# Estrutura de Dados em Python

Prof. Nisston Moraes Tavares de Melo

uniesp
Centro Universitário

# A complexidade de um algoritmo

- **Big O Notation**
  - Fornece uma maneira de classificar algoritmos em categorias de complexidade com base em como seu desempenho se comporta conforme o tamanho dos dados aumenta. A notação utiliza símbolos como "O" (ordem de) seguido por uma função que expressa a complexidade do algoritmo.
  - Por exemplo, O(1), O(log n), O(n), O(n^2), O(2^n), entre outros.

# Resultado da complexidade

| | n=> | 10 |
|---|---|---|
| O(1) | 1 | 1 |
| O(n) | n | 10 |
| O(log(n)) | Log(n) | 1 |
| O(n^2) | n^2 | 100 |
| O(2^n) | 2^n | 1024 |

# Algoritmos

```python
def soma1(n):
    soma = 0
    for i in range(n + 1):
        soma = soma + i
    return soma
```

```
O resultao foi de..: 5000050000
Tempo de inicio....: 1692623270.97544 segundos
Tempo de termino...: 1692623270.98363 segundos
Tempo de execução..: 0.0081813335 segundos
==========================================================
```

**Função 1**
**O(função 1) → n(passos)**
**Atribuição da variável + quantidade n = (n + 1)**

```python
def soma2(n):
    return (n * (n + 1)) / 2
```

```
O resultao foi de..: 5000050000.0
Tempo de inicio: 1692623270.99162 segundos
Tempo de termino: 1692623270.99162 segundos
Tempo de execução: 0.0000000000 segundos
```

**Função 2**
**O(função 2) → 3 passos**
**Somatório + multiplicação e divisão = 3**

# Estrutura de dados em Python

Dra. Ana Carolina Costa de Oliveira
prof2121@iesp.edu.br

EXPERT ACADEMY

Aula 03

# Principais elementos da OO

- Classe: Modelo do objeto do mundo real sendo representado computacionalmente.
- Atributo: Característica do objeto.
- Método: Comportamento do objeto (Funções).
- Construtor: Método especial utilizado para criar os objetos.
- Objeto: Instância da classe.

# Classes, Atributos e Métodos

- Um **Classe** é uma representação de um item do mundo real, físico ou abstrato, na forma de um tipo de dado personalizado.
- As classes possuem estruturas internas chamadas de **Atributos** e de **Métodos**.
- **Atributos** são usados para armazenar os dados dos objetos de uma classe.
- **Métodos** são as operações, ou funções que a instância de classe vão executar ou sofrer.
- Uma instância de classe é chamada de **Objeto**.

https://pt.depositphotos.com/stock-photos/diagrama-de-classe.html?filter=all

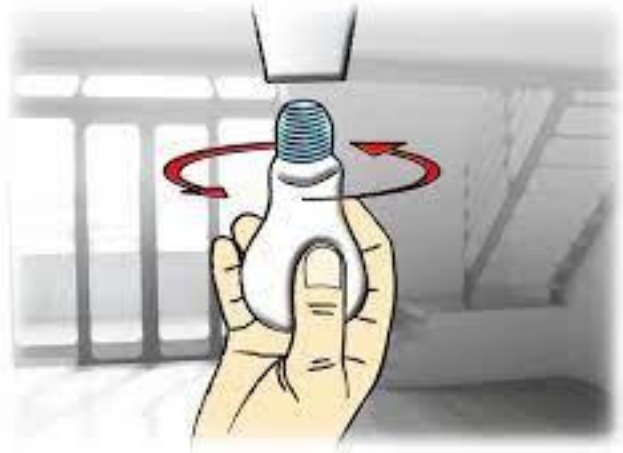
# Exercício

- Trocar uma lâmpada

A classe Lampada podem conter quais atributos ?

# Exercício

- Trocar uma lâmpada



A classe Lampada podem conter quais métodos ?

