Ciência da Computação

Programação Orientada a Objetos

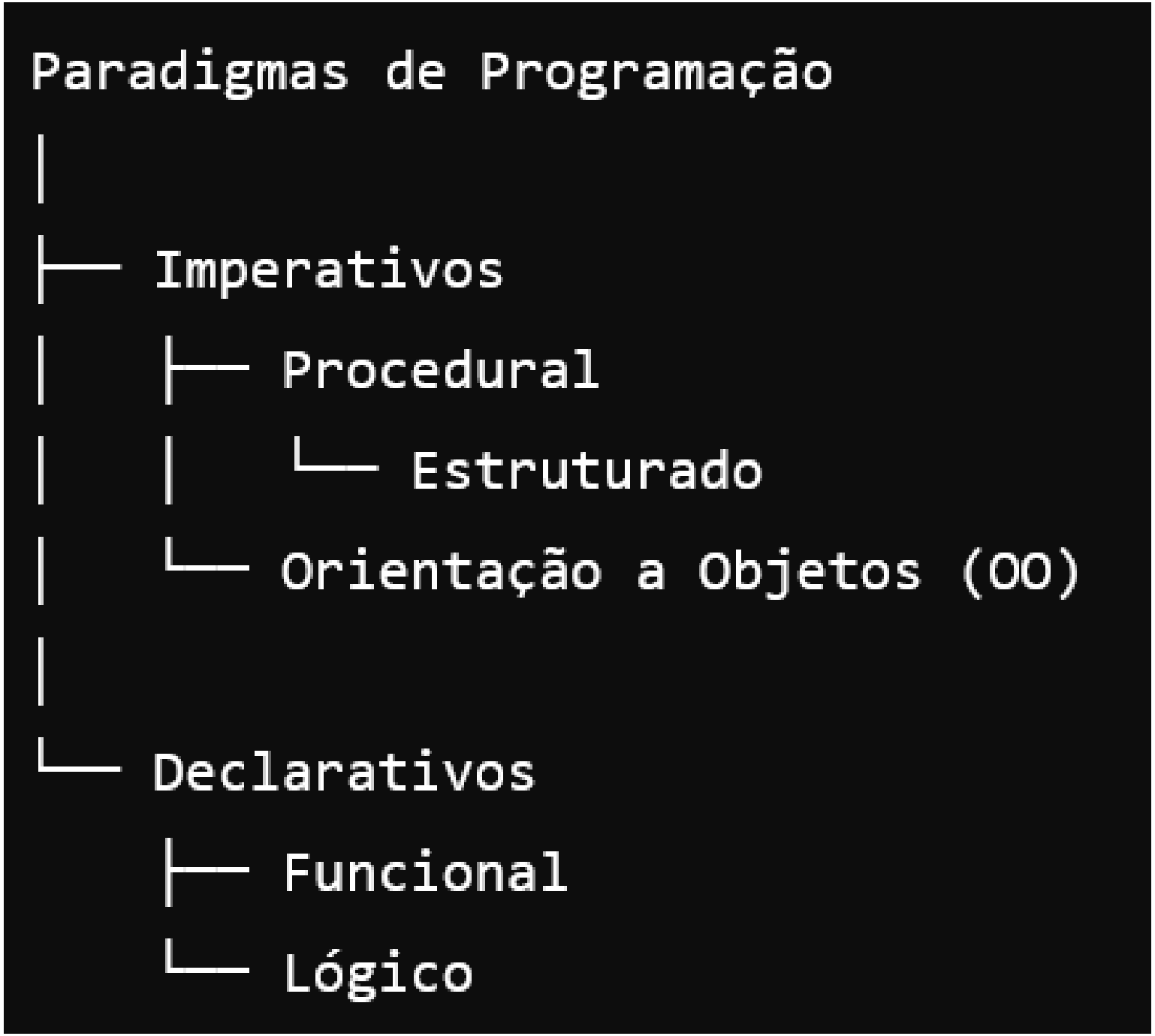
Prof. Luciano Rodrigo Ferretto

2024/02 - Org. Abs. na Programação

Grupo do WhatsApp

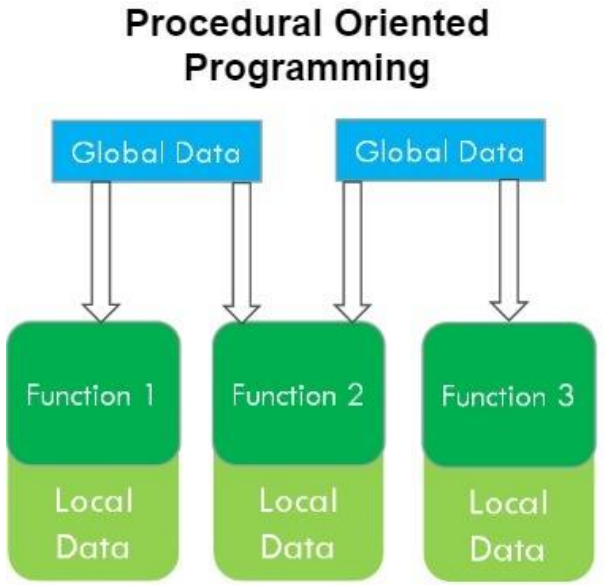
Nossa Aula de hoje

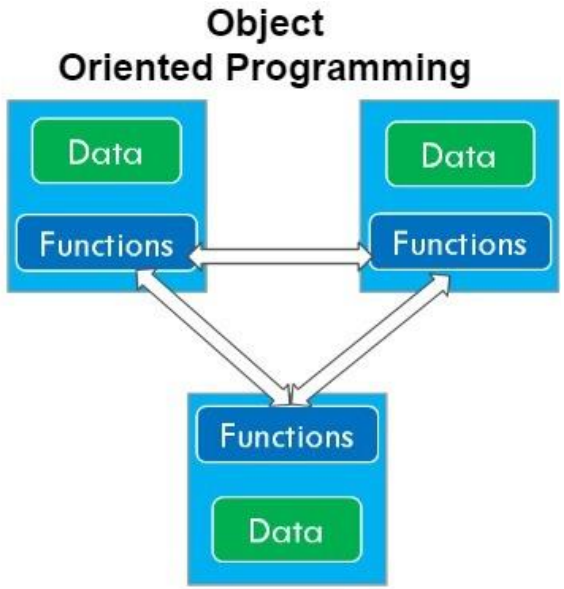
- Finalizar o Cadastro de Veículos em Python
- Introdução ao Paradigma Orientado a Objetos
  - Vantagens
  - Pilares
- Abstração – Orientação a Objetos em Python


Paradigmas de Programação
Imperativos
Procedural
Estruturado
Orientação a Objetos (OO)
Declarativos
