INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 11 – POO I

O QUE IREMOS APRENDER

01 PARADIGMA DE PROGRAMAÇÃO

02 PARADIGMA DE ORIENTAÇÃO A OBJETOS

03 ABSTRAÇÃO

04 OBJETOS

05 CONTEXTUALIZANDO

06 CLASSE

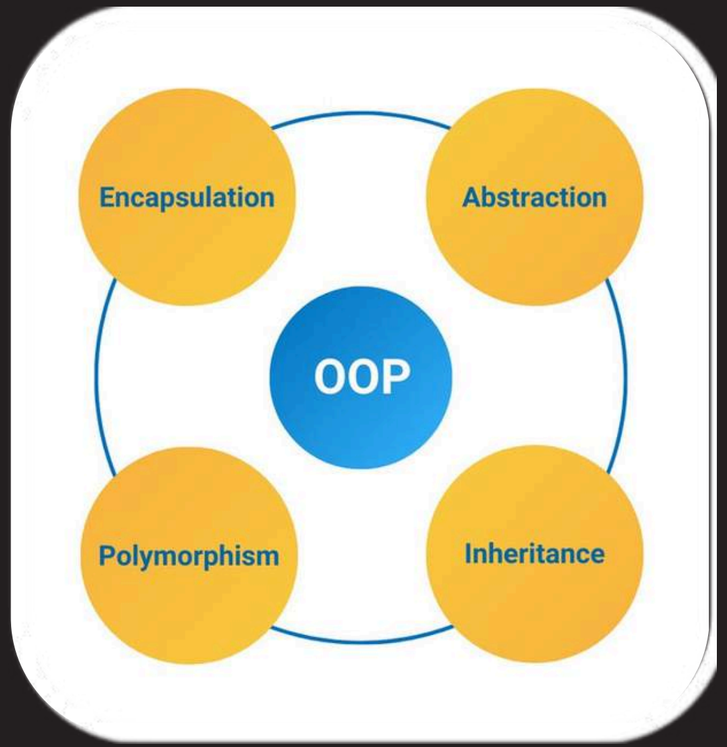
07 CLASSES E OBJETOS

08 PEDINDO AJUDA AO CHATGPT

09 MÃOS NO CÓDIGO

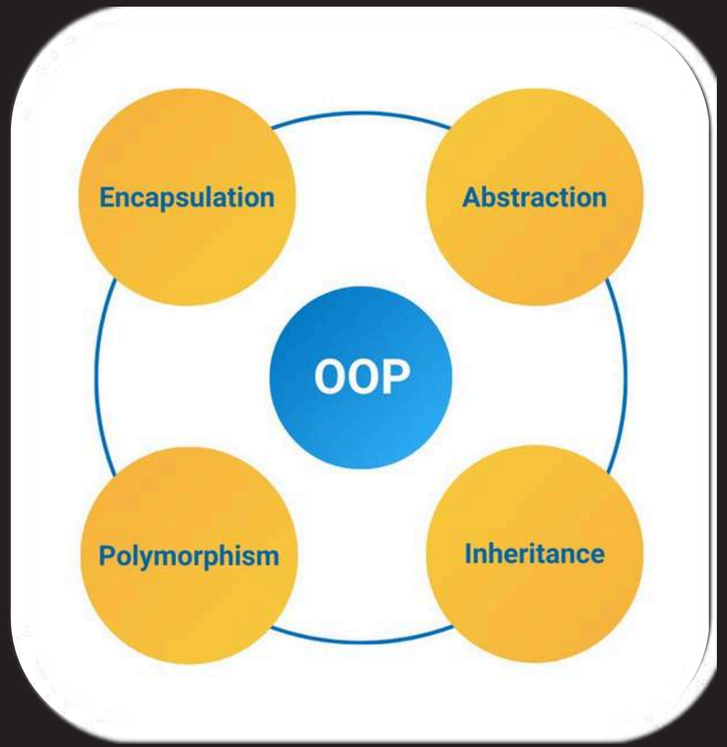
POO (Programação Orientada a Objetos)

Quem desenvolve softwares sabe o quanto o trabalho pode ser desafiador. Afinal de contas, programar é trazer soluções para os problemas dos clientes utilizando algoritmos em uma linguagem de programação. Portanto, a programação orientada a objetos trouxe agilidade no desenvolvimento de software, reduzindo o tempo para identificar e corrigir erros de programação. Isso revitalizou o paradigma da programação como um todo. E desse assunto que iremos tratar nesta aula.



