
# Estrutura de Dados Avançadas

## Engenharia de Software

Aula 01

Prof. Alvaro Leiroz

# Estrutura de Dados Avançadas

Alvaro Leiroz

alvaro.leiroz@univassouras.edu.br

## Apresentação

- Pós-graduando em Gestão de Projetos e Negócios em TI pela IFRJ
- Pós-graduado em Análise e Desenvolvimento de Sistemas, e Tecnologias Aplicadas na Educação para Faculdade Descomplica
- Professor Universitário (UniVassouras)
- Analista de Sistemas – Gerência de Sistemas FUSVE
- Coordenador da Gameficada

# Estrutura de Dados Avançadas

**Ementa / Conteúdo Programático**

Definição de estruturas de dados; Revisão de vetores, matrizes e registros; Complexidade de algoritmos; Ordem de funções; Análise teórica da complexidade; Taxa de crescimento das funções; A notação O (Big-Oh); Análise assintótica das funções; Taxa de crescimento das funções; Indução matemática; Recursividade; Recorrências; Pilhas; Filas; Fila linear; Fila circular; Listas; Lista encadeada; Lista duplamente encade

Árvores, Algoritmos de busca e ordenação, grafos

# Estrutura de Dados Avançadas

UNIVERSIDADE DE VASSOURAS

**Ementa / Conteúdo Programático**

Trabalho para composição da P1 (2,0 pontos)

Avaliação individual Prática P1 (8,0 pontos) – Realizada pela plataforma de Ensino e/ou Prova Fácil

Trabalho para composição da P2 (2,0 pontos)

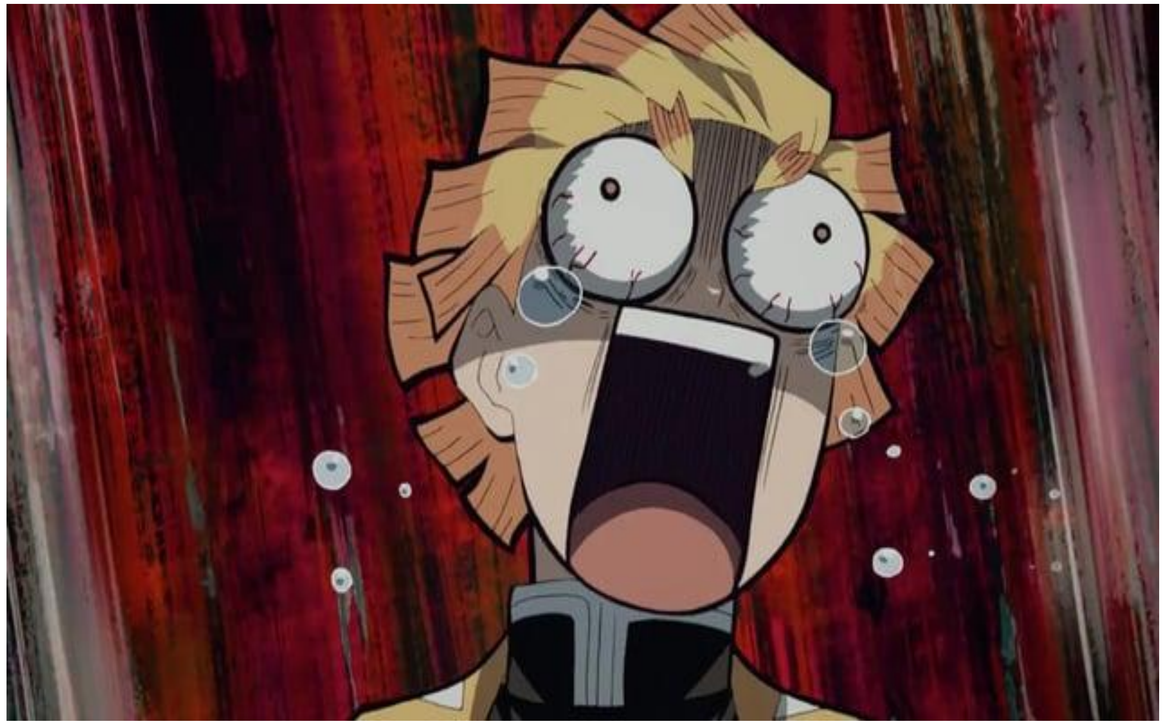
Avaliação individual Prática P2 (8,0 pontos) – Realizada pela plataforma de Ensino e/ou Prova Fácil

