



IN

INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

PY - POO

01 Introdução à programação orientada a objetos

02 Conceitos básicos

03 Exercícios



PY - POO

Paradigma de programação



Um paradigma de programação é uma "regra", uma estrutura, que uma linguagem de programação segue para podermos classificá-la.

Em outras palavras, podemos entender um paradigma de programação em como determinada linguagem irá fornecer recursos para resolver determinado problema.

Um paradigma de programação fornece e determina a visão que o programador possui sobre a estruturação e execução do programa. Por exemplo, em programação orientada a objetos, os programadores podem abstrair um programa como uma coleção de objetos que interagem entre si.

Por exemplo: Programação orientada a objetos, programação procedural, programação funcional.

PY - POO

Paradigma de orientação a objetos



O paradigma da orientação a objetos surgiu no fim dos anos 60

Alan Kay, um dos pais do paradigma da orientação a objetos, formulou a chamada a analogia biológica

Mais sobre paradigmas da programação:
https://pt.wikipedia.org/wiki/Paradigma_de_programa%C3%A7%C3%A3o

PY - POO

Paradigma de orientação a objetos



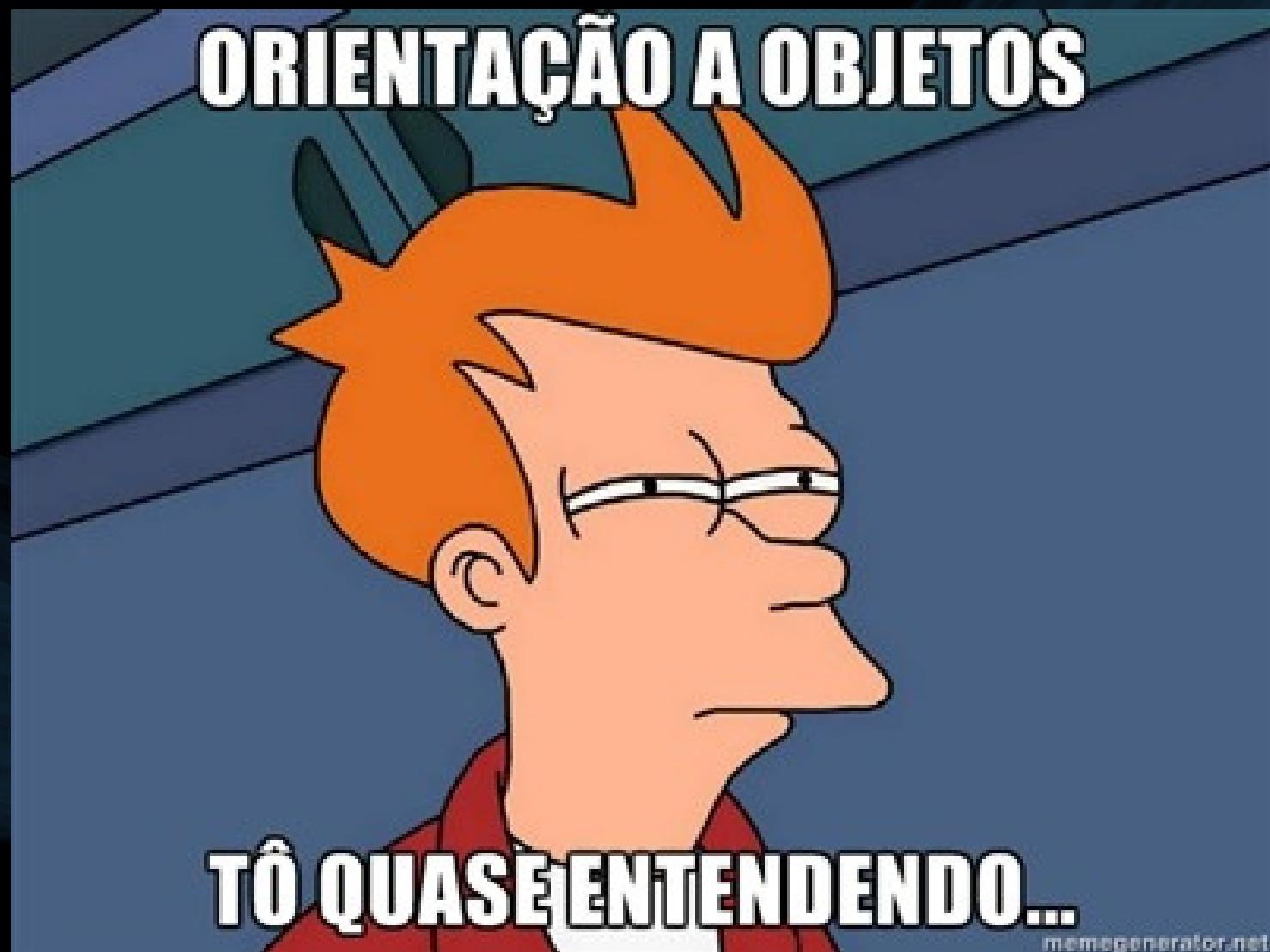
A POO foi criada para tentar aproximar o mundo real do mundo virtual.

