Introdução às Listas em Python

Objetivos

- Compreender o conceito de listas em Python.
- Aprender a criar, acessar e modificar elementos em listas.
- Explorar operações comuns em listas, como adição, remoção e ordenação.

1. Introdução

Explicação breve sobre a importância das listas em programação

As listas desempenham um papel fundamental na programação, oferecendo uma maneira eficiente e flexível de organizar e manipular conjuntos de dados. Elas são estruturas de dados versáteis que permitem armazenar uma coleção ordenada de elementos, podendo ser números, strings, objetos ou até mesmo outras listas. A importância das listas em programação reside em vários aspectos.

Armazenamento de Dados

Listas fornecem uma estrutura de armazenamento eficiente para dados relacionados. Elas permitem agrupar informações similares, tornando mais fácil e intuitivo trabalhar com conjuntos de dados complexos.

Acesso e Manipulação

A indexação em listas permite o acesso rápido a elementos individuais, facilitando a leitura e a modificação de dados. Além disso, as operações de adição, remoção e modificação de elementos são essenciais para a manipulação dinâmica de informações.

Flexibilidade

Listas são dinâmicas e podem ser facilmente modificadas durante a execução do programa. Essa flexibilidade é crucial para lidar com situações em que o tamanho ou a composição dos dados podem variar ao longo do tempo.

Iteração e Controle de Fluxo

A capacidade de iterar sobre os elementos de uma lista é uma ferramenta poderosa para controle de fluxo em programas. Ela permite a execução de operações repetitivas e simplifica a lógica do código.

Contextualização em Situações do Mundo Real

Imagine um sistema de gerenciamento de uma biblioteca, onde as informações sobre os livros podem ser organizadas em listas. Cada livro pode ser representado como um elemento na lista, contendo detalhes como **título**, **autor** e **número de cópias disponíveis**. Listas proporcionam uma maneira intuitiva e eficiente de armazenar e acessar esses dados.

Em aplicações mais amplas, como um sistema de comércio eletrônico, listas podem representar **carrinhos de compras**, **históricos de pedidos** ou **catálogos de produtos**. Essas estruturas de dados simplificam a manipulação e organização de informações, permitindo uma gestão eficiente dos dados em um contexto do mundo real.

Em resumo, as listas são uma ferramenta poderosa na programação, facilitando a manipulação de dados e proporcionando flexibilidade para lidar com uma variedade de situações do mundo real de forma eficaz e elegante.

2. O que são Listas

Definição: Uma lista é uma coleção ordenada de elementos mutáveis em Python.

Sintaxe básica: lista = [elemento1, elemento2, ..., elementoN].

3 - Criando Listas

Para criar listas no Python é necessário

- Utilizar o símbolo [] (colchetes) para as listas;
- Armazenar a lista em uma VARIÁVEL;
- Separa itens da lista pela vírgula;

```
# Exemplos

lista_vazia = []

numeros = [1, 2, 3, 4, 5]

nomes = ["Alice", "Bob", "Charlie"]

misturado = [1, "dois", 3.0, True]
```

4. Acessando Elementos

Em Python, a **indexação** é o processo de acessar elementos em uma sequência, como uma lista, tupla ou string, por meio de seus índices. É importante observar que a indexação em Python começa em 0, ou seja, o primeiro elemento de uma sequência possui índice 0, o segundo tem índice 1, e assim por diante.

Criando uma lista

frutas = ["Maçã", "Banana", "Morango", "Uva", "Pera"]

Índices: 0 1 2 3 4

Elementos: Maçã Banana Morango Uva Pera

Demonstração de como acessar elementos individuais e fatias.

 # Acessando elementos individualmente

primeira_fruta = frutas[0]

segunda_fruta = frutas[1]

print(primeira_fruta) # Saída: Maçã

print(segunda_fruta) # Saída: Banana

 # Acessando elementos individualmente

primeira_fruta = frutas[0]

segunda_fruta = frutas[1]

print(primeira_fruta) # Saída: Maçã

print(segunda_fruta) # Saída: Banana

Acessando Fatias (Slices)

Além de acessar elementos individualmente, podemos obter fatias (subconjuntos contínuos) da lista utilizando a notação de slice. A sintaxe básica é lista[início:fim], onde início é o índice inicial da fatia e fim é o índice final (não inclusivo).

 # Acessando uma fatia da lista

fatia = frutas[1:4] # Elementos do índice 1 ao 3 (não inclusivo)

print(fatia) # Saída: ['Banana', 'Morango', 'Uva']

5. Modificando Listas

Demonstração de como adicionar, modificar e remover elementos de uma lista.

 # Exemplos

numeros = [1, 2, 3, 4, 5]

numeros.append(6) # Adiciona 6 ao final da lista

numeros [1] = "novo_valor" # Modifica o valor no índice 1

del numeros [2] # Remove o elemento no índice 2

6. Operações Comuns

Apresentação de algumas operações comuns em listas, como ***len(), sum(), sort(), count().***

 # Exemplos

tamanho = len(lista)

soma = sum(numeros)

numeros.sort() **# Ordena a lista**

quantidade_dois = misturado.count("dois")

Exemplos Práticos de Listas em Python

1. Lista de Números Pares

 # Criando uma lista de números pares de 0 a 10

```
numeros_pares = [x for x in range(0, 11, 2)]
print(numeros_pares)
```

2. Adicionando Elementos Dinamicamente

 # Criando uma lista vazia e adicionando elementos dinamicamente

```
frutas = []
frutas.append("Maçã")
frutas.append("Banana")
frutas.append("Morango")
print(frutas)
```

3. Somando Elementos de uma Lista

```
# Somando todos os elementos de uma lista
valores = [10, 20, 30, 40, 50]
soma = sum(valores)
print(f"A soma dos valores é: {soma}")
```

4. Filtrando Elementos com List Comprehension

 # Criando uma nova lista com apenas os números ímpares de uma lista existente

```
numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
impares = [x for x in numeros if x % 2 != 0]
print(impares)
```

5. Removendo Duplicatas de uma Lista

 # Removendo duplicatas de uma lista

```
cores = ["vermelho", "azul", "verde", "vermelho", "amarelo", "azul"]

cores_sem_duplicatas = list(set(cores))

print(cores_sem_duplicatas)
```