



Curso: Programação em Python

UC3: Desenvolver Aplicações Back-End com Python

Indicadores:

- Cria estruturas de código utilizando linguagem de programação para *back-end*, de acordo com os requisitos do projeto de software.
-

Conhecimentos:

- Linguagem de programação para *back-end* e orientação a objetos: sintaxe, definições e funcionalidades.
-




Orientação a Objetos

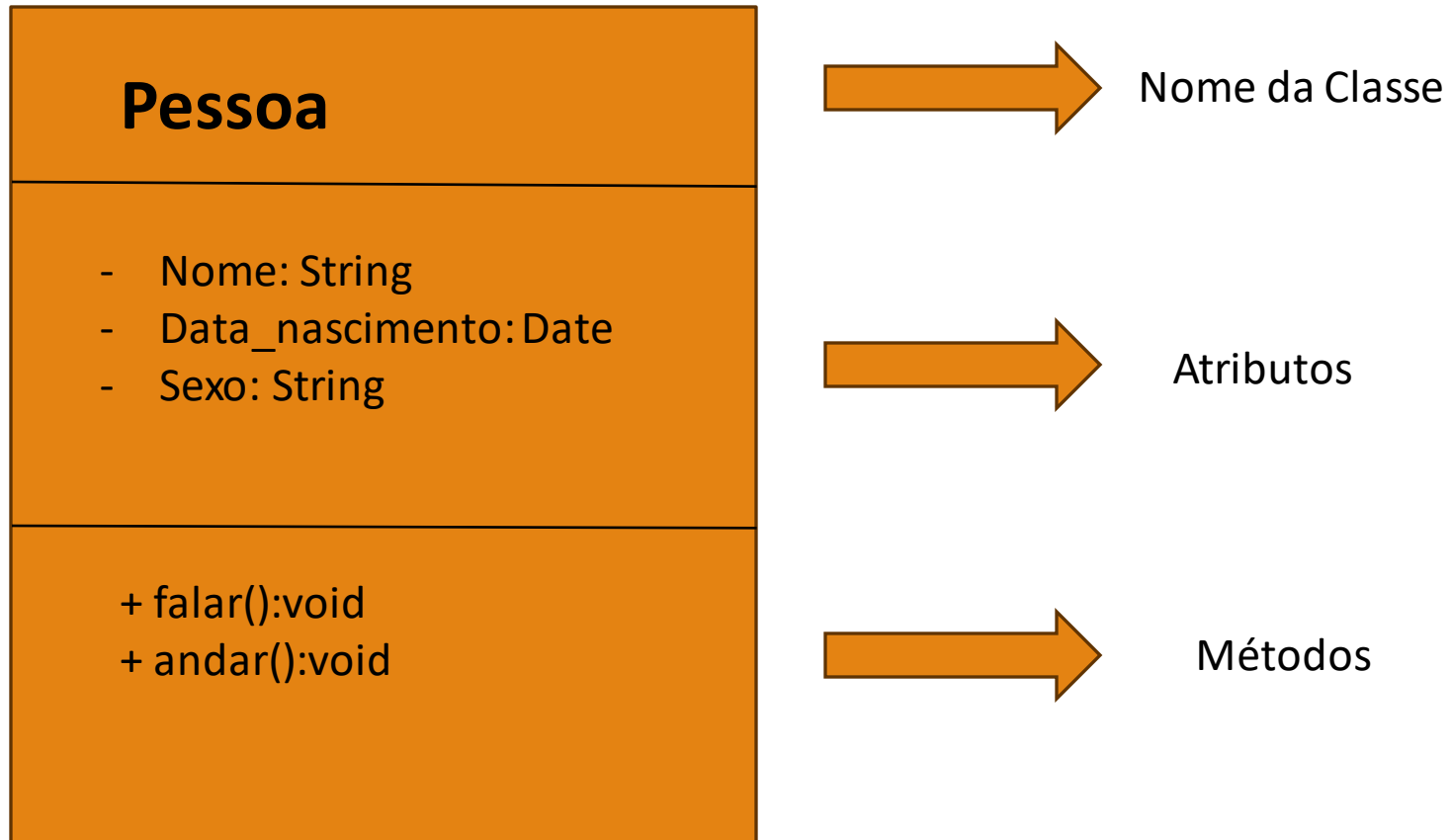
Trabalhar com orientação a objetos no Python envolve a criação e manipulação de classes, objetos e seus atributos e métodos.



