



# PROGRAMAÇÃO ORIENTADA A OBJETOS



**mentorama.**  
@prof.felipeassuncao

# Introdução a POO



mentorama.

# O que é a POO?

- É um dos mais importantes paradigmas das linguagens de programação atuais
- Origem na década de 60
- Maior popularidade na década de 80

# POO



# Recursos e ferramentas

- Jupyter Notebook
- Editor de código de sua preferência

# Neste módulo

Aula 1 - Programação Orientada a Objetos

Aula 2 - Prática

Aula 3 - Exercícios

# 1. PROGRAMAÇÃO ORIENTADA A OBJETOS

mentorama.

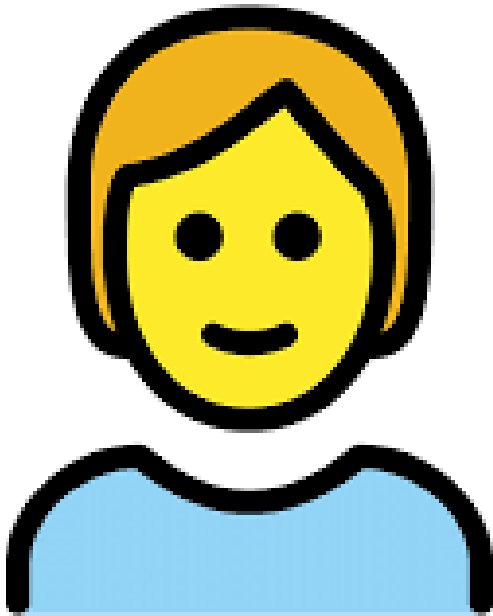
# Modelando um problema





# Modelando um problema

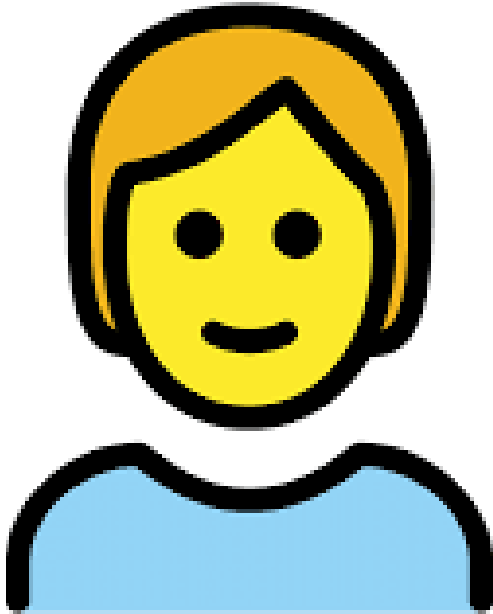
- Como podemos modelar uma pessoa?



PROPRIEDADES

MÉTODOS

# Modelando um problema



## PROPRIEDADES

- [nome: Maria; idade: 33]

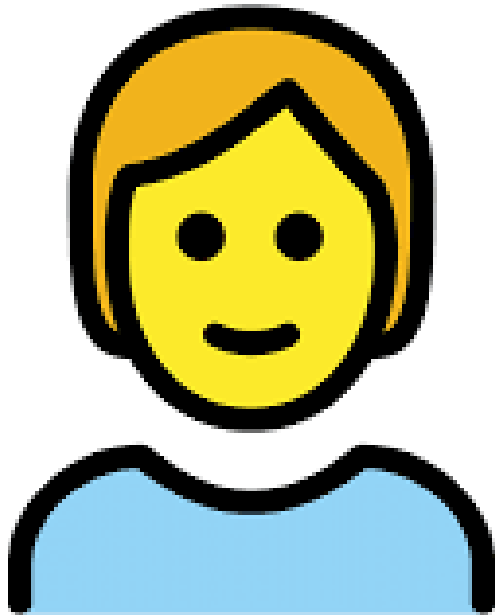
## MÉTODOS

- [andar, correr, deitar, sentar, balançar, falar, comer]

# Métodos vs. Funções

- O conceito de função e método se diferenciam só e somente só pelo retorno de valores.
- Toda função é um bloco de instrução que, possui um nome único que a identifica em seu escopo, pode receber parâmetros e **SEMPRE** retorna um valor.
- Um método é um bloco de instrução, possui um nome único que o identifica em seu escopo, pode receber parâmetros e **NUNCA** retorna valores.