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.



Paradigma de Programação

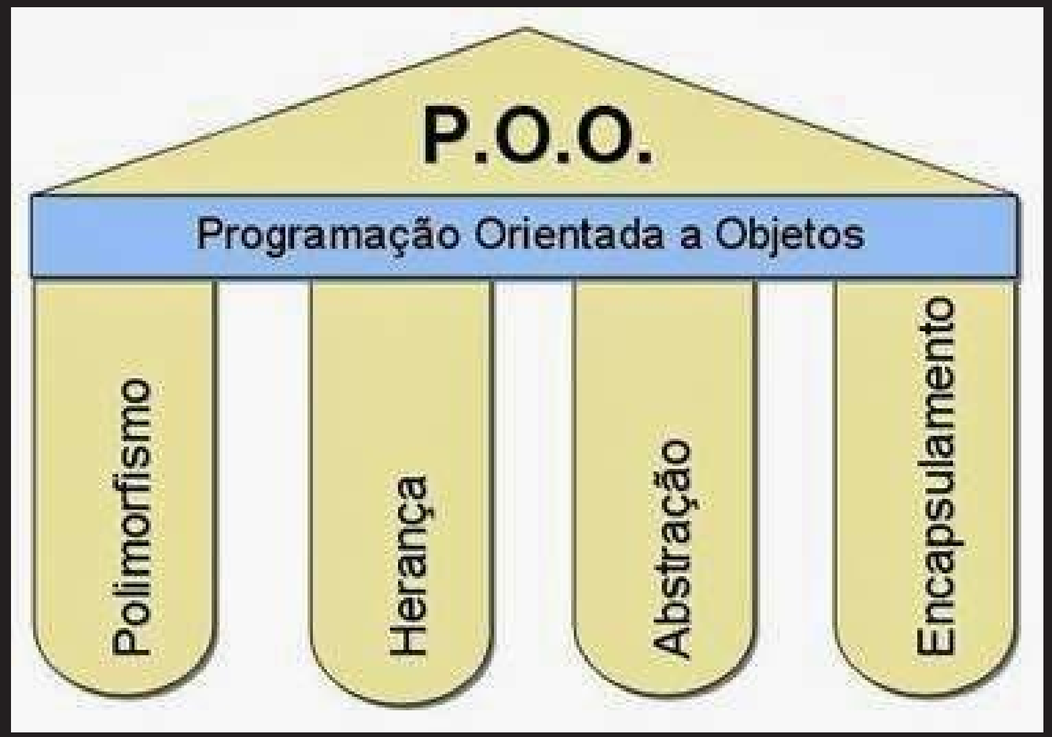
Um paradigma de programação fornece e determina a visão que o programador possui sobre a estruturação e execução do programa. Por exemplo, em programação orientada a objetos, os programadores podem abstrair um programa como uma coleção de objetos que interagem entre si.

Paradigma de Programação

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?!

