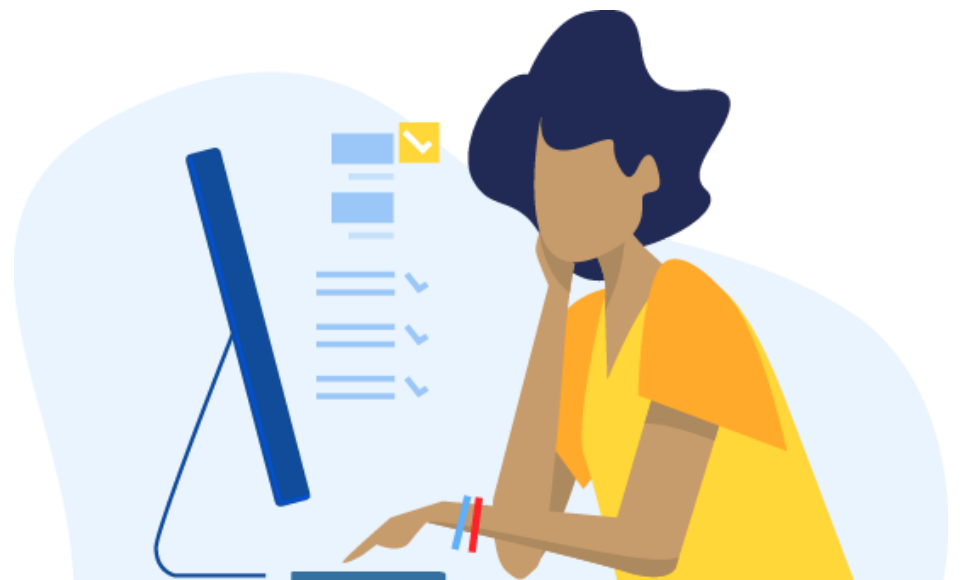


Classes e métodos em Python

O desenvolvimento de software orientado a objetos (OO) existe desde o início dos anos 1960, mas foi somente em meados da década de 1990 que o paradigma orientado a objetos começou a ganhar impulso. Uma linguagem é entendida como orientada a objetos se ela aplica o conceito de abstração e suporta a implementação do encapsulamento, da herança e do polimorfismo.



Fonte: Shutterstock.

Definições importantes

Antes de aprendermos como criar uma classe em Python vamos conhecer os conceitos de objeto, classe e instância.

» Objetos

São os componentes de um programa OO. Um programa que usa a tecnologia OO é basicamente uma coleção de objetos.

» Classe

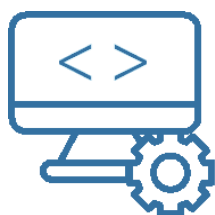
Uma **classe** é um modelo para um objeto.

Segundo a *Python Software Foundation* (PSF, 2020a), podemos considerar uma classe como uma forma de organizar os dados (de um objeto) e seus comportamentos.

» Instância

Entende-se por **instância** a existência física, em memória, do objeto.

Saiba mais



Vamos pensar na construção de uma casa: antes do "objeto casa" existir, um arquiteto fez a planta dela, determinando tudo que deveria fazer parte daquele objeto. Portanto, a classe é o modelo e o objeto é uma instância.

Como criar uma classe em Python

Atributos

Para criar uma classe em Python é necessária a sintaxe a seguir. Utiliza-se a palavra reservada "class" para indicar a criação de uma classe, seguida do nome e de dois pontos. No bloco indentado devem ser implementados os atributos e métodos da classe.

```
1 class ClassName:
2     < statement-1 >
3     .
4     .
5     .
6     < statement-N >
```

Exemplo de criação de uma classe

```
1 class PrimeiraClasse:
2     nome = None
3
4     def imprimir_mensagem(self):
5         print("Olá seja bem vindo!")
```

Após criada uma classe, os objetos podem ser instanciados, sendo importante lembrar que uma classe determina um tipo de estrutura de dados. Os atributos e os métodos de uma classe podem ser acessados pelo objeto, colocando o nome deste seguido de ponto; por exemplo: objeto.atributo.

A seguir apresentamos um exemplo.

```
1 objeto1 = PrimeiraClasse()
2 objeto1.nome = "Aluno 1"
3
4 print(objeto1.nome)
5 objeto1.imprimir_mensagem()
```

Os dados armazenados em um objeto representam o estado do objeto. Na terminologia de programação OO, esses dados são chamados de **atributos**. Os atributos contêm as informações que diferenciam os vários objetos – neste caso, os funcionários.