# Declarando uma classe



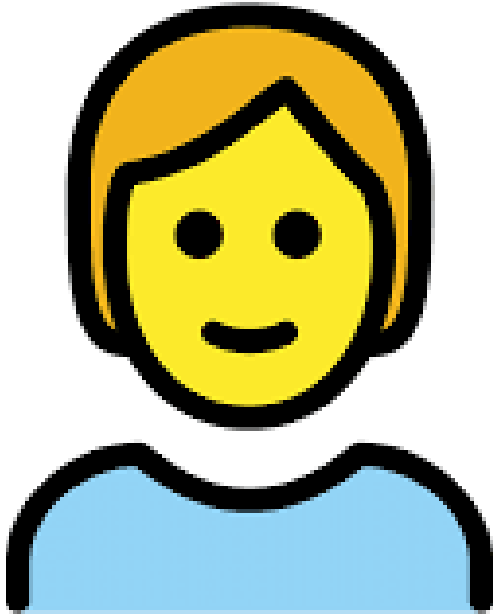
```
>>> class Pessoa:
>>>     def __init__(self, nome, idade):
>>>         self.nome = nome
>>>         self.idade = idade
>>> pessoa1 = Pessoa('Maria', '33')
>>> pessoa2 = Pessoa('José', '25')
>>> print(pessoa1.nome)
```

→ CRIA UM OBJETO PESSOA 1

→ CRIA UM OBJETO PESSOA 2

→ IMPRIME O VALOR DO NOME

# Declarando uma classe



mentorama.

```
>>> class Pessoa:
>>>     def __init__(self, nome, idade):
>>>         self.nome = nome
>>>         self.idade = idade
>>>     def setNome(self, nome):
>>>         self.nome = nome
>>>     def setIdade(self, idade):
>>>         self.idade = idade
>>>     def getNome(self):
>>>         return self.nome
>>>     def getIdade(self):
>>>         return self.idade
```

Diagram illustrating the structure of the `Pessoa` class:

- The `__init__` method is highlighted with a dashed blue box and labeled **CONSTRUTOR** (Constructor) with a blue arrow.
- The `setNome` method is highlighted with a dashed blue box and labeled **MÉTODO** (Method) with a blue arrow.
- The body of the `setNome` method, `self.nome = nome`, is highlighted with a dashed blue box and labeled **CORPO DO MÉTODO** (Body of the Method) with a blue arrow.

# Objetos

**EM PYTHON,  
TUDO É UM  
OBJETO**

- Um objeto é qualquer coisa, real ou abstrata sobre a qual armazenamos dados e realizamos operações que manipulam tais dados