Funcional
Lógico

Paradigma Procedural x Orientado a Objetos

- Procedural:
  - Procedimentos (Funções)
  - Controle de Fluxo
  - Variáveis e Dados Globais
  - **Melhor desempenho (quando bem escrita)**
- POO:
  - Reutilização de códigos
  - Abstração de dados
  - Desacoplamento
  - Etc.



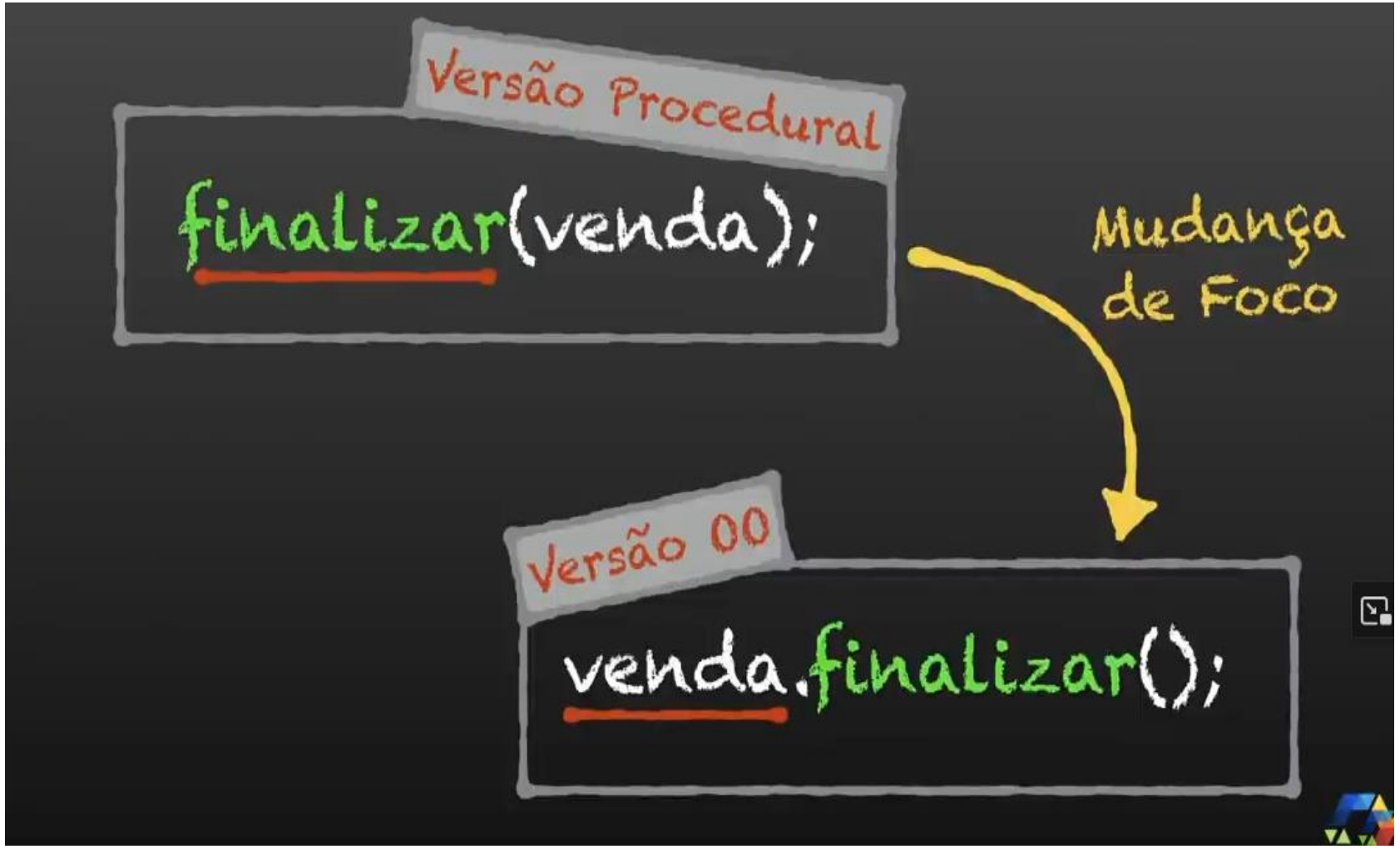


Orientação a objetos – OO

- É um Paradigma de Programação
- Em vez de operar apenas com tipos de dados primitivos, podemos construir novos tipos de dados (abstração);
- Baseia-se fundamentalmente no conceito de **Objetos**