# Exemplo

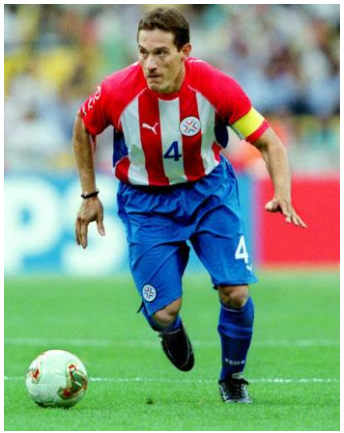
```
class Lampada:
    pass

Lamp = Lampada ()
print(type(Lampada))
```

# Exemplo

**Classe**

Representa do mundo real sendo representado computacionalmente.

**Exemplo:** Jogador



**Atributo**

Representa características de uma classe.

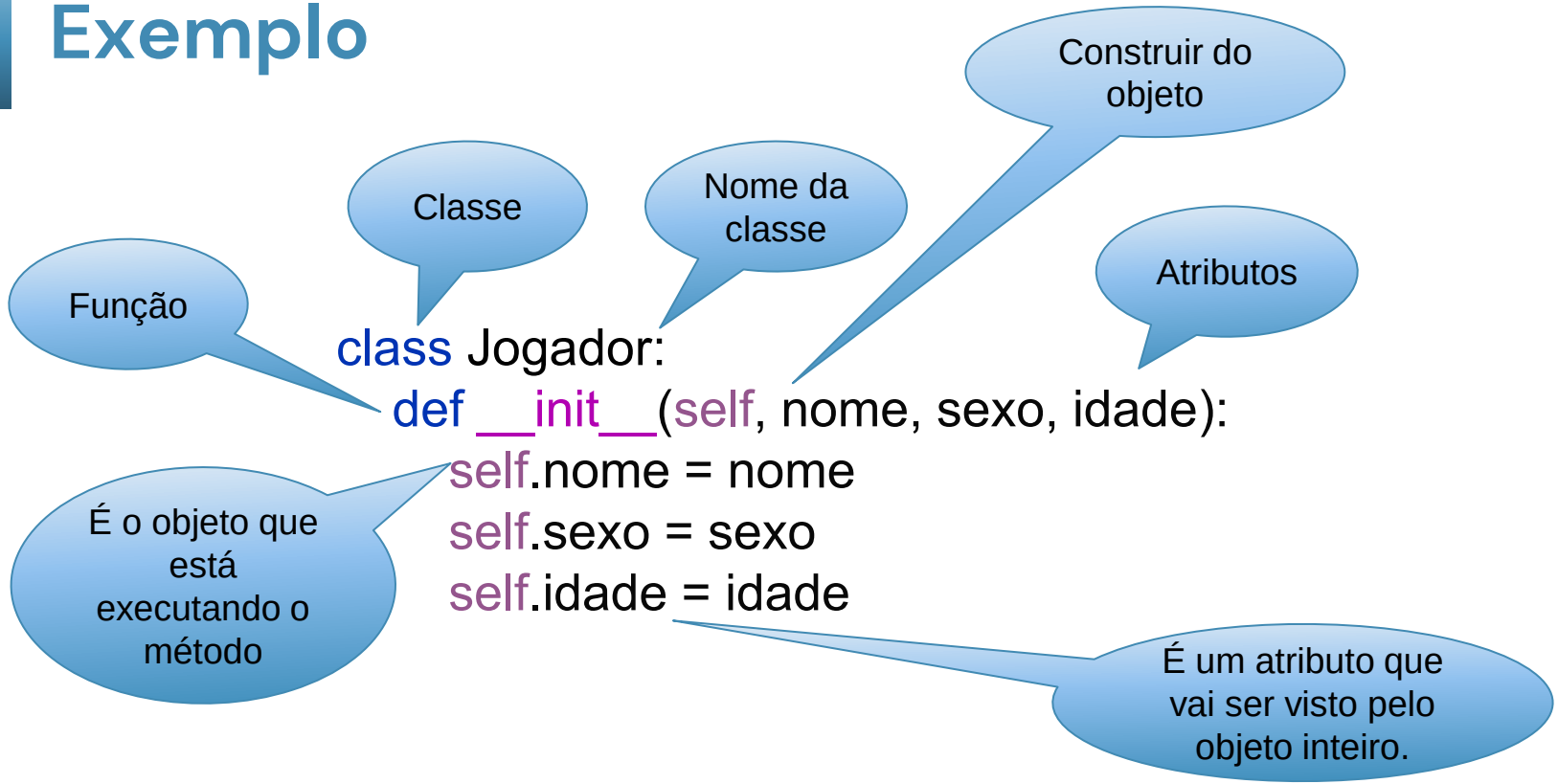
**Exemplo:** Jogador (nome, sexo, idade etc.).

