()

p2 = Estudante("Jose", "Candido", 2024)
p2.mensagem()
```