Ciência da Computação

Mudança de FOCO

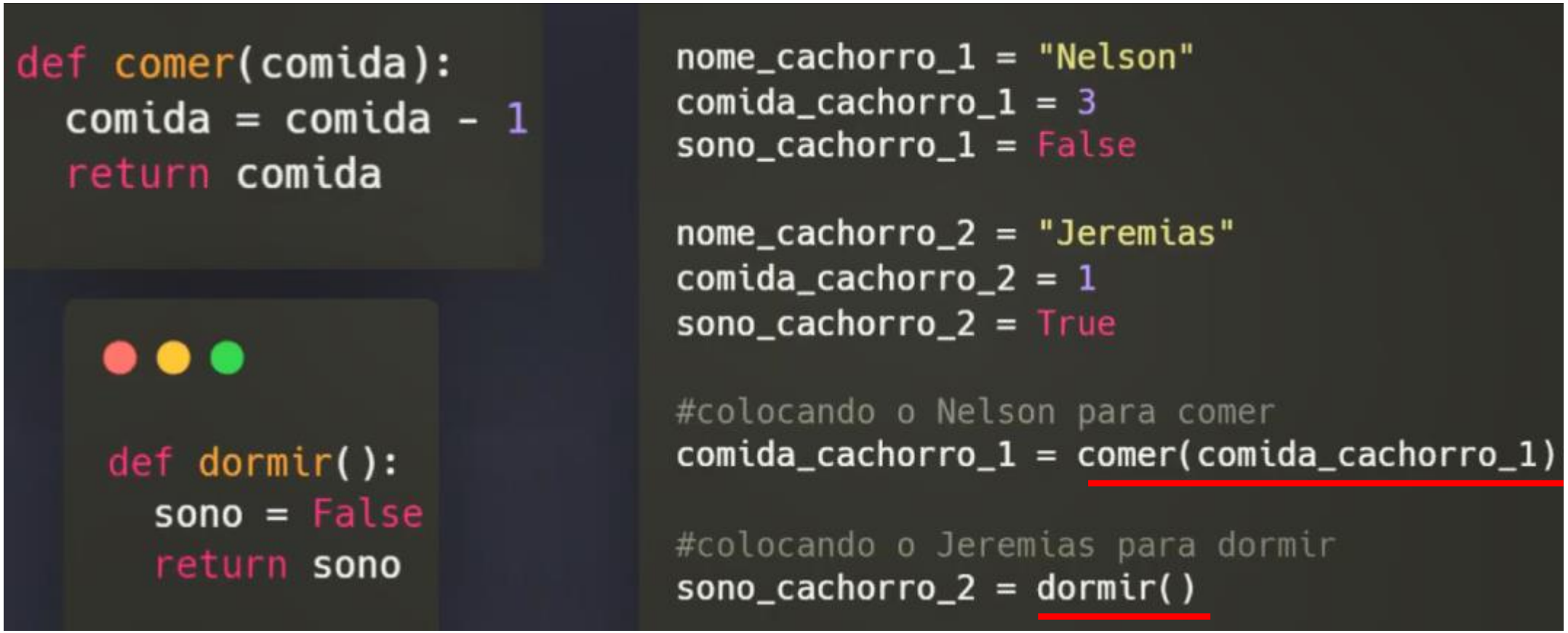


Computaçao

https://www.youtube.com/watch?v=pbb0jzXt_xA

Mudança de FOCO

- Aqui temos o Foco no **Processo** (comer, dormir)



Mudança de FOCO

- Aqui mudamos o Foco para o **Objeto** (cachorro_1, cachorro_2)

```
class Cachorro:
    def __init__(self, nome, comida, sono):
        self.nome = nome
        self.comida = comida
        self.sono = sono

    def comer(self):
        self.comida -= 1

    def dormir(self):
        self.sono = False
```

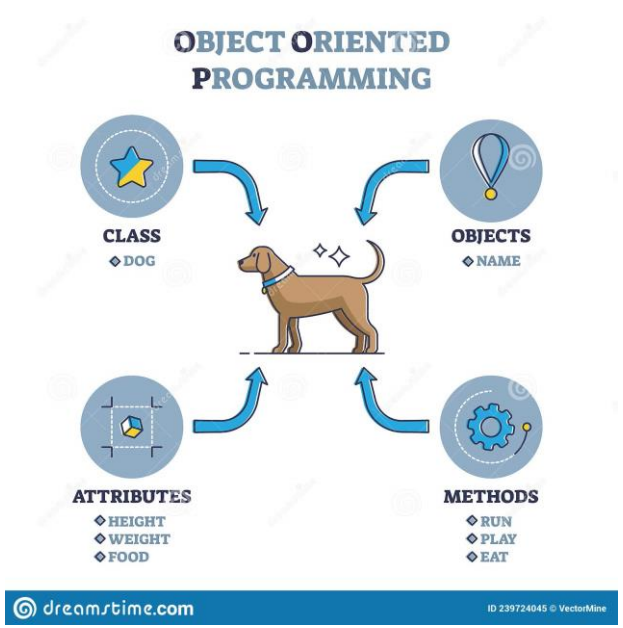
```
cachorro_1 = Cachorro("Nelson", 3, False)
cachorro_2 = Cachorro("Jeremias", 1, True)

cachorro_1.comer()
cachorro_2.dormir()
```

Mas atenção ...