Segunda chamada / Exame Final / Segunda época (10,0 pontos) – Realização de uma única Avaliação individual Prática – Realizada pela plataforma de Ensino e/ou Prova Fácil

# Estrutura de Dados Avançadas

UNIVERSIDADE DE VASSOURAS

**BORA INICIAR?**



UNIVERSIDADE DE VASSOURAS

# PYTHON

## Contexto Histórico

Python foi criado por Guido van Rossum no final dos anos 1980 e lançado oficialmente em 1991. O nome "Python" foi inspirado no grupo de comédia britânico Monty Python, refletindo o humor e a simplicidade que Guido queria para a linguagem.

UNIVERSIDADE DE VASSOURAS

**PYTHON**

## Filosofia do Python

A linguagem foi projetada para ser legível e simples, com foco na produtividade do programador.

O Zen do Python (acessível digitando **import this** no interpretador) resume essa filosofia, destacando princípios como "Legibilidade conta" e "Simples é melhor que complexo".

UNIVERSIDADE DE VASSOURAS

**PYTHON**

## Versões do Python

**Python 2.x** foi a versão dominante por muitos anos, mas foi descontinuada em 2020.

**Python 3.x** é a versão atual e recomendada, com melhorias significativas e suporte contínuo.

Versão em desenvolvimento 3.14

Versão estável 3.13

## CARACTERÍSTISCAS PYTHON

### Linguagem de Alto Nível

Python abstrai muitos detalhes de baixo nível, como gerenciamento de memória, permitindo que o programador se concentre na lógica do programa.

Linguagens de programação de alto nível são mais próximas da linguagem humana, enquanto as de baixo nível estão mais próximas da linguagem de máquina.

## CARACTERÍSTISCAS PYTHON

### Linguagem Interpretada

O código Python é executado linha por linha por um interpretador, sem necessidade de compilação prévia.

O intérprete lê o código linha por linha durante a execução do programa.

## CARACTERÍSTISCAS PYTHON

### Tipagem Dinâmica e Forte

**Tipagem Dinâmica:**O tipo de uma variável é determinado em tempo de execução, e não em tempo de compilação. Não é necessário declarar o tipo das variáveis explicitamente.

**Tipagem Forte:**Garante que apenas funções e operadores definidos explicitamente para um tipo podem ser aplicados a ele, então, operações entre tipos incompatíveis (ex: **int** + **str**) geram erros.

UNIVERSIDADE DE VASSOURAS

## CARACTERÍSTISCAS PYTHON

### Linguagem Programação Multiparadigma

Suporta paradigmas de programação procedural, orientada a objetos e funcional.

**Imperativo**(estruturado)

```
1 numeros = [1,2,3,4,5,6,7,8,9,10]
2 numeros_pares = []
3
4 for numero in numeros:
5     if (numero % 2 == 0):
6         numeros_pares.append(numero)
7 print(numeros_pares)
```

**Declarativo**(funcional)

```
1 numeros = [1,2,3,4,5,6,7,8,9,10]
2 numeros_pares = list(filter(lambda numero: numero % 2 == 0, numeros))
3
4 print(numeros_pares)
```

## CARACTERÍSTISCAS PYTHON

**Paradigma procedural**

Baseia-se em procedimentos, que são blocos de código que podem ser chamados por um programa principal
As instruções são executadas na ordem em que aparecem, de cima para baixo

## CARACTERÍSTISCAS PYTHON

### Paradigma Orientado a Objetos

A OO determina que o código deve ser modelado de forma a se aproximar ao mundo real, e que o mesmo em execução no computador tenha tal modelagem representada por uma estrutura de objetos, características e ações.

