

The background features abstract geometric shapes in blue and orange. At the top center is a blue semi-circle. To its left are two orange triangles. On the right is a large orange circle. The bottom left is composed of several overlapping blue and orange triangles. A thin orange horizontal line spans the width of the slide, positioned below the main title.

PROGRAMAÇÃO ORIENTADA A OBJETOS

prof. Arthur Silva

conteúdo

1

Classes

2

Objetos

3

Atributos

4

Métodos

5

Criando Classes e Objetos

6

Exercícios



orientação a objetos

- Definição: POO é um paradigma de programação que organiza o design de software em torno de objetos e classes.
- Conceitos-Chave:
 - Classes: Modelos ou templates para criar objetos.
 - Objetos: Instâncias de classes que representam entidades do mundo real.
 - Abstração: Simplificar a complexidade ao modelar apenas os aspectos relevantes de um objeto.
 - Encapsulamento: Agrupamento de dados e métodos que operam nesses dados.
 - Herança: Criação de novas classes a partir de classes existentes.
 - Polimorfismo: Uso de uma única interface para representar diferentes tipos.



orientação a objetos

- Por que POO?
 - Modularidade: Divide problemas complexos em componentes menores e reutilizáveis.
 - Reutilização: Reutiliza classes e objetos em diferentes programas.
 - Manutenibilidade: Facilita a atualização e manutenção do código.

o que é uma classe?

- Definição: Uma classe é um modelo ou template para criar objetos. Ela define as propriedades (atributos) e comportamentos (métodos) que os objetos terão.
- Abstração: A classe abstrai os detalhes desnecessários, focando apenas nos aspectos essenciais do objeto.

```
class Carro:
    def __init__(self, marca, modelo, ano):
        self.marca = marca
        self.modelo = modelo
        self.ano = ano

    def ligar(self):
        print(f"{self.marca} {self.modelo} está ligando.")

    def desligar(self):
        print(f"{self.marca} {self.modelo} está desligando.")
```

o que é um objeto?

- Definição: Um objeto é uma instância de uma classe. Ele representa uma entidade específica com seus próprios dados e comportamentos.
- Exemplo: Se Carro é uma classe, então meu_carro = Carro("Toyota", "Corolla", 2022) é um objeto.
- Abstração no Objeto: O objeto representa uma instância concreta da abstração definida pela classe.

```
# Criando um objeto
meu_carro = Carro("Toyota", "Corolla", 2022)


# Acessando atributos
print(meu_carro.marca) # Saída: Toyota

# Chamando métodos
meu_carro.ligar()      # Saída: Toyota Corolla está ligando.
```



Abstração

- Definição: Abstração é o processo de esconder os detalhes complexos de um objeto e mostrar apenas as características essenciais.
- Por que usar abstração?
- Simplifica o entendimento e o uso de objetos.
- Foca no "o que" o objeto faz, não no "como" ele faz.
- Exemplo: Um motorista não precisa saber como o motor de um carro funciona para dirigir. Ele só precisa saber como usar o volante, os pedais e o câmbio (abstração do carro).



```
class ContaBancaria:
    def __init__(self, saldo):
        self.__saldo = saldo # Atributo privado (abstração)

    def depositar(self, valor):
        self.__saldo += valor
        print(f"Depósito de R${valor} realizado. Novo saldo: R${self.__saldo}.")

    def sacar(self, valor):
        if valor <= self.__saldo:
            self.__saldo -= valor
            print(f"Saque de R${valor} realizado. Novo saldo: R${self.__saldo}.")
        else:
            print("Saldo insuficiente.")

    def ver_saldo(self):
        print(f"Saldo atual: R${self.__saldo}.")
```




Exercícios

1. Crie uma classe **Conta Bancária** com os seguintes atributos:
 - a. titular
 - b. saldo
2. Na classe **Conta Bancária**, crie métodos para exibir os atributos, como:
 - a. **ver_saldo()**
 - b. **depositar()**
 - c. **sacar()**.
3. Instancie (crie) um objeto da classe **Conta Bancária** e utilize os métodos da classe.



Exercícios

1. Crie uma classe **Aluno** com os seguintes atributos:
 - a. nome
 - b. matrícula
 - c. notas (uma lista para armazenar as notas do aluno)
2. Na classe Aluno, crie métodos para:
 - a. **adicionar_nota(nota)**: Adiciona uma nota à lista de notas.
 - b. **calcular_media()**: Calcula e retorna a média das notas.
 - c. **exibir_info()**: Exibe o nome, matrícula e média do aluno.
3. Instancie um objeto da classe Aluno e teste os métodos.

The background features abstract geometric shapes. A dark blue triangle is in the top-left corner. Two large orange shapes, one on the left and one on the right, point towards the center. A smaller, darker orange triangle is nested within the left orange shape. The central area is white.

MUITO OBRIGADO!!