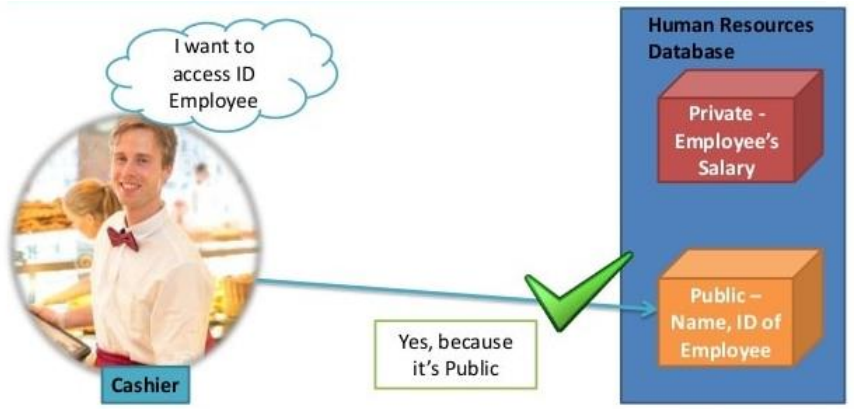
- Em Programação Orientada a Objetos (POO), um objeto **NÃO** se refere apenas a coisas físicas, como carros, cachorros, ou produtos.
- Um objeto pode também representar processos e ações, como uma venda, uma compra, ou até mesmo uma transação financeira.
- Esses processos, assim como os objetos físicos, possuem **características (atributos)** e **comportamentos (métodos)** que podemos modelar e manipular em nosso código.
- Assim, a POO nos permite criar representações **abstratas** de conceitos do mundo real e também de processos que acontecem nele.

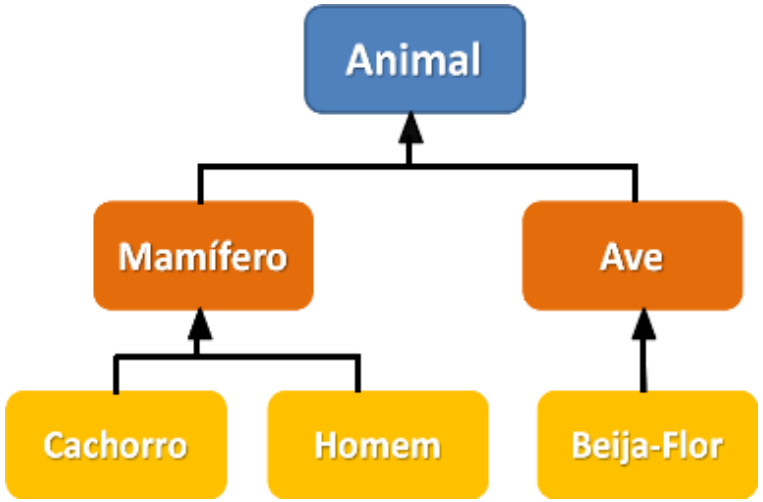
Pilares da Orientação a objetos

Ciência da
Computação


OBJECT ORIENTED PROGRAMMING
CLASS
DOG
OBJECTS
NAME
ATTRIBUTES
HEIGHT
WEIGHT
FOOD
METHODS
RUN
PLAY
EAT
dreamstime.com
ID 239724045 © VectorMine


The Four Pillars
OOP
Abstraction
Inheritance
Polymorphism
Encapsulation


I want to access ID Employee
Cashier
Yes, because it's Public
Human Resources Database
Private - Employee's Salary
Public – Name, ID of Employee


Animal
Mamífero
Ave
Cachorro
Homem
Beija-Flor


"Now Speak!"
"Quack"
"Woof"
"Meow"

Abstração

- **Abstração** é o processo de identificar e definir características essenciais de objetos do mundo real, representando essas características em forma de **classes** e **objetos**.

- Uma **classe** é uma estrutura que define atributos e métodos representando um tipo de objeto, fornecendo um modelo para criar instâncias desse tipo.
- **Atributos** são as características que vamos utilizar.
- **Métodos** são os comportamentos que os objetos dessa classe terão.

