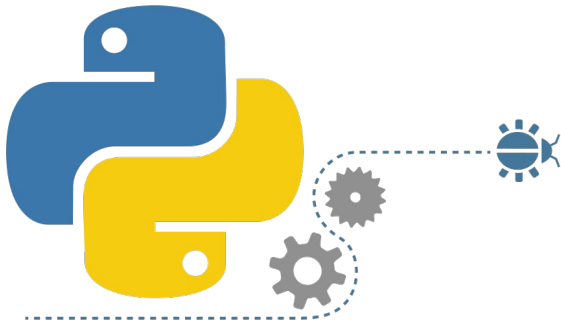




# Linguagem de Programação

Professor Anderson I. S. Abreu

# Classes e Métodos em Python



Tópicos da nossa aula:

1. Introdução a orientação a objetos;
2. Classes em Python;
3. Herança em Python.

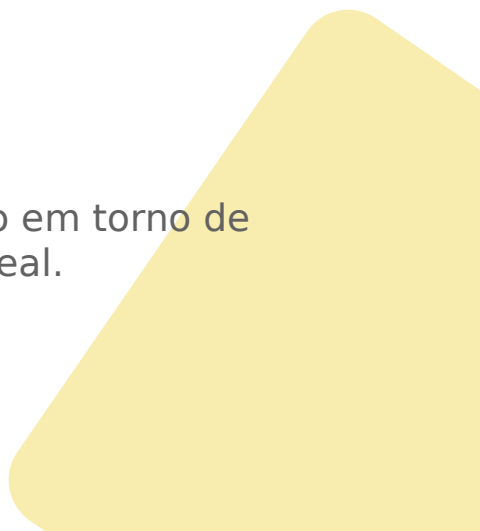


1

# Introdução a orientação a objetos

---

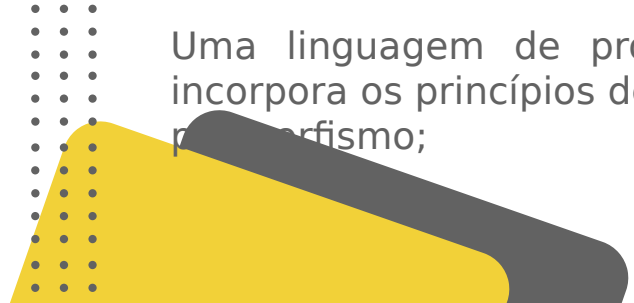
Um paradigma essencial na programação, que organiza o código em torno de objetos, cada um representando entidades do mundo real.



# A evolução constante.

A evolução é constante, com tecnologias que mudam rapidamente, mas os conceitos fundamentais do paradigma de programação orientada a objetos permanecem sólidos e cruciais;

Compreender esses conceitos é essencial, pois permite que você os implemente em qualquer tecnologia adotada pela sua empresa, independentemente de mudanças frequentes;



Uma linguagem de programação é considerada orientada a objetos quando incorpora os princípios de abstração e suporta o uso de encapsulamento, herança e polimorfismo;

# Classe vs Objeto

**Classe:** Pessoa

**Atributos** (Dados):

**Nome:**

**Idade:**

**Gênero:**

**Métodos** (Comportamentos):

**Cumprimentar:** Saúda como "Olá, meu nome é."

**Aniversário:** Aumenta a idade em 1.

**Objeto 1:** Pessoa 1

**Atributos** (Dados):

**Nome:** João

**Idade:** 30

**Gênero:** Masculino

**Métodos** (Comportamentos):

**Cumprimentar:** Saúda como "Olá, meu nome é João."

**Aniversário:** Aumenta a idade em 1.



2

# Classes em Python

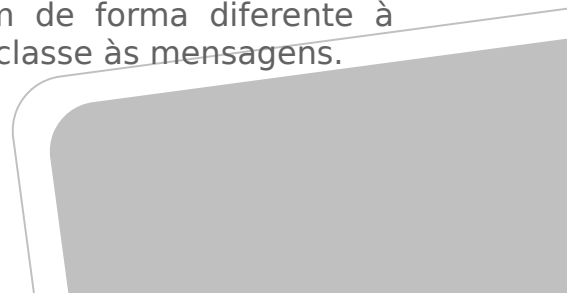
---

São os modelos para a criação de objetos, e como definir atributos e métodos dentro delas.



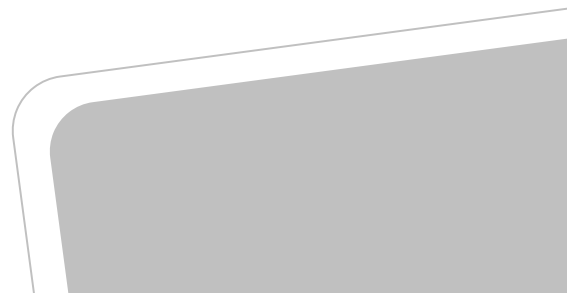
# Componentes Principais de uma classe

- Atributos: São os dados que representam o estado do objeto, como nome e idade.
- Métodos: Definem o comportamento do objeto, sendo as ações que ele pode executar, como cumprimentar ou fazer login.
- Encapsulamento: Combina atributos e métodos em uma entidade, permitindo controlar o acesso a atributos por meio de métodos.
- Herança: Permite que uma classe herde atributos e métodos de outra, promovendo o reuso de código e a organização hierárquica, como na relação entre as classes pessoa, funcionário e cliente.
- Polimorfismo: Refere-se à capacidade de várias classes responderem de forma diferente à mesma mensagem, graças à herança e às respostas específicas de cada classe às mensagens.



# Criando nossa Classe Pessoa

Partiu pro código!







3

# Herança em Python.

---

Permite que classes-filhas herdem atributos e métodos de classes-pai, promovendo a reutilização de código e a criação de hierarquias de classes.



# Até onde vamos?

A herança é um dos pilares fundamentais da programação orientada a objetos;

Ela permite que uma classe (a classe-filha) herde características e comportamentos de outra classe (a classe-pai);

Em Python, essa técnica é amplamente suportada e flexível, permitindo que uma classe-filha herde de múltiplas classes-pai, um conceito conhecido como herança múltipla

```
class ClasseFilha(ClassePai):
```

```
# Definição da classe-filha
```

```
class ClasseFilha(ClassePai1, ClassePai2, ClassePai3):
```

```
# Definição da classe-filha
```



# Criando nossa Classe Pessoa

- 1.Reutilização de Código: A herança permite que você reutilize o código existente, aproveitando a estrutura e funcionalidade de classes-pai em suas subclasses.
- 2.Extensibilidade: Você pode estender ou adicionar comportamentos específicos às classes filhas sem modificar as classes-pai, mantendo a coesão e a organização do código.
- 3.Hierarquia de Classes: A herança permite criar uma hierarquia de classes, na qual classes filhas podem herdar características comuns de classes-pai e, por sua vez, serem herdadas por outras classes.

Bora pro código!






# Aplicando a aula!

---

Suponha agora, que você, precisa criar um algoritmo que a partir de características imputadas, mostre um resumo e status de um veículo. E ainda, você deve fazer algo semelhante para uma subdivisão dessa classe veículos. Vamos juntos para resolver isso?



# Obrigado!