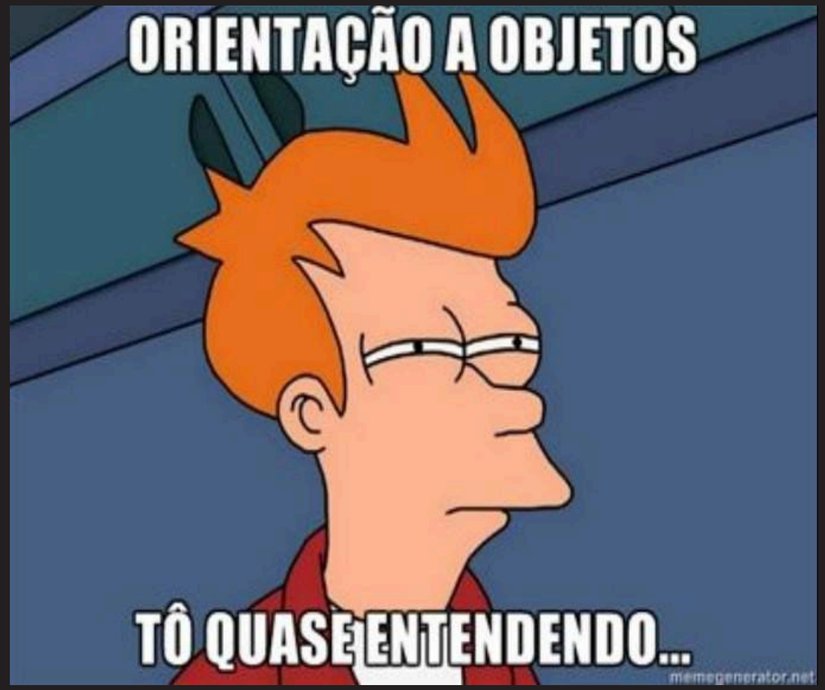
Paradigma de Orientação a Objetos



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

Abstração

Abstração envolve a criação de classes e objetos que representam entidades e conceitos do mundo real de forma simplificada. Isso permite que os programadores se concentrem apenas nos detalhes relevantes para o problema que estão resolvendo.



Objetos - POO

Em POO 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.

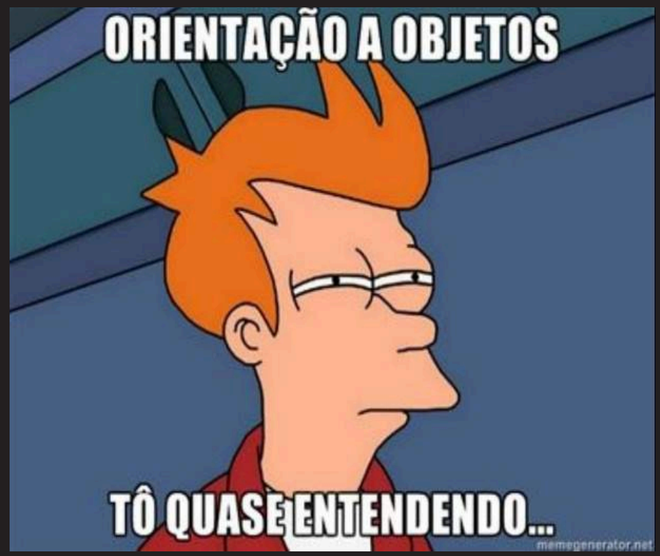
Objetos - POO

EXEMPLO:

```python
#Usando o método upper() da classe Str do python
my_name = 'Raama'.upper()
print(type(my_name)) #<class 'str'>

#Classe list
my_numbers = [4,3,2,1]

#Método append da class List do Python
my_numbers.append(0)
print(type(my_numbers)) #<class 'list'>
```



Objetos - POO

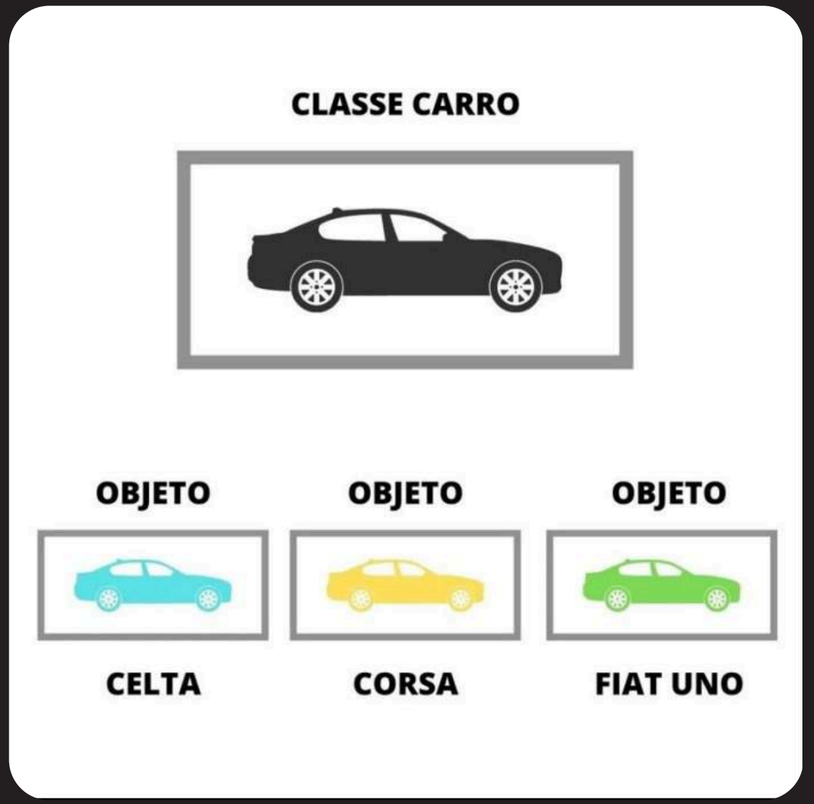
Um programa é uma coleção de objetos dizendo uns aos outros o que fazer!