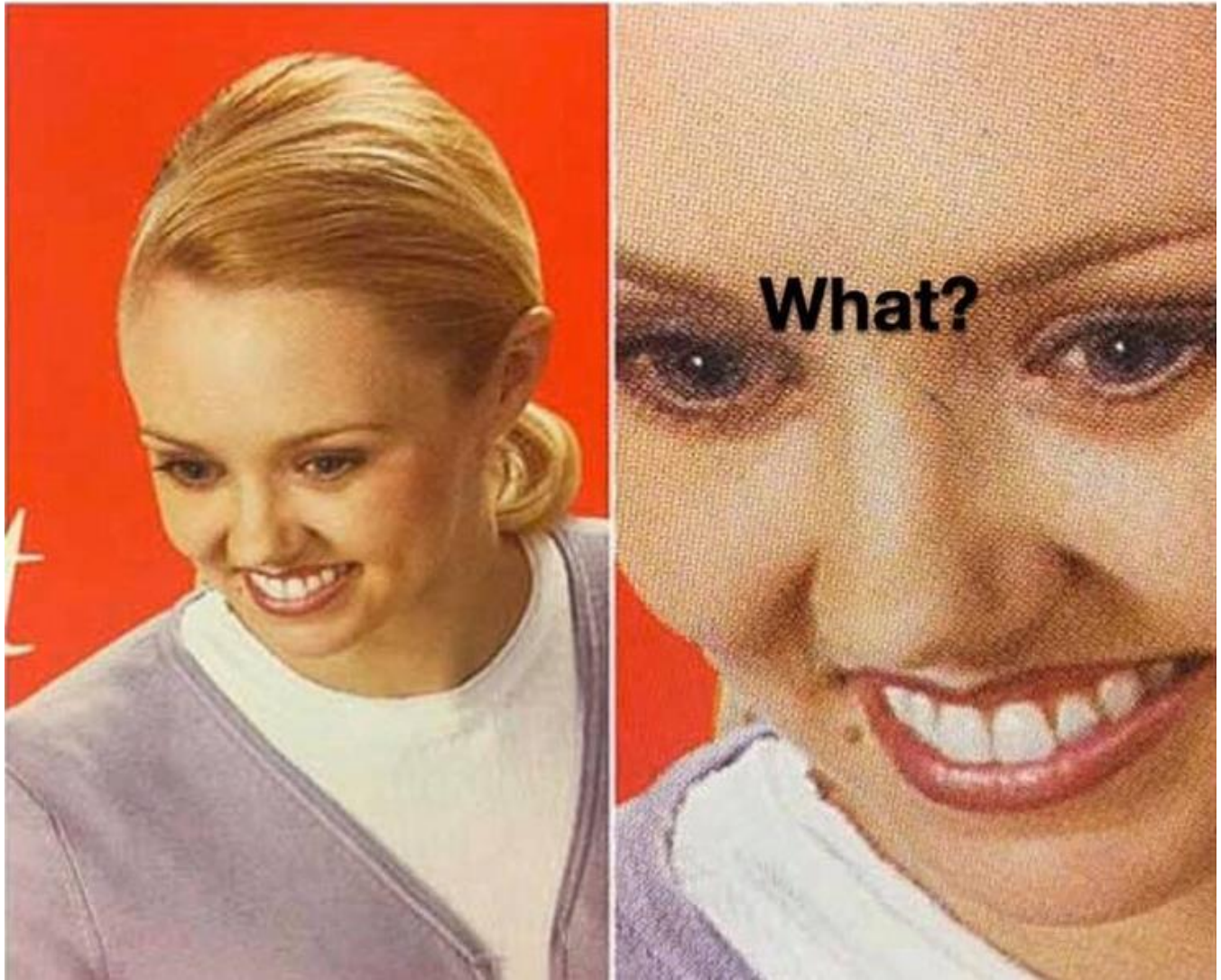
Abstração

- Trata-se de representar uma entidade do mundo real (pode ser um objeto, uma pessoa, um processo, etc.) na forma de **classes.**
-
- Essa visão simplificada é então transformada em **Classes** para que possamos utilizá-las em nosso sistema.

- Esse processo de criar uma classe pode ser considerado como o processo de criar um novo tipo de dado.

Ciência da Computação

Você sabia que em Python
TUDO É OBJETO???



```
class int:
    @overload
    def __new__(cls, x: ConvertibleToInt = ..., /) -> Self: ...
    @overload
    def __new__(cls, x: str | bytes | bytearray, /, base: SupportsIndex)
    def as_integer_ratio(self) -> tuple[int, Literal[1]]: ...
    @property
    def real(self) -> int: ...
```

```
class str(Sequence[str]):
    @overload
    def __new__(cls, object: object = ...) -> Self: ...
    @overload
    def __new__(cls, object: ReadableBuffer, encoding: str =
    @overload
    def capitalize(self: LiteralString) -> LiteralString: ..
    @overload
```

Abstração - Classes

- Classe serve como um modelo ou plano para criar objetos.
- Ela define as características (**atributos**) e comportamentos (**métodos**) que os objetos desse tipo terão.
- Uma classe é uma estrutura abstrata que encapsula o estado e o comportamento associados a um conceito específico.
- Em termos simples, uma classe pode ser considerada como um "**molde**" a partir do qual os objetos são criados.
- Ela descreve quais informações um objeto pode armazenar e quais operações ele pode realizar.

Ciência da Computação

Exemplo de Classe

- Seguindo o exemplo da aula anterior, caso precisamos desenvolver um algoritmo para receber os dados do veículo, então o primeiro passo seria abstrair o "veículo" do mundo real para o nosso algoritmo OO.
- Para isso temos que elencar quais as características (atributos) do veículo são necessárias dentro do nosso escopo, e também quais são as ações que serão executadas pelos objetos desse tipo (métodos).

