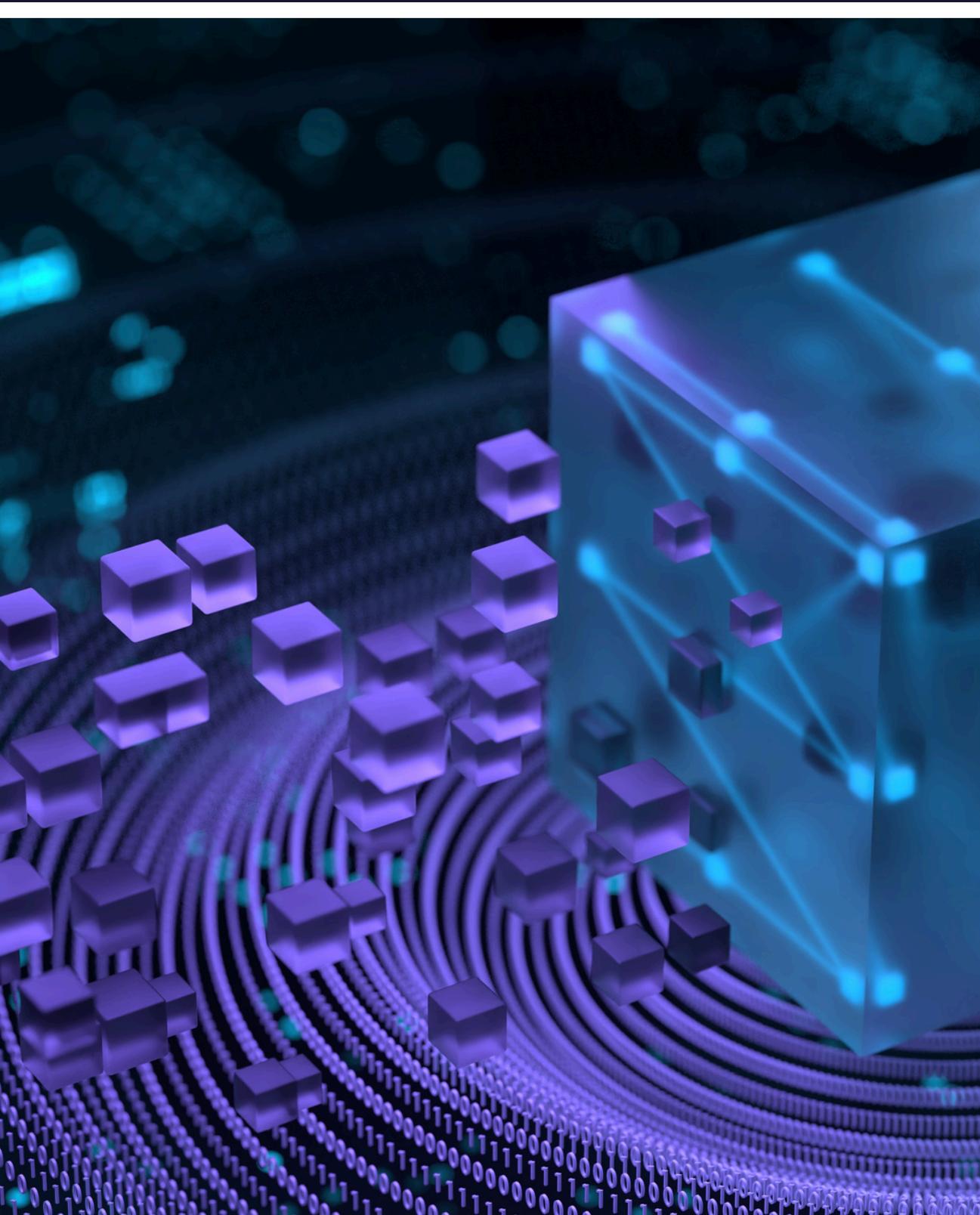
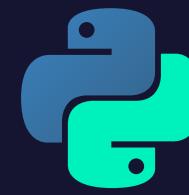


Python Course

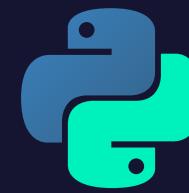
Minicurso ESTRUTURA DE DADOS EM PYTHON

VICTOR EMMANUEL OLMI LOUZADA



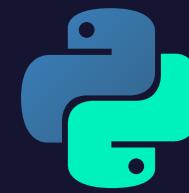
O que são estruturas de Dados?