## CARACTERÍSTISCAS PYTHON

### Paradigma Funcional

O paradigma funcional difere dos demais citados por não ser derivado da programação imperativa e sim da declarativa, onde o objetivo é declarar ao computador o resultado esperado, e não o passo a passo para construção deste resultado.

## CARACTERÍSTISCAS PYTHON

### ECOSSISTEMA RICO

Possui uma vasta biblioteca padrão e pacotes de terceiros para diversas aplicações.

UNIVERSIDADE DE VASSOURAS

## PYTHON

## VARIÁVEIS E CONSTANTES

- Por meio de variáveis que um algortimo “guarda” os dados do problema
- Todo dado que tem a possibilidade de ser alterado no decorrer do tempo deverá ser tratado como uma variável
- Quando um dado não tem nenhuma possibilidade de variar no decorrer do tempo, deverá ser tratado como constante

UNIVERSIDADE DE VASSOURAS

## PYTHON

## VARIÁVEIS

- Inteiros(int): valores positivos ou negativos, que não possuem uma parte fracionária. Exemplos: 1, 30, 40, 12, -50
- Float (float): valores positivos ou negativos, que podem possuir uma parte fracionária. Exemplos: 1.4, 6.7, 10.3, 100, -47
- Caracteres (char ou string): qualquer elemento presente no teclado. Exemplos: “Maria”, “João”, ‘M’, ‘F’
- Lógico (bool): verdadeiro ou falso. Exemplos: true, false, 1, 0

UNIVERSIDADE DE VASSOURAS

**PYTHON**

## CONSTANTES

É uma variável que deve ser tratada como um valor fixo. Embora a linguagem não tenha um mecanismo específico para declarar constantes, há uma convenção para identificá-las.

Por exemplo, PI = 3.14159 ou GRAVIDADE = 9.8

UNIVERSIDADE DE VASSOURAS

**PYTHON**

# EXEMPLOS

## EXERCÍCIO 01

1. Ler dois números inteiros, executar e mostrar o resultado das seguintes operações: adição, subtração, multiplicação e divisão

2. Efetuar o cálculo da quantidade de litros de combustível gasto em uma viagem, utilizando um automóvel que faz 12 Km por litro. Para obter o cálculo, o usuário deve fornecer o tempo gasto na viagem e a velocidade média durante ela. Desta forma, será possível obter a distância percorrida com a fórmula DISTANCIA = TEMPO * VELOCIDADE. Tendo o valor da distância, basta calcular a quantidade de litros de combustível utilizada na viagem, com a fórmula: LITROS_USADOS = DISTANCIA / 12. O programa deve apresentar os valores da velocidade média, tempo gasto na viagem, a distância percorrida e a quantidade de litros utilizada na viagem

UNIVERSIDADE DE VASSOURAS

## Operadores lógicos

**Operador and / &**
O operador and retorna True se todas as condições forem verdadeiras. Caso qualquer uma das condições seja falsa, ele retorna False.

**Operador or / |**
O operador or retorna True se qualquer uma das condições for verdadeira. Ele só retorna False se todas as condições forem falsas.

**Operador not**
O operador not inverte o valor de uma condição. Se a condição for True, ele retorna False, e vice-versa.