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.

Contextualizando POO

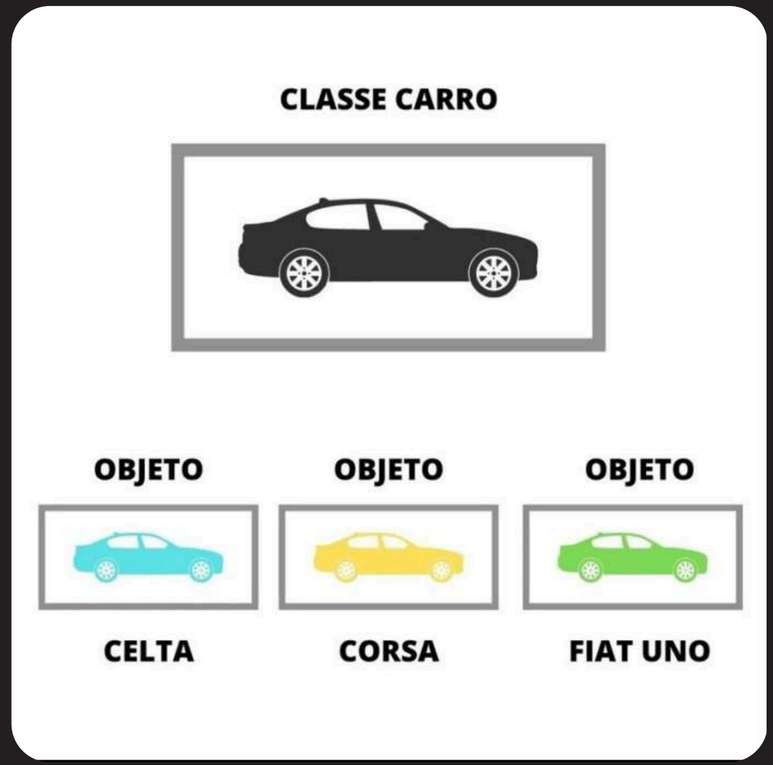
De uma classe se gera um objeto, esses objetos possuem dados e instruções sobre como manipular os dados que estão ligados à solução do problema.