Fonte: Shutterstock.

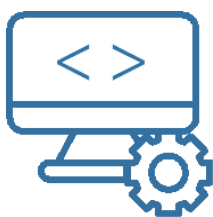
Métodos

O comportamento de um objeto representa o que o objeto pode fazer. Nas linguagens procedurais, o comportamento é definido por:

- » Procedimentos
- » Funções
- » Sub-rotinas.

Na terminologia de programação OO, esses comportamentos estão contidos nos **métodos**, e você invoca um método enviando uma mensagem para ele.

Saiba mais



A combinação dos atributos e métodos na mesma entidade, na linguagem OO, é chamada de **encapsulamento**. Alguns autores também consideram como encapsulamento a prática de tornar atributos privados, encapsulando-os em métodos para guardar e acessar seus valores.

Atenção

Atributos: variáveis de classe e de instância



Os atributos de uma classe podem ser variáveis de instância ou da classe. Uma variável de instância significa que, para cada objeto, é guardado um valor diferente; já as variáveis de classe são comuns a todas as instâncias de uma classe.

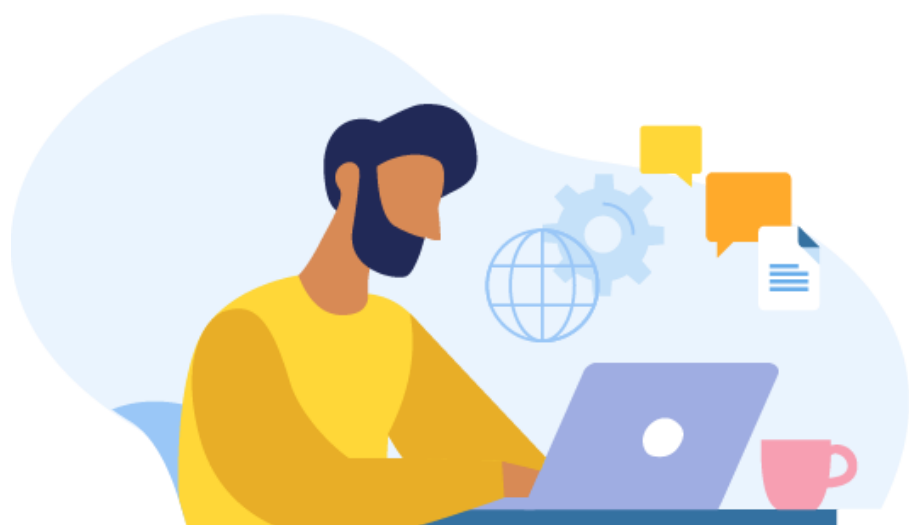
As variáveis de instâncias podem ter seus valores inicializados no momento da construção da classe, por meio do método construtor de classe.

Construtor da classe – `__init__()`

Ao instanciar um novo objeto, é possível determinar um estado inicial para variáveis de instâncias (atributos) por meio do método construtor da classe.

Em Python, o método construtor é chamado de `__init__()` e deve ser usado conforme o código a seguir. Na classe *FuncionarioTecnico*, o atributo `status`, que é uma variável de instância, recebe o valor no momento da criação do objeto, pois está no construtor.

```
1  class FuncionarioTecnico:
2  def __init__(self, status):
3      self.status = status
4
5  nivel = 'Técnico'
6  func1 = FuncionarioTecnico('Ativo')
7  func2 = FuncionarioTecnico('Licença Mestrado')
8
9  print(func1.nivel)
10 print(func2.nivel)
11 print(func1.status)
12 print(func2.status)
```



Fonte: Shutterstock.

Todo método em uma classe deve receber como primeiro parâmetro uma variável que indica a referência à classe – por convenção, adota-se o parâmetro **self**. O parâmetro **self** será usado para acessar os atributos e métodos dentro da própria classe.

Toda variável de instância possui o prefixo **self**, pois é dessa forma que é identificado que o atributo faz parte de um objeto específico.

Para se utilizar um método, dentro da classe, também é necessário utilizar o prefixo **self**.

Pesquise mais

No Capítulo 8 (*Programação orientada a objetos*) do livro referenciado a seguir, você encontrará a explicação sobre herança e sobrescrita de método em Python. Faça a leitura das páginas 279 a 284.

LJUBOMIR, P. **Introdução à computação usando Python**: um foco no desenvolvimento de aplicações. Rio de Janeiro: LTC, 2016.



Fonte: Shutterstock.