Estruturas de Dados são formas organizadas de **armazenar, gerenciar e manipular** dados em um programa de computador. Elas definem a maneira como os dados são **coletados, acessados e organizados**, permitindo que algoritmos sejam **eficientes** em termos de **tempo e memória**.



Por que as Estruturas de Dados são Importantes?

- Facilitam a organização de informações.
- Melhoram a eficiência do código.
- São usadas em diversas áreas, como inteligência artificial, análise de dados e desenvolvimento de sistemas.
- Exemplo do dia a dia: Usar uma lista para armazenar os nomes de alunos de uma sala.



Tipos de Estruturas de Dados

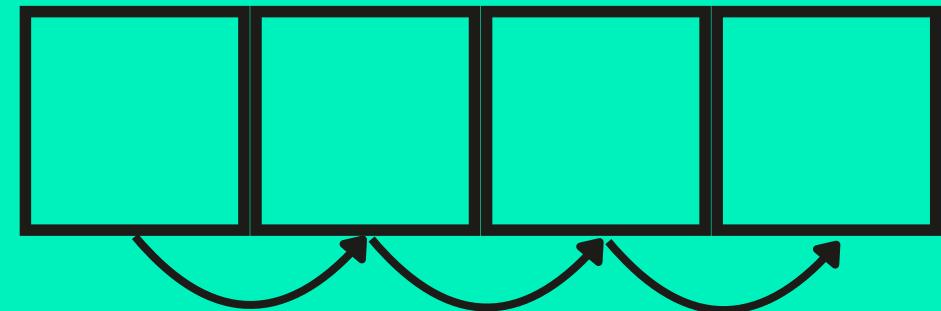
As estruturas de dados podem ser classificadas em dois grandes grupos:

Estruturas de Dados Lineares

Os elementos são organizados de forma sequencial.

- Lista: Sequência ordenada de elementos.
 - Exemplo: Lista de compras.
- Fila (Queue): Elementos seguem o princípio FIFO (First In, First Out).
 - Exemplo: Filas de atendimento em bancos.
- Pilha (Stack): Segue o princípio LIFO (Last In, First Out).
 - Exemplo: Histórico de navegação no navegador.

LISTA



PILHA





Tipos de Estruturas de Dados

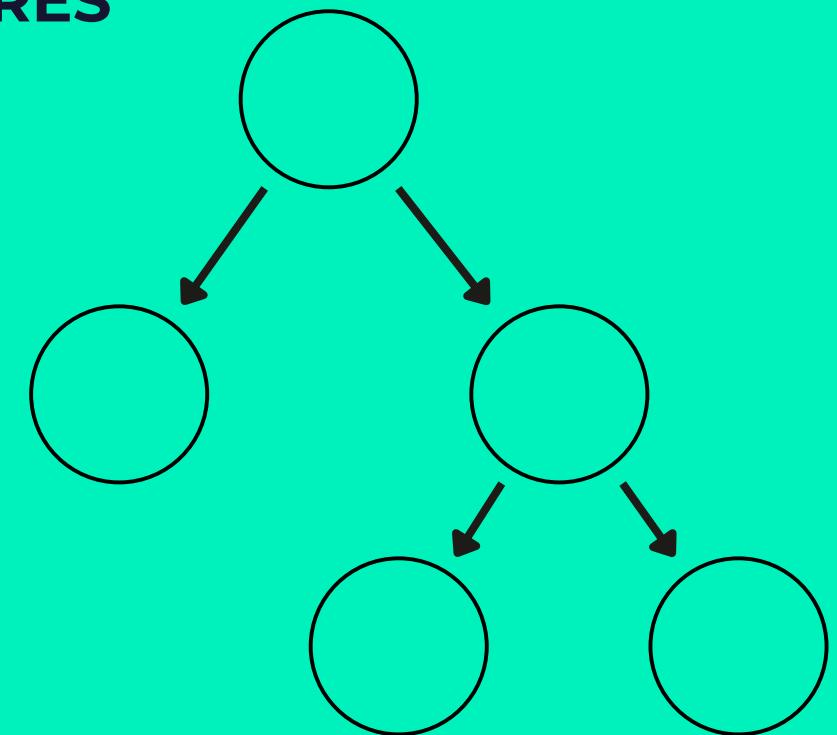
As estruturas de dados podem ser classificadas em dois grandes grupos:

Estruturas de Dados Não Lineares

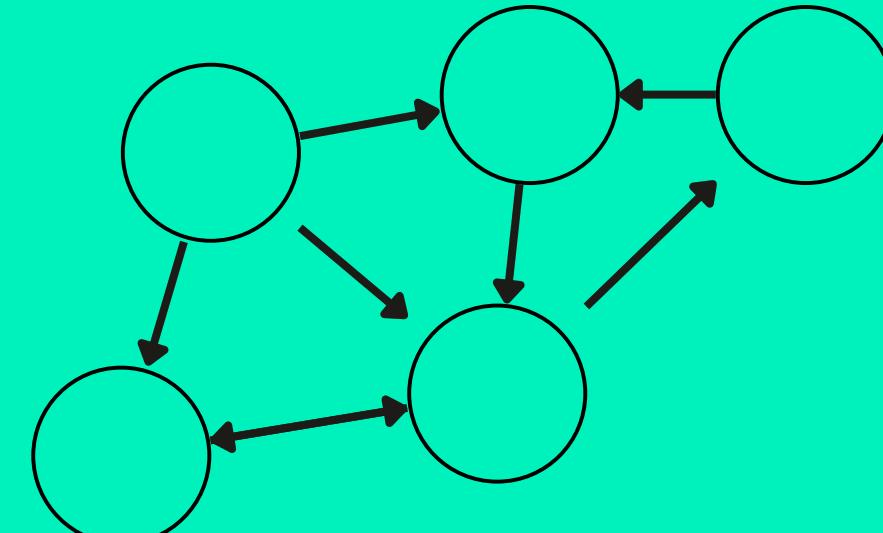
Os elementos não seguem uma sequência linear.

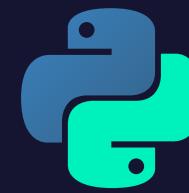
- Árvores: Estruturas hierárquicas com nós e ramificações.
 - Exemplo: Sistema de arquivos em um computador.
- Grafos: Conjunto de vértices conectados por arestas.
 - Exemplo: Rede de amigos em redes sociais.
 -

ÁRVORES



GRAFOS





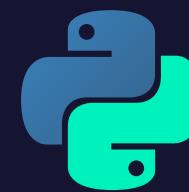
Listas

As listas são coleções ordenadas e mutáveis (podem ser alteradas após a criação). Elas permitem armazenar múltiplos itens, que podem ser de tipos diferentes.

Características:

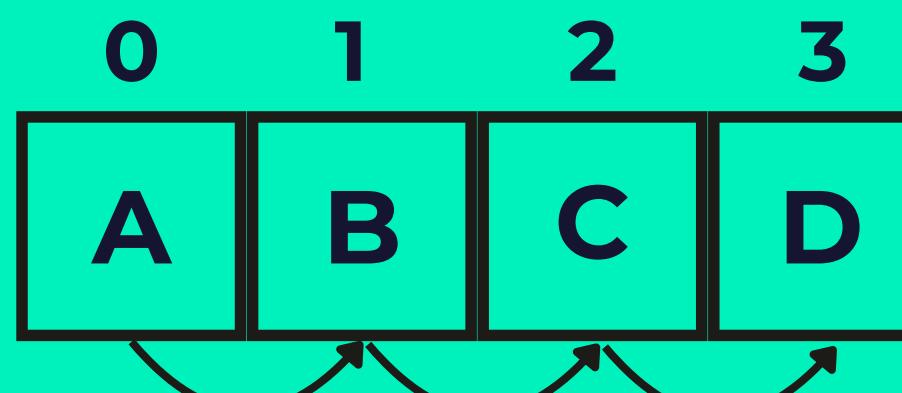
- Ordenadas: Mantêm a ordem dos elementos.
- Mutáveis: Você pode adicionar, remover ou alterar elementos.
- Aceitam duplicatas.

COLCHETES []