Classe POO

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

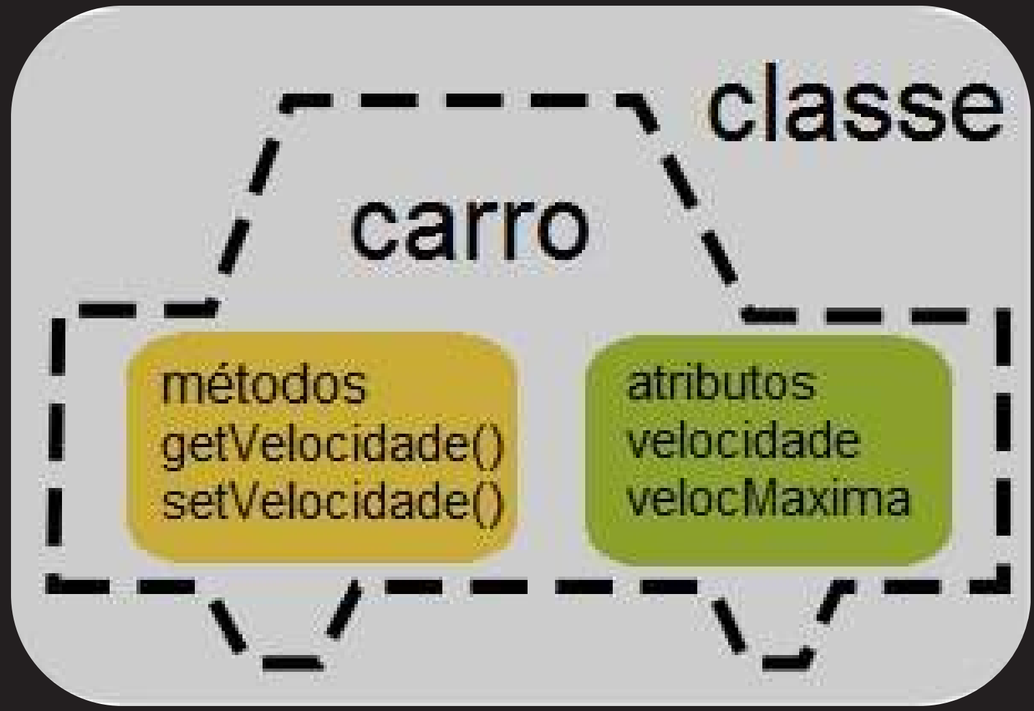
Conseguimos entender os atributos como as variáveis que pertencem àquela classe, ou seja, os atributos são todas as características daquela classe, no exemplo de um carro temos os atributos:



Classe POO

```python
class Carro:
    def __init__(self, modelo, cor, ano):
        self.modelo = modelo
        self.cor = cor
        self.ano = ano
        self.velocidade = 0  # Velocidade inicial é zero

```



Classe POO

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.

Classe POO

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.

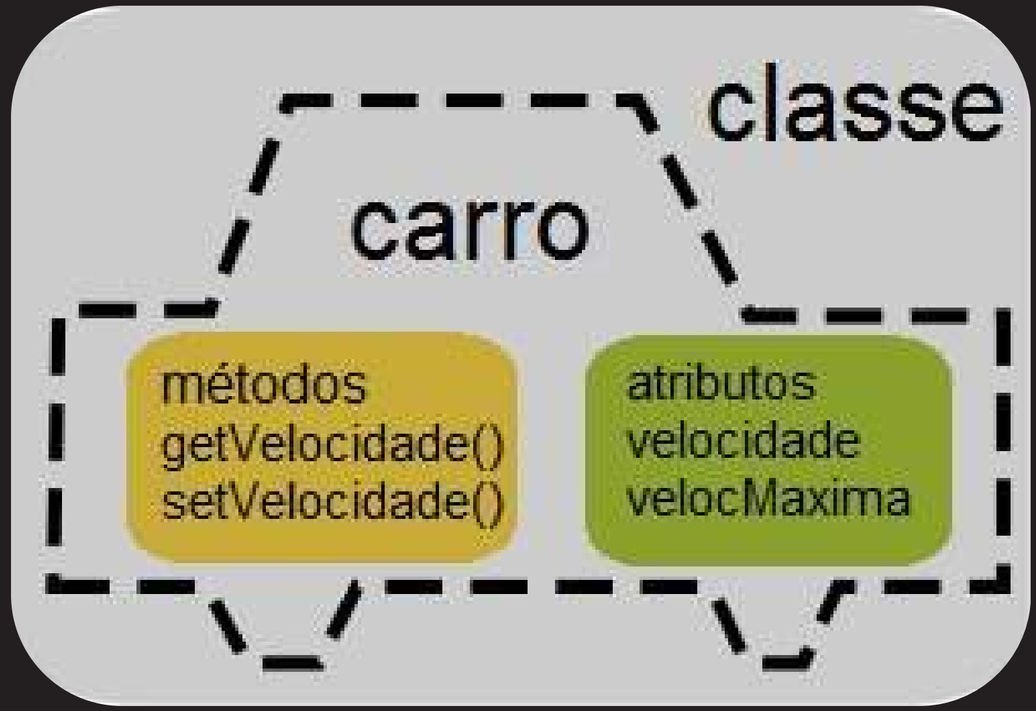
No exemplo de uma string temos a função .upper() que é um método da classe Str que retorna a mesma string, maiúscula.