A idéia fundamental é tentar simular o mundo real dentro do computador. Para isso, nada mais natural do que utilizar Objetos, afinal, nosso mundo é composto de objetos, certo?!

Na POO o programador é responsável por moldar o mundo dos objetos, e explicar para estes objetos como eles devem interagir entre si

PY - POO

Em orientação a objetos tudo é OBJETO!



Pense em um objeto como uma "*super variável*": O objeto armazena dados, também, pode-se fazer requisições a esse objeto, pedindo que ele execute operações.

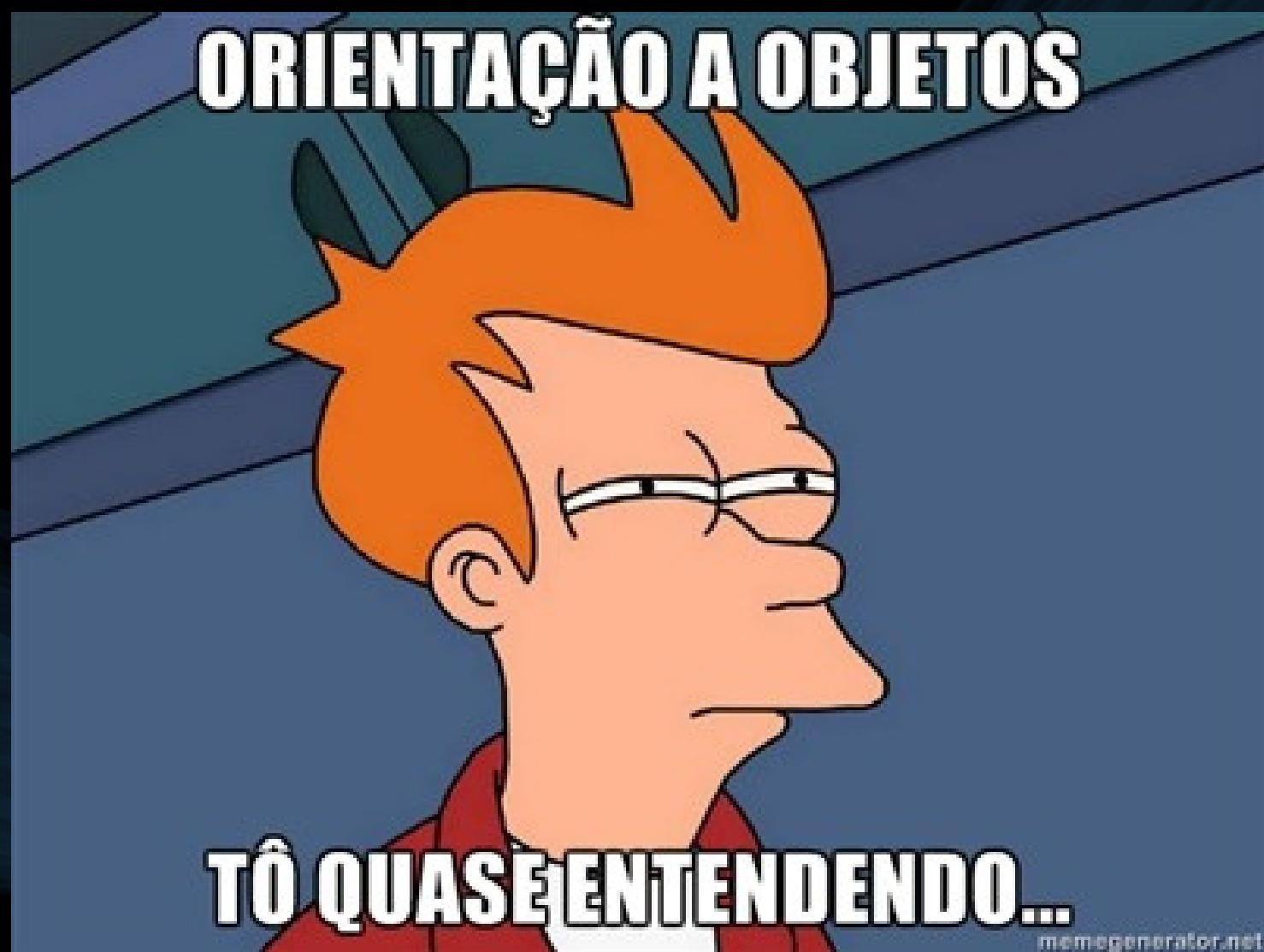
O exemplo abaixo é uma prova que usamos conceitos da programação orientada a objeto desde o começo de nossa jornada.



```
1 my_name = 'José'.upper() # Usando o método upper() da classe Str do Python
2 print(type(my_name)) # <class 'str'>
3 my_numbers = [1, 2, 3, 4] # Classe List
4 my_numbers.append(5) # Método append() da classe List do Python
5 print(type(my_numbers)) # <class 'list'>
```