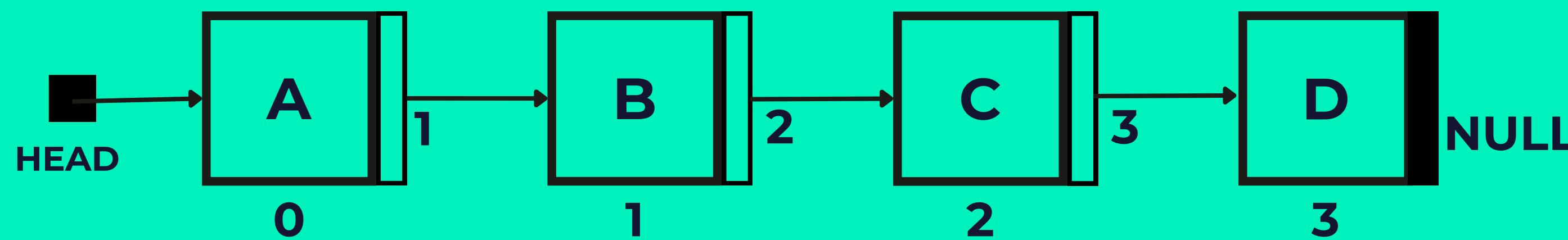


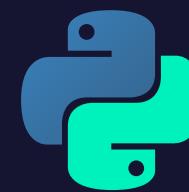
Exemplos de Listas

LISTA

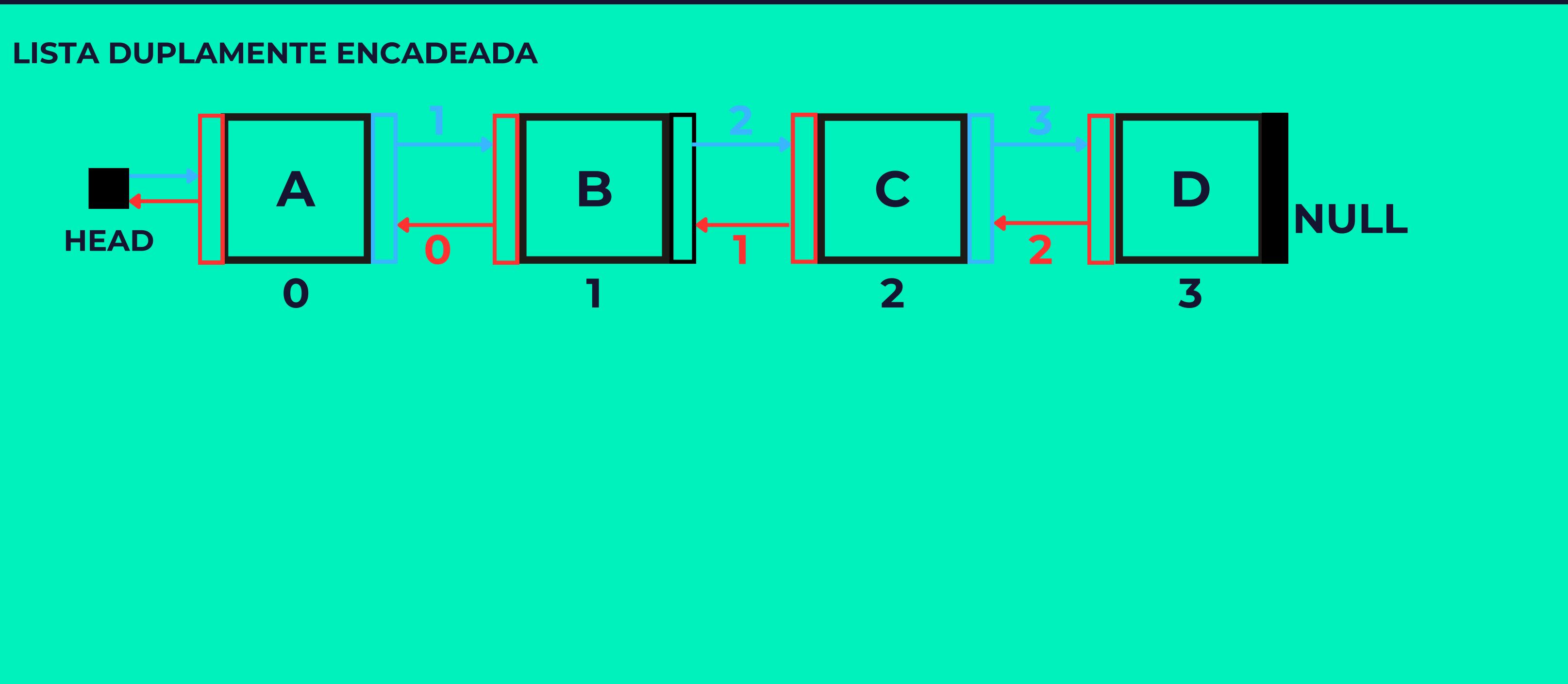


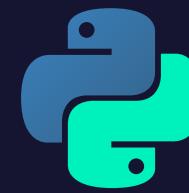
LISTA ENCADEADA





Exemplos de Listas





Exemplo prático de Lista

```
# Criando uma lista
frutas = ["maçã", "banana", "laranja"]

# Acessando elementos
print(frutas[0]) # Saída: maçã

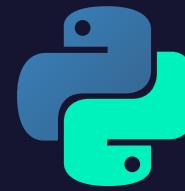
# Modificando um elemento
frutas[1] = "uva"
print(frutas) # Saída: ["maçã", "uva", "laranja"]

# Adicionando elementos
frutas.append("abacaxi")
print(frutas) # Saída: ["maçã", "uva", "laranja", "abacaxi"]

# Removendo elementos
frutas.remove("uva")
print(frutas) # Saída: ["maçã", "laranja", "abacaxi"]
```

MÉTODOS COMUNS:

- **append()**: Adiciona um elemento ao final da lista.
- **remove()**: Remove um elemento especificado.
- **pop()**: Remove e retorna o último elemento.
- **sort()**: Ordena a lista.
- **len()**: Retorna o tamanho da lista.

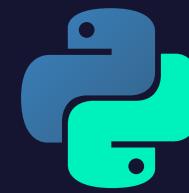


Tuplas

As tuplas são nativas dUma tupla é uma coleção ordenada e imutável. Ela é ideal para armazenar dados que não devem ser alterados.

o Python e representão uma coleção ordenada e imutável de elementos. Assim como as listas, as tuplas permitem armazenar diferentes tipos de dados, mas seu conteúdo não pode ser alterado após a criação.

PARÊNTESES ()



Exemplo prático de Tupla

```
# Criando uma tupla
cores = ("vermelho", "azul", "verde")

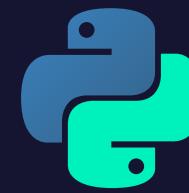
# Acessando elementos
print(cores[1]) # Saída: azul

tupla = (1, 2, 3, 2, 2)
print(tupla.count(2)) # 3 ocorrências

print(tupla.index(3)) # index 2
```

MÉTODOS COMUNS:

- **count()**: Conta quantas vezes um elemento aparece na tupla:
- **index()**: Retorna o índice da primeira ocorrência de um elemento:

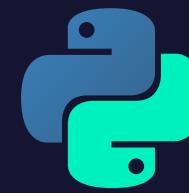


Exemplo prático de Tupla

RETORNO DE MÚLTIPLOS VALORES EM FUNÇÕES

```
def dividir(dividendo, divisor):
    quociente = dividendo // divisor
    resto = dividendo % divisor
    return quociente, resto # Retorna uma tupla

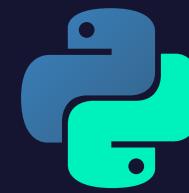
resultado = dividir(10, 3)
print(resultado)      # (3, 1)
print(resultado[0])   # 3 (quociente)
```



Exemplo prático de Tupla

AGRUPAR DADOS

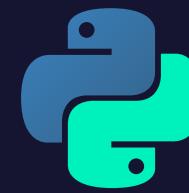
```
pessoa = ("João", 25, "Engenheiro")
nome, idade, profissao = pessoa # Desempacotamento
print(nome)      # João
print(idade)     # 25
print(profissao) # Engenheiro
```



Tuplas

Vantagens:

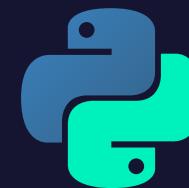
Mais rápidas do que listas.
Imutáveis, garantindo a integridade dos dados.



Conjuntos

Conjuntos são coleções desordenadas e sem elementos duplicados. São úteis para realizar operações como união, interseção e diferença.

CHAVES { }



Exemplo prático de Conjuntos

```
# Criando um conjunto
numeros = {1, 2, 3, 4}

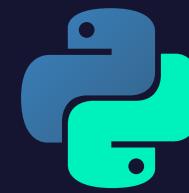
# Adicionando elementos
numeros.add(5)
print(numeros) # Saída: {1, 2, 3, 4, 5}

# Removendo elementos
numeros.remove(3)
print(numeros) # Saída: {1, 2, 4, 5}

# Operações com conjuntos
pares = {2, 4, 6}
print(numeros & pares) # Interseção: {2, 4}
```

MÉTODOS COMUNS:

- **add():** Adiciona um elemento ao conjunto.
- **remove():** Remove um elemento especificado (gera erro se não existir).
- **discard():** Remove um elemento especificado (não gera erro se não existir).
- **union():** Retorna a união de dois conjuntos.
- **difference():** Retorna a diferença entre dois conjuntos.
- **clear():** Remove todos os elementos do conjunto.