UNIVERSIDADE DE VASSOURAS

## Operadores de comparação

**Operador ==**
O operador de igualdade verifica se dois valores são iguais.

**Operador !=**
Esse operador verifica se dois valores são diferentes.

**Operadores >, <, >=, <=**
Esses operadores comparam grandezas entre dois valores:
> verifica se o valor à esquerda é maior que o valor à direita.
< verifica se o valor à esquerda é menor que o valor à direita.
>= verifica se o valor à esquerda é maior ou igual
<= verifica se o valor à esquerda menor ou igual ao valor à direita

## Operadores de atribuição

**Operador =**
O operador igual é de atribuição

**Operador +=**
Muito utilizado em estruturas de repetição.

**Operadores -=, *=, /=**
Dificilmente verão

**Operador := (operador walrus)**
A partir da versão 3.8, o Python introduziu o Operador Walrus, também conhecido como operador de atribuição em expressões. Esse operador permite atribuir um valor a uma variável ao mesmo tempo em que a utiliza em uma expressão. Isso torna o código mais compacto e eficiente, especialmente em loops e condições.
Normalmente, atribuímos valores a variáveis em uma linha e depois utilizamos esses valores em outra. Com o operador :=, podemos fazer isso simultaneamente.

## Operador de Identidade – Operador is

No Python, o operador de identidade é o is. Esse operador verifica se dois objetos são o mesmo objeto na memória (ou seja, ocupam o mesmo endereço de memória), em vez de apenas comparar seus valores.
É importante não confundir o operador is com o operador ==. O == verifica se os valores de dois objetos são iguais, enquanto o is verifica se os dois objetos são idênticos em termos de localização na memória.
O operador is é particularmente útil ao trabalhar com objetos mutáveis, como listas e dicionários. Por exemplo, podemos ter duas listas com os mesmos valores, mas essas listas são objetos diferentes na memória do computador.

```
1.  a = [1, 2, 3]
2.  b = a
3.  c = [1, 2, 3]
4.
5.  print("a e b são o mesmo objeto:")
6.  print(a is b)
7.  print("a e c são o mesmo objeto:")
8.  print(a is c)
```

```
a e b são o mesmo objeto:
True
a e c são o mesmo objeto:
False
```

# Estrutura de Dados Avançadas

UNIVERSIDADE DE VASSOURAS

## Operador de Pertencimento (Membership) – Operador in

O operador de pertencimento no Python é o operador in. Ele é usado para verificar se um elemento está presente em uma coleção, como listas, dicionários ou strings. Esse operador é bastante intuitivo e amplamente utilizado, especialmente para validação de dados.
O operador in realiza essa verificação retornando True se o elemento estiver na coleção e False caso contrário.
Por exemplo, ele pode ser usado para verificar se determinados funcionários estão incluídos na lista de vendedores de uma empresa.

```
1.  lista_vendedores = ["Lira", "João", "Amanda", "Alon", "Larissa"]
2.
3.  print("O Alon é um vendedor:")
4.  print("Alon" in lista_vendedores)
5.
6.  print("O Diego é um vendedor:")
7.  print("Diego" in lista_vendedores)
```

```
O Alon é um vendedor:
True
O Diego é um vendedor:
False
```

UNIVERSIDADE DE VASSOURAS

**Operadores**

**Condicionais (if, elif, else)**

**Laços de Repetição (for, while)**

UNIVERSIDADE DE VASSOURAS

# EXEMPLOS

## EXERCÍCIO 02

1.Leia a idade do usuário e classifique-o em:
- Criança – 0 a 12 anos
- Adolescente – 13 a 17 anos
- Adulto – acima de 18 anos
- Se o usuário digitar um número negativo, mostrar a mensagem que a idade é inválida

2. Calcular a média de um aluno que cursou a disciplina de Programação I, a partir da leitura das notas M1, M2 e M3; passando por um cálculo da média aritmética. Após a média calculada, devemos anunciar se o aluno foi aprovado, reprovado ou pegou exame
- Se a média estiver entre 0.0 e 4.0, o aluno está reprovado
- Se a média estiver entre 4.1 e 6.0, o aluno pegou exame
- Se a média for maior do que 6.0, o aluno está aprovado
- Se o aluno pegou exame, deve ser lida a nota do exame. Se a nota do exame for maior do que 6.0, está aprovado, senão; está reprovado

## EXERCÍCIO 03

1. Ler 5 notas e informar a média

2. Imprimir a tabuada do número 3 (3 x 1 = 1 - 3 x 10 = 30)