PY - POO

Em orientação a objetos tudo é OBJETO!



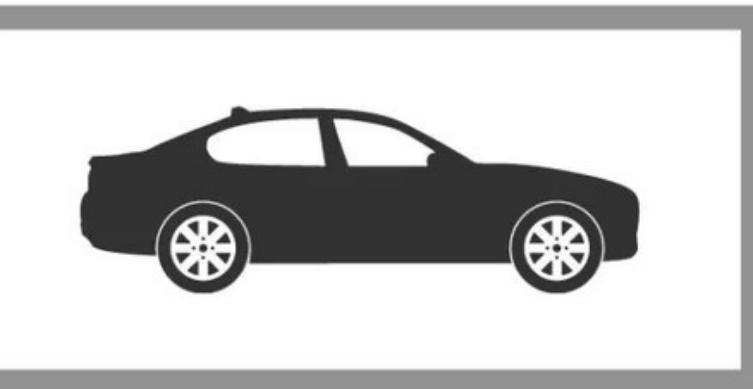
Um programa é uma coleção de objetos dizendo uns aos outros o que fazer!

Para fazer uma requisição a um objeto envia- se uma mensagem a este objeto.

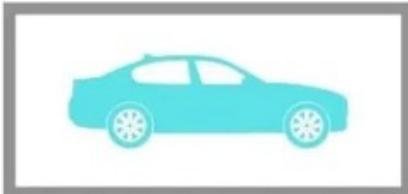
Uma mensagem é uma chamada de um método pertencente a um objeto em particular.