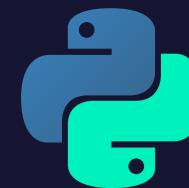


Dicionários

Dicionários armazenam pares de chave-valor.

São ideais para acessar dados de forma rápida
usando uma chave.

CHAVES { }



Exemplo prático de Dicionários

```
# Criando um dicionário
aluno = {"nome": "João", "idade": 25, "curso": "Engenharia"}

# Acessando valores
print(aluno["nome"]) # Saída: João

# Modificando valores
aluno["idade"] = 26
print(aluno) # Saída: {"nome": "João", "idade": 26, "curso": "Engenharia"}

# Adicionando novos pares
aluno["cidade"] = "São Paulo"
print(aluno) # Saída: {"nome": "João", "idade": 26, "curso": "Engenharia", "cidade": "São Paulo"}
```



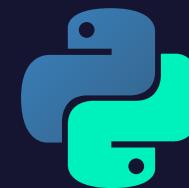
Exemplo prático de Dicionários

```
# Criando um dicionário
aluno = {"nome": "João", "idade": 25, "curso": "Engenharia"}

# Acessando valores
print(aluno["nome"]) # Saída: João

# Modificando valores
aluno["idade"] = 26
print(aluno) # Saída: {"nome": "João", "idade": 26, "curso": "Engenharia"}

# Adicionando novos pares
aluno["cidade"] = "São Paulo"
print(aluno) # Saída: {"nome": "João", "idade": 26, "curso": "Engenharia", "cidade": "São Paulo"}
```



Exemplo prático de Dicionários

MÉTODOS COMUNS:

- **keys()**: Retorna todas as chaves do dicionário.
- **values()**: Retorna todos os valores do dicionário.
- **items()**: Retorna pares de chave-valor como tuplas.
- **get()**: Retorna o valor de uma chave específica (ou um valor padrão, se a chave não existir).
- **update()**: Atualiza o dicionário com outro dicionário ou pares chave-valor.
- **pop()**: Remove e retorna o valor associado a uma chave.
- **clear()**: Remove todos os itens do dicionário.



Exercícios

EXERCÍCIO 1: LISTAS

Crie uma lista com os nomes de 5 cidades. Em seguida:

- Adicione uma nova cidade à lista.
- Remova a terceira cidade.
- Ordene a lista em ordem alfabética.

EXERCÍCIO 2: LISTAS

Crie uma lista de números de 1 a 10 e use um laço para calcular a soma de todos os números.

EXERCÍCIO 3: LISTAS

Verifique se o número 7 está presente na lista de números [3, 5, 7, 9, 11].

EXERCÍCIO 1: TUPLAS

Crie uma tupla com os números de 1 a 10. Depois:

- Acesse o quinto elemento.
- Tente modificar o terceiro elemento e observe o erro.

EXERCÍCIO 2: TUPLAS

Converta a tupla (1, 2, 3, 4) em uma lista e adicione o número 5.

EXERCÍCIO 3: TUPLAS

Crie uma tupla com 3 nomes e imprima cada um deles utilizando um laço for.



Exercícios

EXERCÍCIO 1: CONJUNTOS

Dado o conjunto {1, 2, 3, 4, 5}:

- Adicione o número 6.
- Remova o número 3.
- Crie outro conjunto {4, 5, 6, 7} e encontre a interseção.

EXERCÍCIO 2: CONJUNTOS

Verifique se o conjunto {1, 2} é subconjunto de {1, 2, 3, 4}.

EXERCÍCIO 3: CONJUNTOS

Remova todos os elementos de um conjunto utilizando o método adequado.

EXERCÍCIO 1: DICIONÁRIOS

Crie um dicionário para armazenar informações de um aluno (nome, idade, curso). Depois:

- Adicione uma nova informação: "nota final".
- Modifique o curso do aluno.
- Imprima todas as chaves e valores do dicionário.

EXERCÍCIO 2: DICIONÁRIOS

Verifique se a chave "endereço" está presente no dicionário {"nome": "Maria", "idade": 30}.

EXERCÍCIO 3: DICIONÁRIOS

Crie um dicionário com números e seus quadrados (ex.: {1: 1, 2: 4, 3: 9}) para os números de 1 a 5.