Ciência da Computação

Exemplo de Classe

```python
class Veiculo:
    def __init__(self, marca, modelo, ano, placa):
        self.marca = marca
        self.modelo = modelo
        self.ano = ano
        self.placa = placa

    def calcularTempoUso(self):
        return 2024 - self.ano
```

Ciência da Computação

Objeto

- Objeto é uma **instância** concreta de uma classe na POO.
- Ele representa uma entidade específica com características (atributos) e ações (métodos) associadas, conforme definidas na classe da qual foi criado.
- **Em outras palavras, um objeto é a representação real de um conceito abstrato (classe).**

Ciência da Computação

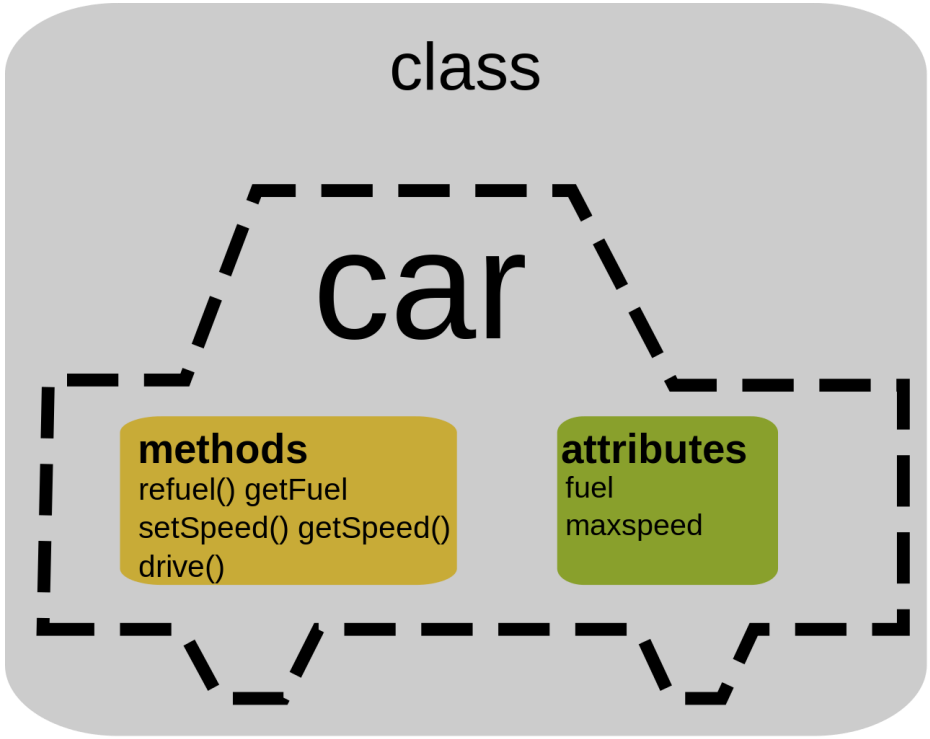
Objeto

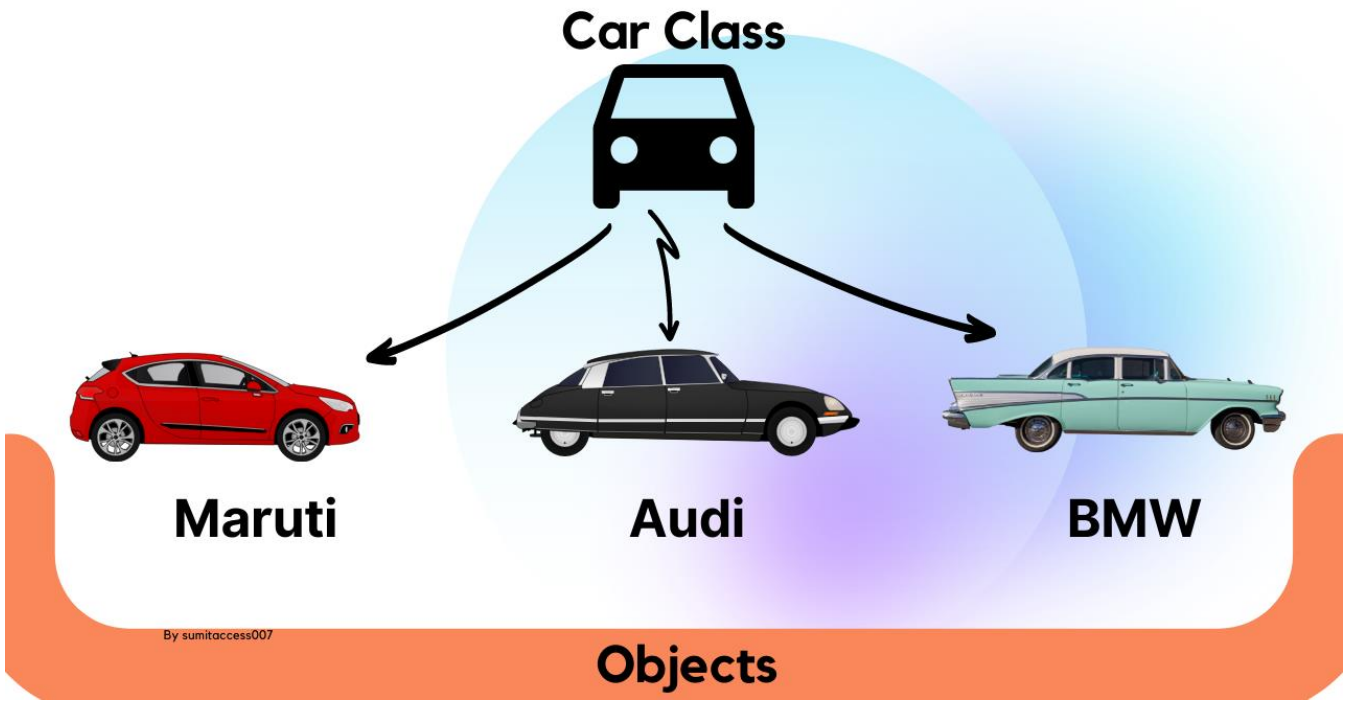
- Pense nos objetos como os "indivíduos" ou "casos" que seguem o modelo estabelecido pela classe.

