# Apostila: Estruturas de Dados em Python

## