

Introdução a Programação Orientação a Objetos I

Código da turma no classroom: iqmtxhd

Aulas: Segunda - 18:30 e Quinta - 18:30

O QUE TEREMOS NA AULA DE HOJE!



TUDO É OBJETO

Python é uma linguagem totalmente orientada a objetos. Tudo em Python é um objeto! Sempre que utilizamos uma função ou método que recebe parâmetros, estamos passando objetos como argumentos.

Um sistema orientado a objetos é um grande conjunto de classes que vai se comunicar, delegando responsabilidades para quem for mais apto a realizar determinada tarefa.

Dizemos que esses objetos colaboram, trocando mensagens entre si. Por isso, acabamos tendo muitas classes em nosso sistema, e elas costumam ter um tamanho relativamente curto.

COMPOSIÇÃO

Quando a existência de uma classe depende de outra classe, como é a relação da classe Telefone com a classe Pessoa, dizemos que a classe Telefone compõe a classe Pessoa. Esta associação chamamos Composição.

MODIFICADORES DE ACESSO

O Python não utiliza o termo **private**, que é um **modificador de acesso** e também chamado de **modificador de visibilidade**. No Python, inserimos dois underscores ('__') ao atributo para adicionarmos esta característica. Dessa maneira, não conseguimos acessar o atributo de um objeto fora da classe original.

Um **modificador de acesso** define que os atributos não podem ser acessados de fora da classe

O prefixo com apenas um underscore ('_') não tem significado para o interpretador quando usado em nome de atributos, mas entre programadores Python é uma convenção que deve ser respeitada. O programador alerta que esse atributo não deve ser acessado diretamente.

Um atributo com apenas um underscore é chamado de protegido, mas quando usado **sinaliza** que deve ser tratado como um atributo "privado", fazendo com que acessá-lo diretamente possa ser perigoso.

As mesmas regras de acesso aos atributos
valem para os métodos.

É muito comum, e faz todo sentido, que seus atributos sejam privados e quase todos seus métodos sejam públicos (não é uma regra!). Desta forma, toda conversa de um objeto com outro é feita por troca de mensagens, isto é, acessando seus métodos.

ENCAPSULAMENTO

Encapsular é fundamental para que seu sistema seja suscetível a mudanças: não precisamos mudar uma regra de negócio em vários lugares, mas sim em apenas um único lugar, já que essa regra está encapsulada. O conjunto de métodos públicos de uma classe é também chamado de **interface da classe**, pois esta é a única maneira a qual você se comunica com objetos dessa classe.

O underscore _ alerta que ninguém deve modificar, nem mesmo ler, o atributo em questão.

Precisamos então arranjar uma maneira de fazer esse acesso. Sempre que precisamos arrumar uma maneira de fazer alguma coisa com um objeto, utilizamos métodos!

Para permitir o acesso aos atributos (já que eles são 'protegidos') de uma maneira controlada, a prática mais comum é criar dois métodos, um que retorna o valor e outro que muda o valor.

A convenção para esses métodos em muitas linguagens orientadas a objetos é colocar a palavra **get** ou **set** antes do nome do atributo.

Getters e *setters* são usados em muitas linguagens de programação orientada a objetos para garantir o princípio do encapsulamento de dados.

O encapsulamento de dados é visto como o agrupamento de dados com os métodos que operam nesses dados.

Esses métodos são, obviamente, o ***getter*** para recuperar os dados e o ***setter*** para alterar os dados.

POLIMORFISMO

Polimorfismo é a capacidade de um objeto poder ser referenciado de várias formas

!cuidado! Polimorfismo não quer dizer que o objeto fica se transformando, muito pelo contrário, um objeto nasce de um tipo e morre daquele tipo, o que pode mudar é a maneira como nos referimos a ele.

Quando ele receber um objeto que realmente é de uma classe X, o seu método reescrito será invocado.

Não importa como nos referenciamos a um objeto, o método que será invocado é sempre o que é dele.

Introdução a Programação Orientação a Objetos I

Código da turma no classroom: iqmtxhd

Aulas: Segunda - 18:30 e Quinta - 18:30