- Cada objeto tem seu próprio conjunto de valores de atributos, que definem seu estado único, e pode executar os métodos associados para realizar ações específicas.

Ciência da Computação

Abstração





Ciência da Computação

Objeto

- Por exemplo, usando a classe “Veiculo" definida anteriormente, podemos criar um objeto chamado “meuVeiculo” e um segundo objeto chamado “teuVeiculo”.
- Apesar de serem objetos do mesmo tipo, são objetos distintos, ou seja, cada um armazena informações em um espaço de memória separados.
- Uma mesma classe pode ter **infinitos** objetos instanciados a partir dela;

```
1   class Veiculo:
2       def __init__(self, marca, modelo, ano, placa):
5           self.ano = ano
6           self.placa = placa
7
8       def calcularTempoUso(self):
9           return 2024 - self.ano
10
11  meuVeiculo = Veiculo("Volkswagem", "Fusca", 1995, "ABC-1234")
12  teuVeiculo = Veiculo("Fiat", "Uno com Escada", 2003, "DEF-5678")
```

Objeto

- Podemos dizer que nas linhas 11 e 12 estamos declarando variáveis do tipo Veiculo e também estamos atribuindo à elas instâncias da classe, ou seja, objetos Veiculo.

```
1   class Veiculo:
2       def __init__(self, marca, modelo, ano, placa):
5           self.ano = ano
6           self.placa = placa
7
8       def calcularTempoUso(self):
9           return 2024 - self.ano
10
11  meuVeiculo = Veiculo("Volkswagem", "Fusca", 1995, "ABC-1234")
12  teuVeiculo = Veiculo("Fiat", "Uno com Escada", 2003, "DEF-5678")
```