`caneta.escrever()`

# Classes, objetos, atributos...

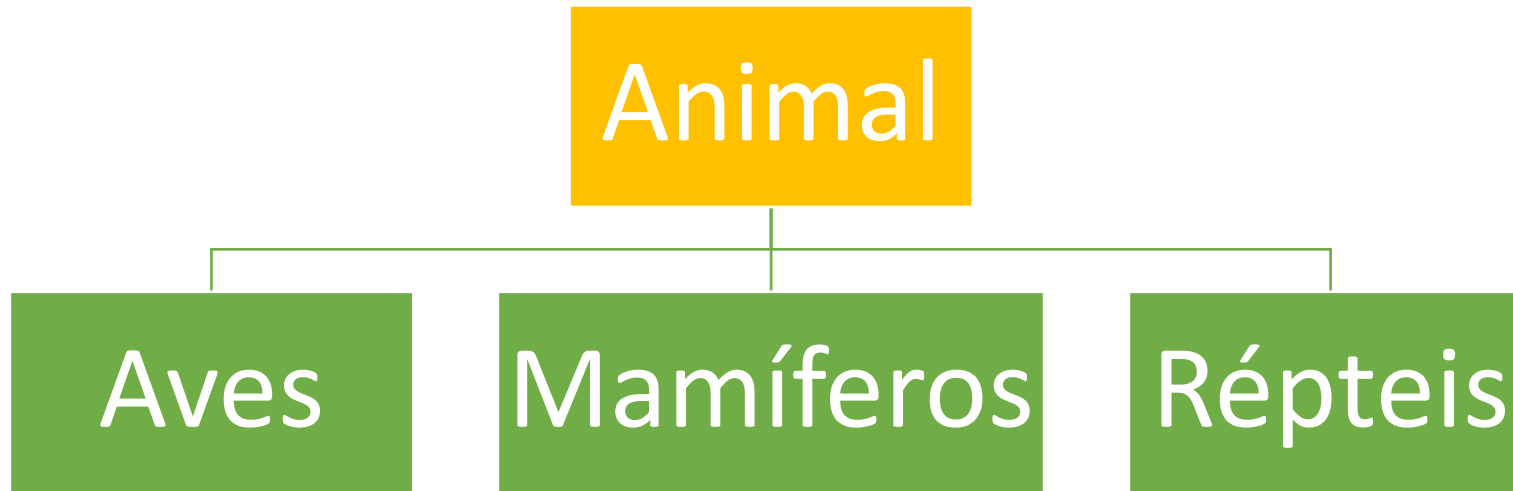
classe: pessoa



objetos

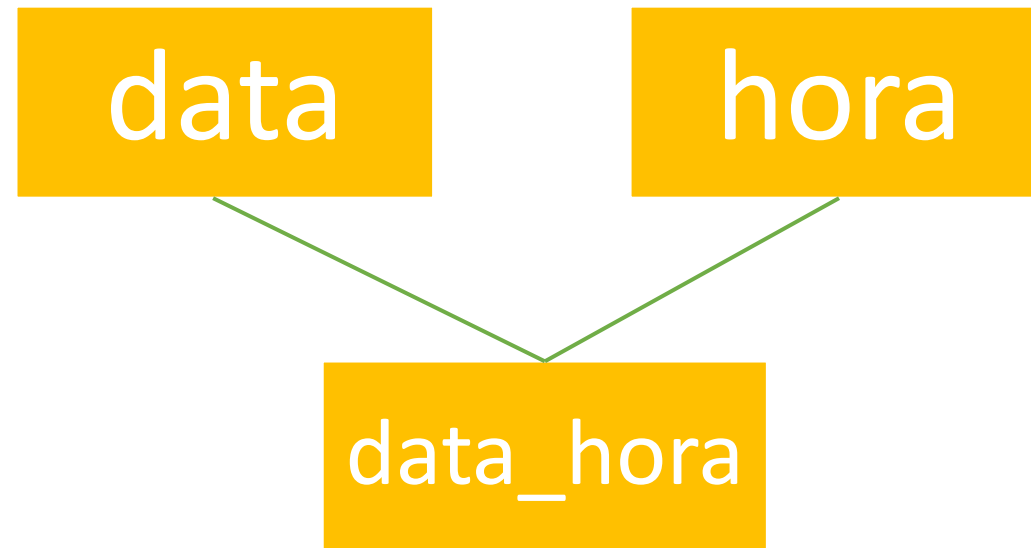
nome  
Idade  
identidade  
cpf

# Herança simples



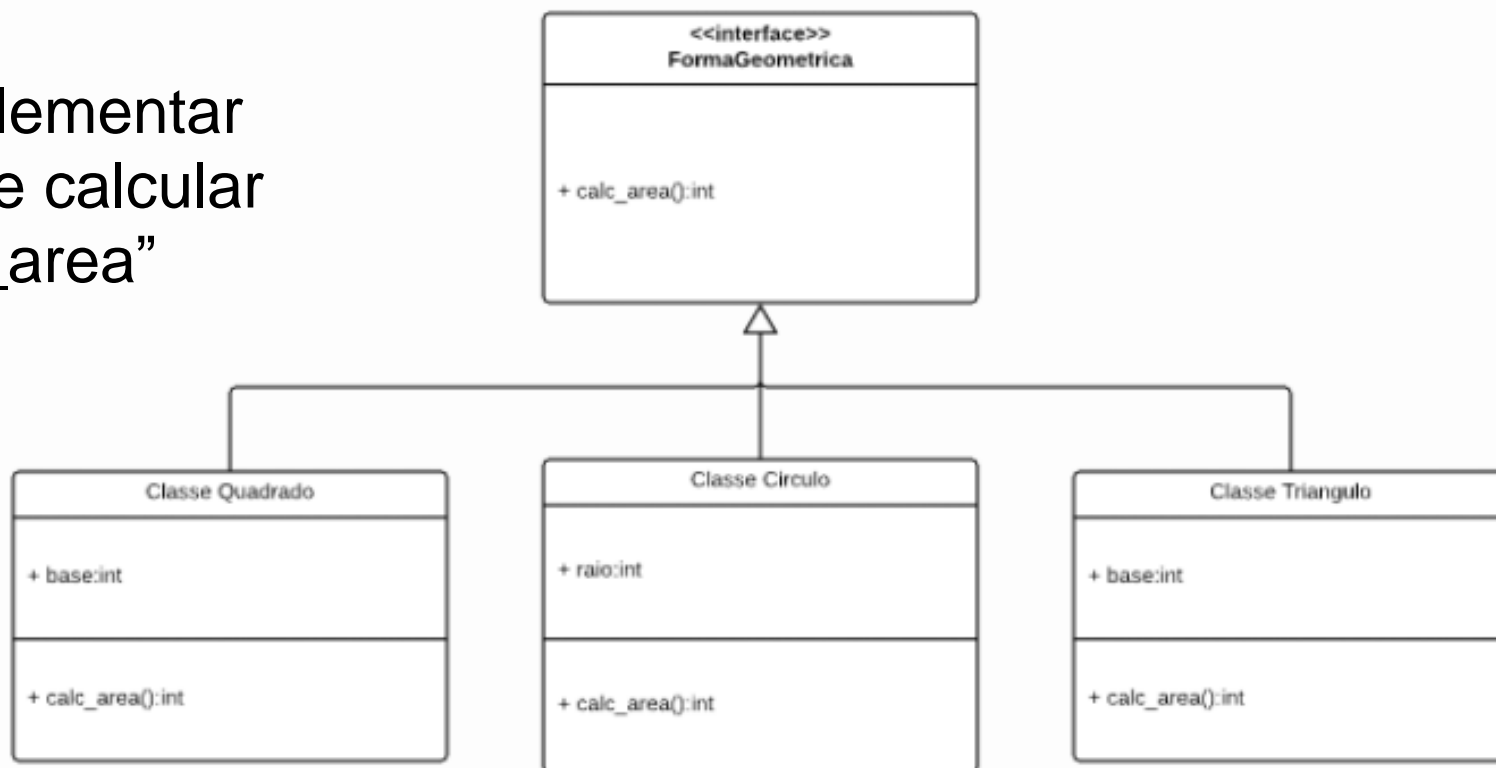


# Herança múltipla



# Polimorfismo

- A classe “Quadrado”, “Circulo” e “Triângulo” herdaram a classe “FormaGeometrica”
- Cada uma delas irá implementar a sua forma particular de calcular a área no método “calc\_area”



# Encapsulamento

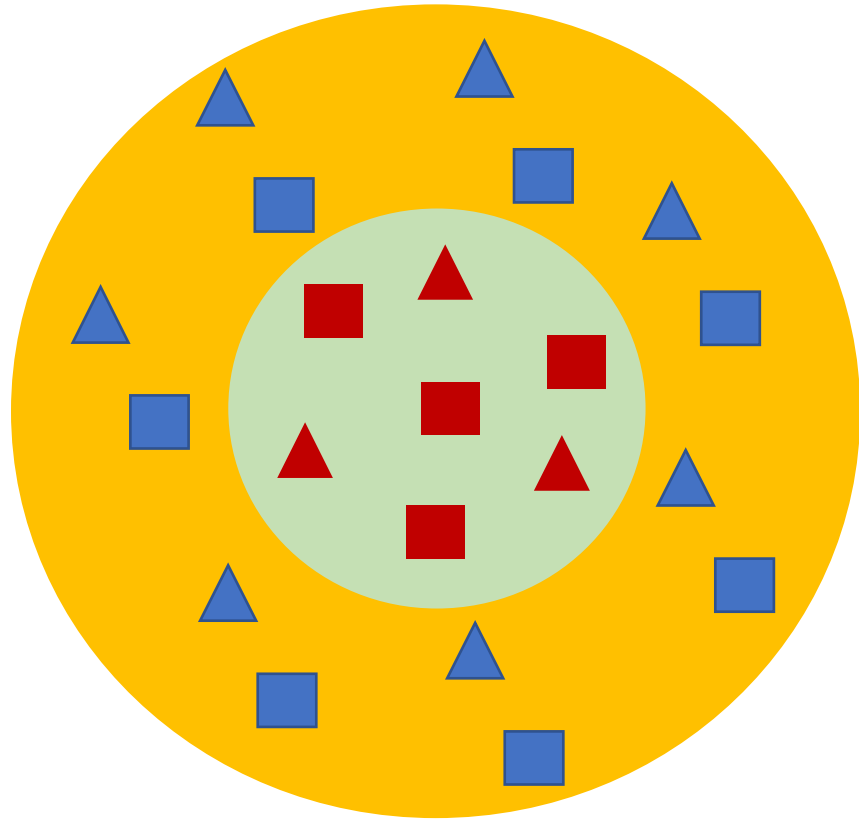
- É a proteção dos atributos ou métodos de uma classe

PUBLIC

PRIVATE

- Atributos ou métodos iniciados por no máximo dois sublinhados (underline) são privados e todas as outras formas são públicas

# Encapsulamento



- ▲ Métodos públicos
- ■ Atributos públicos
- ▲ Métodos privados
- ■ Atributos privados

# Resumo

- Abstração
- Modelagem de problemas
- Classes e Objetos
- Métodos



# 2. PRÁTICA

mentorama.



# Vamos praticar?

- Nesta prática iremos explorar como podemos trabalhar com classes, objetos, métodos dentre outros. Vamos codar?



# Resumo

- POO
- Classes e objetos
- Métodos
- Atributos
- Herança
- Encapsulamento
- Polimorfismo





# EXERCICIOS

mentorama.