Classe POO

Exemplos de métodos para Carro:

```python
def acelerar(self, velocidade):
    self.velocidade += velocidade
    print(f'O carro está agora a {self.velocidade} km/h.')

def frear(self, velocidade):
    if self.velocidade >= velocidade:
        self.velocidade -= velocidade
        print(f'O carro agora está a {self.velocidade} km/h após frear.')
    else:
        print('O carro já está parado.')
```



Objetos POO

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.

Classes e Objetos POO

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.

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

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

Classes e Objetos POO

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

```python
class Carro: #Primeira letra maiúscula
    def __init__(self, marca, modelo, ano, possui_4_rodas):
        self.marca = marca
        self.modelo = modelo
        self.ano = ano
        self.posssui_4_rodas = possui_4_rodas
    #Método construtor da classe carro
```

Classes e Objetos POO

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 e padrão de estrutura utilizá-lo como primeiro parâmetro na criação de nossos métodos.

Classes e Objetos POO

```python
class Carro:

    def __init__(self, marca, modelo, ano, possui_4_rodas):
        self.marca = marca
        self.modelo = modelo
        self.ano = ano
        self.possui_4_rodas = possui_4_rodas

    def ligar(self):
        return f"O {self.modelo} está ligado"

orochinho = Carro("Renault", "Oroch", 2019, True)

print(orochinho.modelo)
print(orochinho.ligar())
```

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

Classes e Objetos POO

```python
class Carro:

    def __init__(self, marca, modelo):
        self.marca = marca
        self.modelo = modelo
        self.velocidade = 0  # Inicializar o atributo velocidade com 0

    def acelerar(self):
        self.velocidade += 10
        print(f"{self.marca} {self.modelo} acelerando. Velocidade atual: {self.velocidade} km/h")

    def frear(self):
        self.velocidade -= 5
        print(f"{self.marca} {self.modelo} freando. Velocidade atual: {self.velocidade} km/h")

# Criando um objeto da classe Carro
meu_carro = Carro("Ford", "Mustang")

# Utilizando a abstração para interagir com o carro
meu_carro.acelerar()
meu_carro.frear()
```

PEDINDO AJUDA AO CHATGPT

You

PODE ME EXPLICAR POO(PARADIGMA DE ORIENTAÇÃO A OBJETOS ABSTRAÇÃO, CLASSES E OBJETOS) EM PYTHON DE FORMA RESUMIDA?

ChatGPT

Claro! Aqui está uma explicação resumida sobre Programação Orientada a Objetos (POO) em Python:

1. **O que é POO:**
   - A Programação Orientada a Objetos é um paradigma de programação que se baseia no conceito de "objetos", que representam entidades do mundo real e possuem características (atributos) e comportamentos (métodos).
2. **Abstração:**
   - Abstração é o processo de identificar as características essenciais de um objeto do mundo real e representá-las em um modelo de programação.
   - Em POO, as classes são usadas para definir esses modelos, representando entidades abstratas que têm atributos e métodos associados.
3. **Classes e objetos:**
   - Uma classe é uma estrutura que define um tipo de objeto, especificando os atributos que os objetos desse tipo terão e os métodos que podem ser usados para interagir com esses objetos.
   - Um objeto é uma instância de uma classe. Ou seja, é uma "cópia" da classe que contém valores específicos para seus atributos.
4. **Como criar classes e objetos em Python:**
   - Para criar uma classe em Python, use a palavra-chave `` `class` ``, seguida pelo nome da classe e, opcionalmente, pela classe pai entre parênteses.
   - Para criar um objeto a partir de uma classe, basta chamar o nome da classe seguido de parênteses, como se estivesse chamando uma função. Isso cria uma instância da classe, ou seja, um objeto.
5. **Atributos e métodos:**
   - Atributos são características dos objetos, como cor, tamanho, etc., representados por variáveis dentro da classe.
   - Métodos são funções associadas a objetos que definem seu comportamento, permitindo que os objetos executem ações ou realizem operações específicas.
6. **Encapsulamento, herança e polimorfismo:**
   - Além de abstração, classes e objetos em POO também envolvem conceitos como encapsulamento (ocultação de informações), herança (reutilização de código) e polimorfismo (capacidade de objetos de diferentes classes responderem ao mesmo método de maneira diferente).