PY - POO

CLASSE CARRO

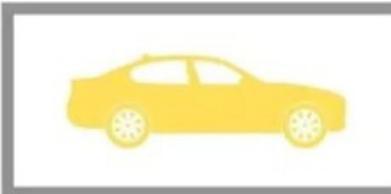


OBJETO



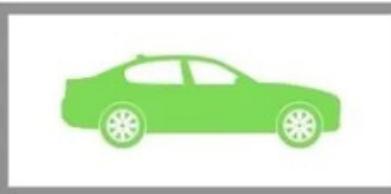
CELTA

OBJETO



CORSA

OBJETO

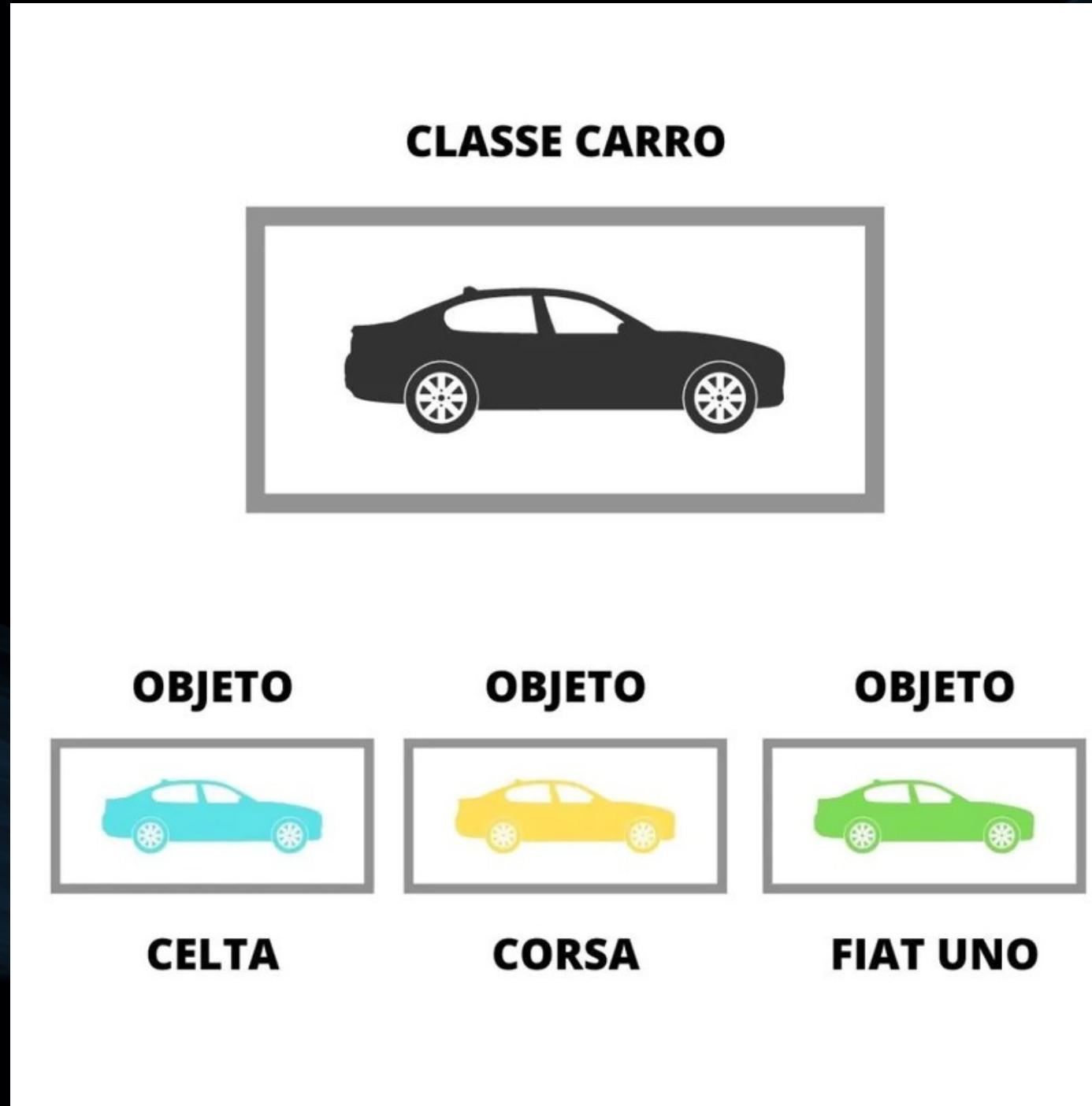


FIAT UNO

Contextualizando

De uma classe se gera um objeto, esses objetos possuem dados e instruções sobre como manipular os dados que estão ligados à solução do problema.