**Método**

Representa atividades que um objeto de uma classe pode executar.

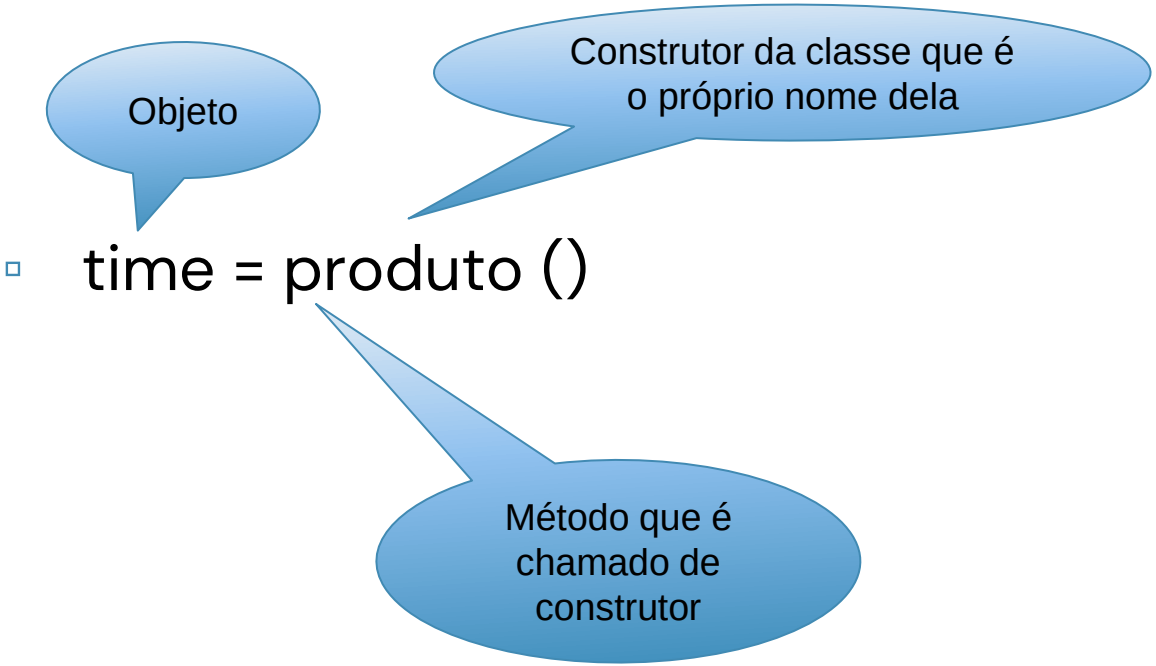
**Exemplo:** Jogador (correr, driblar, chutar).

# Exemplo




**Obs:.** Nome da classe por convenção é iniciado por letra maiúscula.
**Dica:** Em computação não utilizamos acentuação, caracteres especiais, espaços ou similares para nomes de classes, atributos, métodos, arquivos etc.

# Exemplo


Objeto
Construtor da classe que é o próprio nome dela
time = produto ()
Método que é chamado de construtor