Em resumo, POO em Python envolve a criação de classes e objetos que representam entidades abstratas do mundo real, com atributos e métodos que definem seu comportamento. Esses objetos podem interagir entre si, herdar características de outras classes e serem polimórficos, permitindo uma modelagem flexível e poderosa de sistemas complexos.

PEDINDO AJUDA AO CHATGPT

You
me indique 4 exercícios sobre esse assunto

ChatGPT
Claro! Aqui estão quatro exercícios para praticar Programação Orientada a Objetos (POO) em Python:

1. **Modelagem de classe básica:**
   Crie uma classe chamada `Veiculo` com os atributos `marca`, `modelo` e `ano`. Em seguida, crie um objeto dessa classe e atribua valores aos seus atributos. Por fim, crie um método na classe para exibir as informações do veículo.
2. **Herança:**
   Crie uma classe chamada `Carro` que herde da classe `Veiculo`. Adicione um atributo adicional na classe `Carro`, como `tipo_combustivel`, e sobrescreva o método de exibição para incluir também informações específicas de carros.
3. **Encapsulamento:**
   Modifique a classe `Veiculo` para tornar seus atributos privados e forneça métodos públicos para acessá-los e modificá-los. Por exemplo, crie métodos `obter_marca()`, `definir_marca()`, etc.
4. **Polimorfismo:**
   Crie uma classe chamada `Moto` que também herde da classe `Veiculo`. Implemente um método na classe `Moto` que seja diferente do método de exibição das classes `Veiculo` e `Carro`. Em seguida, crie objetos das três classes (`Veiculo`, `Carro` e `Moto`) e chame seus métodos de exibição para observar o polimorfismo em ação.

ATIVIDADE PRÁTICA 1

Crie um classe chamada cachorro com os atributos:
nome, raça, idade

ATIVIDADE PRÁTICA 2

Crie um classe chamada pessoa com os atributos: nome, idade, peso, gênero

ATIVIDADE PRÁTICA 3

Crie uma classe Empresa que permita gerenciar funcionários. Os funcionários devem ter informações como nome, cargo e salário. A empresa deve ser capaz de adicionar, remover e listar funcionários.

ATIVIDADE PRÁTICA 4

Crie uma classe Calculadora que tenha métodos para realizar operações matemáticas básicas (+ , - , * , /).

ATIVIDADE PRÁTICA 5

Crie uma classe chamada Fatura , a classe Fatura deve incluir os seguintes 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. Forneça um método chamado gerar_fatura que calcula o valor da fatura (isto é, multiplicar a quantidade pelo preço por item).

DESAFIO PRÁTICO

Aplicativo de hotelaria

Crie uma classe Hotel que permita gerenciar funcionários, reservas e quartos de hotel. Os funcionários devem ter informações como nome, função e salário. O hotel deve ser capaz de receber reservas, atribuí-las a quartos e calcular a conta final.

Material Complementar

- Exploração: Não tenha medo de explorar e testar diferentes códigos. A experimentação é uma grande aliada da aprendizagem.

- Perguntas: Faça perguntas, seja curioso! Entender o "porquê" das coisas ajuda a consolidar o conhecimento.

- Revisão: Revise o que aprendeu, tente explicar para si mesmo ou para outras pessoas. Ensinar é uma ótima forma de aprender.

- Prática: A prática leva à perfeição. Quanto mais exercícios fizer, mais fácil será lembrar e entender os conceitos.

SE LIGA NO CONTEÚDO DA PRÓXIMA AULA!

AULA 12 DE PYTHON:
POO II - HERANÇA

IN

INFINITY SCHOOL
VISUAL ART CREATIVE CENTER

Aula 12 - POO II - HERANÇA

HERANÇA

PRINCIPAIS CLASSES

TIPOS DE HERANÇAS

POLIMORFISMO

POLIMORFISMO E SUAS CLASSIFICAÇÕES

INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

ATÉ A PRÓXIMA...