PY - POO



Classes

Podemos descrever um carro em termos de seus **atributos** que serão seus componentes ou características.

Conseguimos entender os **atributos** como as **varáveis** que pertencem àquela classe

Ou seja, os **atributos** são todas as características daquela classe, no exemplo de um carro temos os atributos:

- marca (string)
- modelo (string)
- ano (string)
- peso (float)
- possui 4 portas (boolean)

A classe pode ter quantos atributos quisermos e nós decidimos quais atributos farão parte, isso depende do que será importante para o nosso sistema e do nível de **abstração** que estamos fazendo da regra de negócio que o sistema irá atender.

PY - POO

Classes

Podemos também descrever algumas ações que o carro faz (que serão seus métodos)

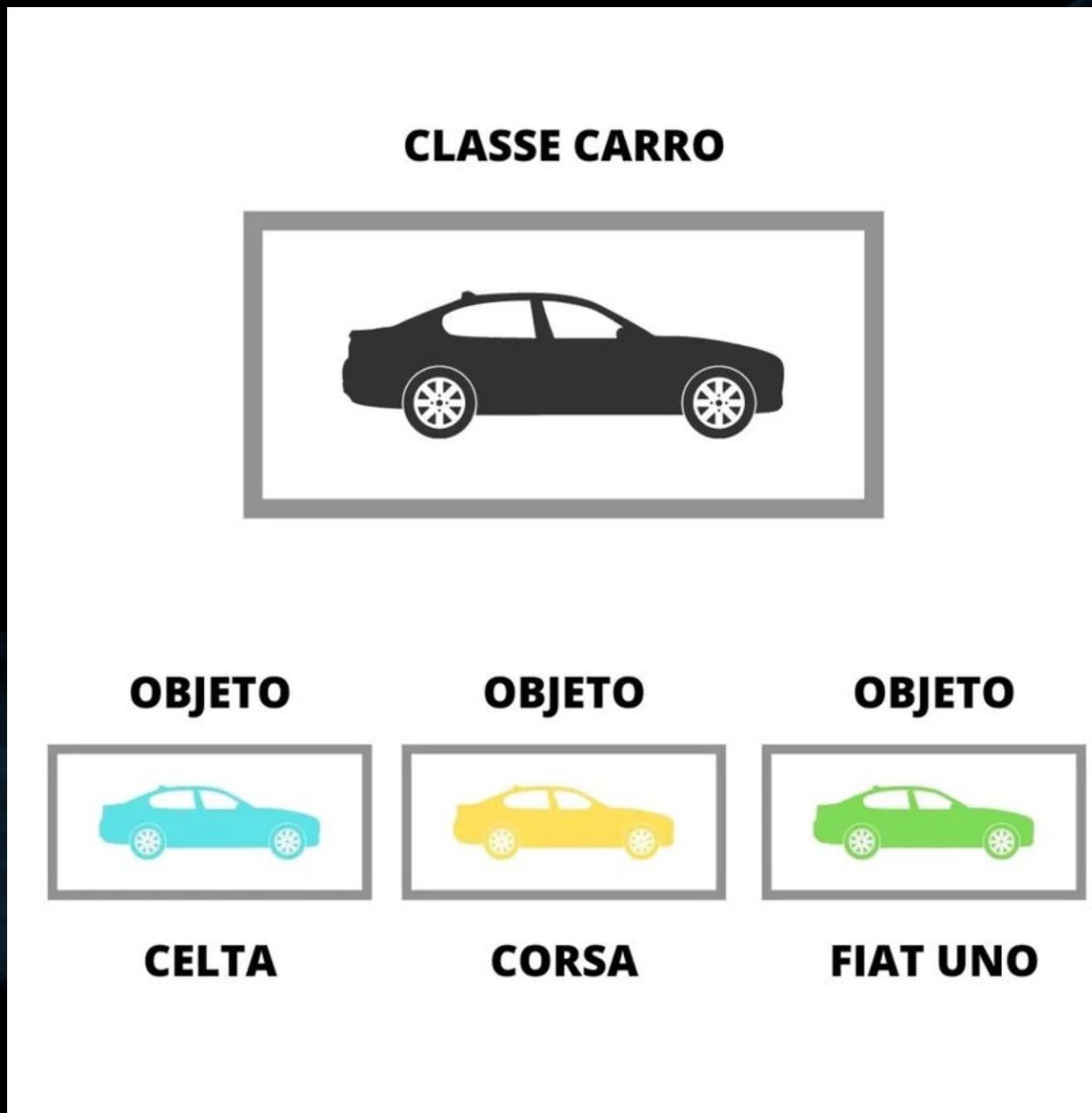
Conseguimos entender um **método** como uma função daquela classe.