Definições

Classe  é uma estrutura fundamental na programação orientada a objetos (POO). Ela é uma representação abstrata de um conceito ou entidade do mundo real, que encapsula atributos (dados) e comportamentos (métodos) relacionados a essa entidade.

Representação



Exemplificando

Suponhamos uma classe chamada Pessoa. Pessoa possui características, tais como: nome, data de nascimento, sexo, etc.

Pessoas também possuem comportamentos como: falar, andar, etc.

Dessa forma: Nome, data de nascimento e sexo são **atributos** da classe Pessoa. Enquanto que falar e andar são **métodos** da classe Pessoa.

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

    def falar(self, mensagem):
        print(f"{self.nome} diz: {mensagem}")
```





Onde:

`class` --> Palavra chave para definir uma nova classe.

`__init__` --> é um construtor especial que é chamado quando um objeto dessa classe é criado. Esse método inicializa os atributos da classe.

`self` --> é uma referência ao próprio objeto.

Objeto



Um objeto é uma instância de uma classe. Ele representa uma entidade específica do mundo real ou do domínio do problema que está sendo modelado.

Um objeto é criado a partir de uma classe e tem acesso aos atributos e métodos definidos na classe.



Criando um Objeto no Python

```
pessoa1 = Pessoa("José", "20/10/1995", "masculino")  
pessoa2 = Pessoa("Maria", "15/01/2001", "feminino")
```

Acessando os Atributos da Classe

```
print("Nome: ", pessoa1.nome, "Data de Nascimento:", pessoa1.data_nascimento)
```

```
print("Nome: ", pessoa2.nome, "Data de Nascimento:", pessoa2.data_nascimento)
```



Acessando os Métodos da Classe

```
pessoa1.falar("Olá!")  
pessoa2.falar("Oi!")
```

Ao acessar um método de uma classe, estamos chamando uma ação que é executada pela classe. Nessa caso a ação desempenhada pelo método falar será mostrar o que cada objeto enviou como mensagem ao método.

Momento Mão na massa.

Agora é sua vez. Lembra do nosso projeto da UC2?

1) Implemente uma classe para produtos com os seguintes atributos:

Codproduto / Nome / Descricao / Tamanho / Preço

2) Crie 3 objetos para a classe produto.

3) Crie o método desconto para o produto: Toda vez que o método desconto for acessado o preço do produto será exibido com 10% de desconto.

Herança



é um mecanismo que permite criar uma nova classe (chamada de classe derivada ou subclasse) com base em uma classe existente (chamada de classe base ou superclasse). A subclasse herda os atributos e métodos da superclasse e pode adicionar novos atributos e métodos ou modificar os existentes. A herança ajuda a promover a reutilização de código e permite modelar relacionamentos hierárquicos entre classes.



```
class Aluno(Pessoa):  
    def __init__(self, nome, data_nascimento, sexo,  
matricula):  
        super().__init__(nome, data_nascimento, sexo)  
        self.matricula = matricula
```



```
# Criando um objeto da classe Aluno  
aluno1 = Aluno("João silva", "10/01/2005", "Masculino",  
"2021001")
```

```
# Acessando os atributos do objeto aluno1  
print(aluno1.nome)  
print(aluno1.data_nascimento)  
print(aluno1.sexo)  
print(aluno1.matricula)
```