Programação Orientada a Objetos

- Método de programação que utiliza tipos de dados personalizados. Em vez de apenas utilizar tipos primitivos (int, float, str..) , permite a construção de novos tipos de dados.

Baseia-se fundamentalmente no conceito de Objeto.

O que é um objeto?

- É uma ocorrência específica de uma classe "instância de classe"

* Representa entidades do mundo real como carro, pessoas, conta corrente... e também outros conceitos gráficos (como quadrado, esfera, triângulo...)

* Possui características próprias (atributos) e executa ações (métodos) provenientes da classe que originou o objeto.

*Todos os objetos de uma classe são idênticos no que diz respeito a sua interface e implementação – o que difere um objeto de outro é seu estado e sua identidade.

Como criar (instanciar) um objeto em Python:

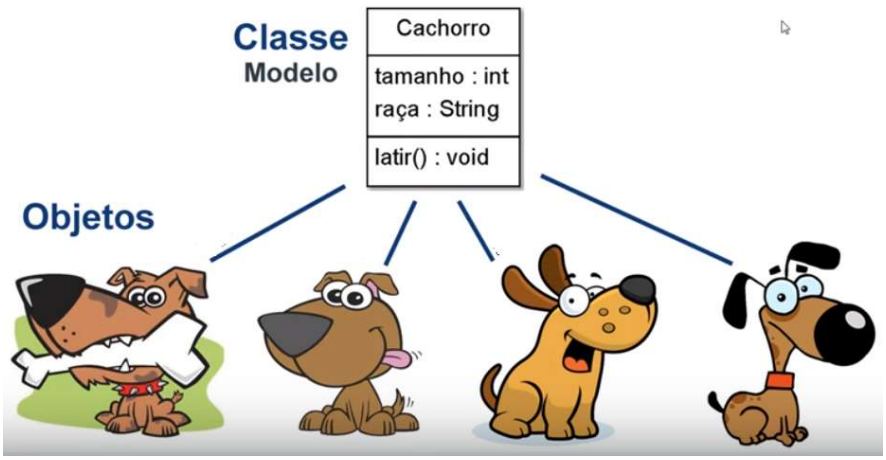
Sintaxe:

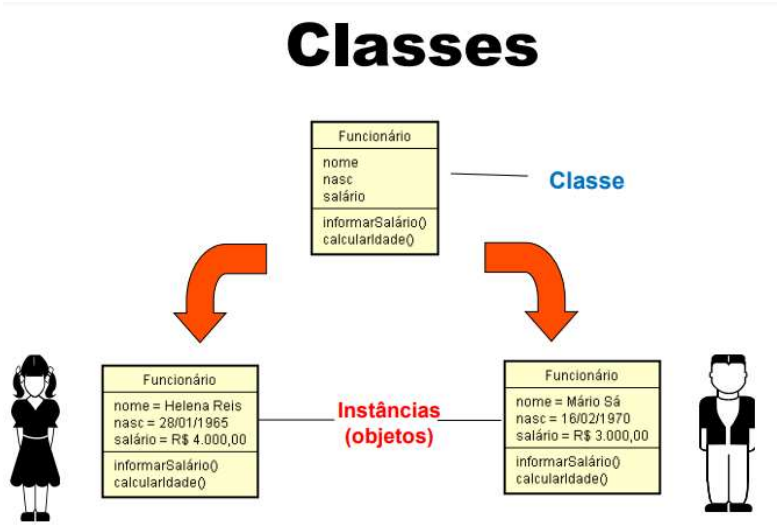
nomeObjeto = NomeClasse(parâmetros)

a atribuição cria uma nova instância da classe, e atribui o objeto criado a variável local nomeObjeto.

Quando o código acima é executado, o método construtor da classe é invocado automaticamente.

É possível instanciar vários objetos a partir de uma classe, e estes objetos são independentes entre si. Possuem os mesmos atributos porém com valores diferentes.





Classe

Representam ideias ou conceitos, classificando entidades que tenham propriedades similares.
* Tipo personalizado de dados, “molde”, para criação de objetos.
* Estrutura de dados personalizada (tipo que eu posso criar com as características que eu desejo)

Uma classe é composta por atributos e métodos.
Atributo: característica (propriedade) particular de uma ocorrência de uma classe, por exemplo, nome e raça de um cachorro.
Existem 2 tipos principais de atributos (ou variáveis):
- variável de classe: pertence a classe em si (o seu valor é compartilhado entre todos os objetos)
- variável de instância: pertence a cada objeto (o seu valor é individual – daquele objeto)

Método: lógica contida dentro de uma classe (sequência de comandos), identificada por um nome.
* Quando um método é executado, dizemos que ele foi invocado.

* Métodos utilitários: métodos usados apenas dentro da classe, não sendo parte da interface pública do utilizada pelo código cliente.
São acessados por outros métodos, dentro do objeto, para realizar atividades específicas.
Devem ser preferencialmente nomeados com um underscore como prefixo.

* Os métodos sempre têm self como primeiro argumento, “self” se refere a uma instância da classe.

Exercícios:

1) Crie uma classe Livro que possui os atributos nome, qtdPaginas, autor e preço.
   Crie os métodos getPreco() para obter o valor do preço e o método setPreco() para setar um novo valor do preço.
   Crie um código de teste

2) Implemente uma classe Aluno,
   com os seguintes atributos: nome, curso, tempoSemDormir (em horas).
   com os seguintes métodos:
   - estudar (que recebe como parâmetro a qtd de horas de estudo e acrescenta tempoSemDormir)
   – dormir (que recebe como parâmetro a qtd de horas de sono e reduz tempoSemDormir)
   Crie um código de teste da classe, criando um objeto da classe aluno e usando os métodos estudar e dormir.
   Ao final imprima quanto tempo o aluno está sem dormir.

3) Implemente uma classe chamada Carro com as seguintes propriedades:
   •Um veículo tem um certo consumo de combustível (medidos em km / litro) e uma certa quantidade de combustível no tanque.
   •O consumo é especificado no construtor e o nível de combustível inicial é 0.
   •Forneça um método andar( ) que simule o ato de dirigir o veículo por uma certa distância, reduzindo o nível de combustível no tanque de gasolina. Esse método recebe como parâmetro a distância em km.
   •Forneça um método obterGasolina( ), que retorna o nível atual de combustível.
   •Forneça um método adicionarGasolina( ), para abastecer o tanque.
   •Faça um programa para testar a classe Carro.

   Exemplo de uso:

```
meuFusca = Carro(15);                 # 15 quilômetros por litro de combustível.
meuFusca.adicionarGasolina(20);       # abastece com 20 litros de combustível.
meuFusca.andar(100);                  # anda 100 quilômetros.
meuFusca.obterGasolina()              # Imprime o combustível que resta no tanque.
```