Essas funções podem retornar alguma coisa ou não.

No exemplo de uma string temos a função `.upper()` que é um **método** da **classe** `Str` que retorna a mesma string, maiúscula.

No exemplo do carro temos:

- `ligar()`
- `buzinar()`
- `frear()`



PY - POO

Objetos

Vamos entender um pouco sobre os objetos

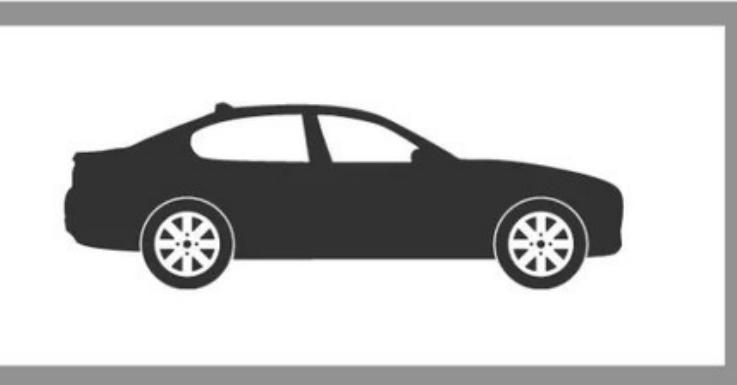
Já entendemos que as classes são abstrações que criamos de algo do mundo real, como fazemos para criar algo concreto dessas classes? Ou seja, criar os **OBJETOS**, que são instâncias da classe.

Para instanciar uma classe, utilizamos o **método construtor**.

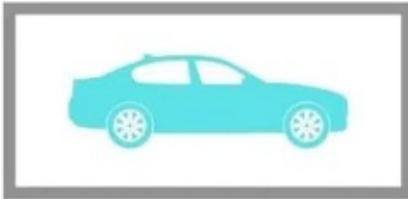
Será o método construtor que irá construir algo concreto daquela classe: O objeto.

Em Python utilizamos `__init__` para criar o método construtor de determinada classe.

CLASSE CARRO

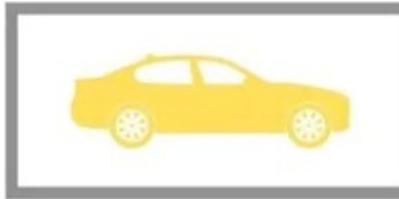


OBJETO



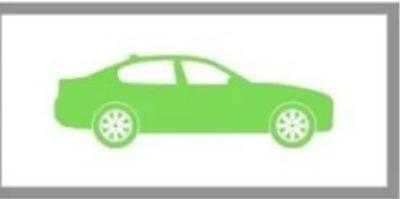
CELTA

OBJETO



CORSA

OBJETO



FIAT UNO

PY - POO

Classes e objetos no Python

Utilizamos a palavra reservada **class** para criar uma classe no Python.