# Exemplo

```
class Produto:


    def __init__(self, nome, descricao, valor):
        self.nome = nome
        self.descricao = descricao
        self.valor = valor


p1 = Produto('Uva', 'Isabel', '6,00')
print(p1.nome, p1.descricao, p1.valor)
print(p1)
p2 = Produto('Maça', 'Argentina', '8,00')
print(p2.nome, p2.descricao, p2.valor)
print(p2)
```

```
class Produto:

    def __init__(self, nome, descricao, valor):
        self.nome = nome
        self.descricao = descricao
        self.valor = valor
        self.valor_imposto = 0.0

    def calcular_valor_imposto(self):
        self.valor_imposto = float(self.valor) * 0.05
        return self.valor_imposto

    def mostra_dados(self):
        print(f'Nome: {self.nome}')
        print(f'Descrição: {self.descricao}')
        print(f'Valor: {self.valor}')
        print(f'Imposto: {float(self.valor_imposto)}')
        print(f'Valor total a Pagar: {float(self.valor_imposto)+float(self.valor)}')

p1 = Produto('Uva', 'Isabel', 6.00)
p1.calcular_valor_imposto()
p1.mostra_dados()
```

# Exercício

- **Crie uma classe chamada aluno com os seguintes atributos:**
  - Nome
  - Nota 1
  - Nota 2
  - Crie um construtor para a classe (__init__)

# Exercício

- **Crie as seguintes funções (métodos):**
  - Calcula média, retornando a média aritmética entre as notas
  - Mostra dados, que somente imprime o valor de todos os atributos
  - Resultado, que verifica se o aluno está aprovado ou reprovado (se a média for maior ou igual a 6.0, o aluno está aprovado)

# Resultado

```
class Aluno:
    def __init__(self, nome, nota1, nota2):
        self.nome = nome
        self.nota1 = nota1
        self.nota2 = nota2
        self.media = 0.0

    def calcular_media(self):
        self.media = (self.nota1 + self.nota2) / 2
        return self.media

    def mostra_dados(self):
        print(f'Nome: {self.nome}')
        print(f'Nota1: {self.nota1}')
        print(f'Nota2: {self.nota2}')
        print(f'Média: {self.media}')
```

# Resultado

```
def resultado(self):
        if self.media >= 7:
            print('Aluno(a) aprovado (a)')
        else:
            print('Aluno(a) reaprovado (a)')


aluno1 = Aluno('Carol', 9.0, 8.7)
aluno2 = Aluno('Paula', 9.0, 8.7)
aluno3 = Aluno('Clara', 9.0, 8.7)

aluno2.calcular_media()
aluno2.mostra_dados()
aluno2.resultado()
```

# Vamos exercitar?

# Exercício

1. Crie uma classe chamada Carro com um método acelerar() que imprime "O carro está acelerando!" e um método frear() que imprime "O carro está freando!".
2. Crie uma classe chamada Pessoa com um método cumprimentar(nome) que imprime "Olá, [nome]!".
3. Crie uma classe chamada Calculadora com métodos soma(a, b), subtracao(a, b), multiplicacao(a, b) e divisao(a, b) que realizam as operações correspondentes e retornam o resultado.
4. Crie uma classe chamada Retangulo com métodos calcular_area(largura, altura) e calcular_perimetro(largura, altura) que retornam a área e o perímetro de um retângulo, respectivamente.


Dúvidas?

# Gabarito

```
class Carro:
    def acelerar(self):
        print("O carro está acelerando!")

    def frear(self):
        print("O carro está freando!")

meu_carro = Carro()
meu_carro.acelerar()
meu_carro.frear()
```

# Gabarito

```
class Pessoa:
    def cumprimentar(self, nome):
        print(f"Olá, {nome}!")

pessoa = Pessoa()
pessoa.cumprimentar("Maria")
```

# Gabarito

```python
class Calculadora:
    def soma(self, a, b):
        return a + b

    def subtracao(self, a, b):
        return a - b

    def multiplicacao(self, a, b):
        return a * b

    def divisao(self, a, b):
        return a / b

calc = Calculadora()
print(calc.soma(10, 5))
print(calc.subtracao(15, 7))
print(calc.multiplicacao(4, 3))
print(calc.divisao(20, 4))
```

# Gabarito

```
class Retangulo:
    def calcular_area(self, largura, altura):
        return largura * altura

    def calcular_perimetro(self, largura, altura):
        return 2 * (largura + altura)

retangulo = Retangulo()
print(retangulo.calcular_area(5, 3))
print(retangulo.calcular_perimetro(5, 3))
```


Dúvidas?

# Lista de Exercício.


uniesp
Centro Universitário