Ciência da Computação

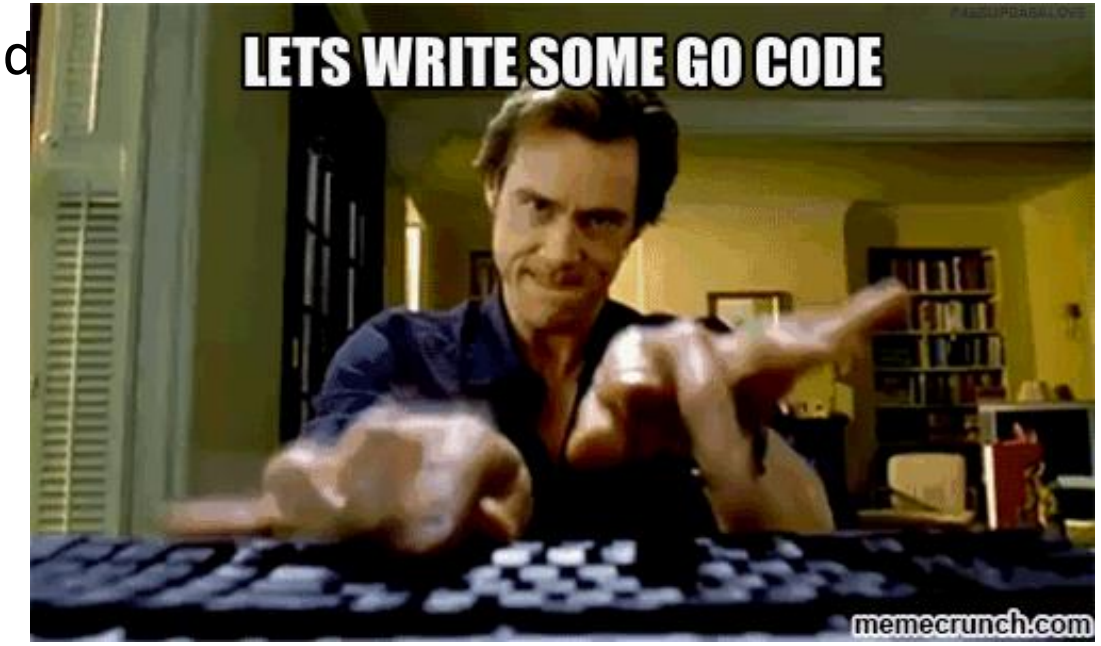
Objeto

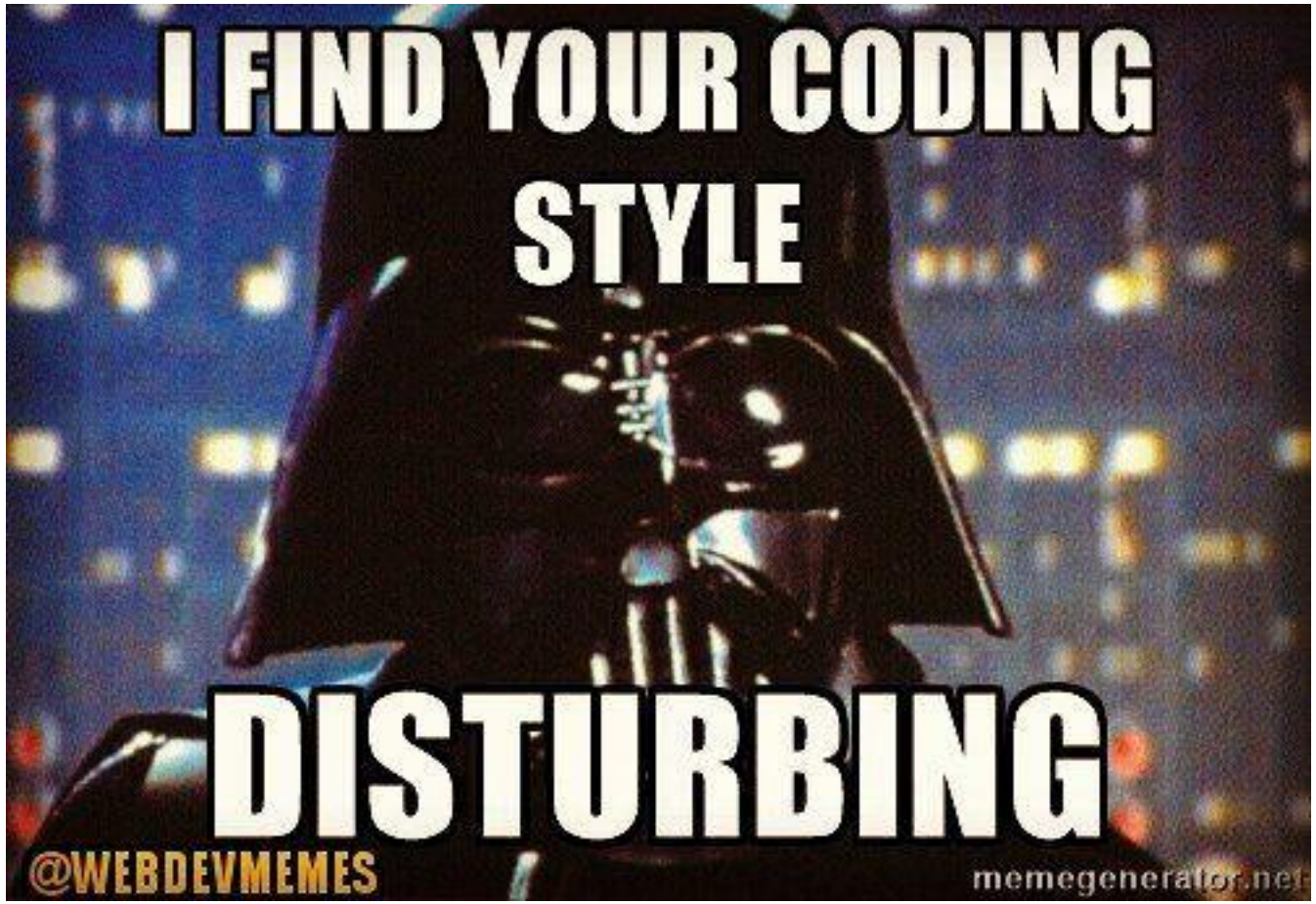
- Através dos variáveis que apontam para esses objetos, podemos acessar os atributos (ler/alterar) e também invocar os métodos relacionados à eles.

```
1   class Veiculo:
2       def __init__(self, marca, modelo, ano, placa):
5           self.ano = ano
6           self.placa = placa
7
8       def calcularTempoUso(self):
9           return 2024 - self.ano
10
11  meuVeiculo = Veiculo("Volkswagem", "Fusca", 1995, "ABC-1234")
12  teuVeiculo = Veiculo("Fiat", "Uno com Escada", 2003, "DEF-5678")
13
14  meuVeiculo.ano = 1995
15
16  print(meuVeiculo.calcularTempoUso())
17  print(teuVeiculo.calcularTempoUso())
```

Vamos *"codar"*...

- Ajuste o nosso sistema de cadastro de veículos da aula anterior de forma a ser um sistema orientado a objetos:
- **Opção 1: Cadastrar Veículo**
  - Coleta informações do veículo (marca, modelo, ano e placa).
  - Armazena as informações do veículo em uma lista.
- **Opção 2: Listar Veículos**
  - Mostra as informações de cada veículo cadastrad
- **Opção 3: Excluir Veículo**
  - Lista os veículos cadastrados.
  - Permite a exclusão de um veículo.
- **Opção 0: Sair**
  - Encerra o programa.





Ciência da Computação