Também definimos o nome da classe

Sempre que criarmos uma classe, vamos definir o nome com a primeira letra maiúscula



```
1 class Carro: # Primeira letra maiúscula
```

PY - POO

Classes e objetos no Python



```
1 class Carro(): # Primeira letra maiúscula
2     def __init__(self, marca, modelo, ano, peso, possui_4_portas):
3         self.marca = marca
4         self.modelo = modelo
5         self.ano = ano
6         self.peso = peso
7         self.possui_4_rimas = possui_4_portas
8     # Método construtor da classe Carro
```

O método construtor é composto por dois underline ou dunder-scores, essa função carregará parâmetros que farão referência aos dados atribuídos em nossos atributos.

Podemos utilizar a palavra reservada self para referenciar cada atributo para objetos diferentes, sabendo que nossa classe é um modelo para gerar novos objetos, utilizamos um método construtor para passar esses dados a nossos objetos



```
1 class Carro(): # Primeira letra maiúscula
2     def __init__(self, marca, modelo, ano, peso, possui_4_portas):
3         self.marca = marca
4         self.modelo = modelo
5         self.ano = ano
6         self.peso = peso
7         self.possui_4_rimas = possui_4_portas
8     # Método construtor da classe Carro
9
10    def ligar(self):
11        return f'O {self.modelo} Está ligado'
12    # Método ligar da classe Carro, está retornando uma string
```

PY - POO

Classes e objetos no Python

Na mesma indentação do método construtor, vamos criar métodos / funções. Esses métodos trazem funcionalidades a nosso objeto, todo comportamento do nosso objeto é referenciado por **self** ela referencia nossa instância, por isso é padrão de estrutura utilizá-lo como primeiro parâmetro na criação de nossos métodos.

PY - POO



```
1 class Carro(): # Primeira letra maiúscula
2     def __init__(self, marca, modelo, ano, peso, possui_4_portas):
3         self.marca = marca
4         self.modelo = modelo
5         self.ano = ano
6         self.peso = peso
7         self.possui_4_rodas = possui_4_portas
8     # Método construtor da classe Carro
9
10    def ligar(self):
11        return f'O {self.modelo} Está ligado'
12    # Método ligar da classe Carro, está retornando uma string
13
14 celtinha = Carro('Chevrolet', 'Celta', '2009', 1200, False)
15 # Objeto (instância) da classe Carro
```

Classes e objetos no Python

Agora é hora de criar nosso objeto/carro, para isso utilizamos **nome_da_classe()**

PY - POO



```
1 class Carro(): # Primeira letra maiúscula
2     def __init__(self, marca, modelo, ano, peso, possui_4_portas):
3         self.marca = marca
4         self.modelo = modelo
5         self.ano = ano
6         self.peso = peso
7         self.possui_4_racas = possui_4_portas
8     # Método construtor da classe Carro
9
10    def ligar(self):
11        return f'O {self.modelo} Está ligado'
12    # Método ligar da classe Carro, está retornando uma string
13
14 celtinha = Carro('Chevrolet', 'Celta', '2009', 1200, False)
15 # Objeto (instância) da classe Carro
16
17 print(celtinha.modelo) # Celta
18 print(celtinha.ligar()) # O Celta está ligado
```

Classes e objetos no Python

Conseguimos acessar os atributos e métodos de determinada classe utilizando o **.Nome_classe.atributo ou Nome_classe.metodo()**

PY - POO

Mão no código

Crie uma classe chamada Fatura que possa ser utilizada por uma loja de suprimentos de informática para representar uma fatura de um item vendido na loja. Uma fatura deve incluir as seguintes informações como atributos:

- o nome do item;
- o preço unitário do item;
- quantidade de item a ser faturado;
- valor total da fatura;

Sua classe deve ter um construtor que inicialize todos os atributos menos o valor total da fatura. Além disso, forneça um método chamado gerar_fatura que calcula o valor da fatura (isto é, multiplicar a quantidade pelo preço por item).





IN

INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

IN

PARABÉNS! VOCÊ TERMINOU A AULA 08 DO MÓDULO DE PYTHON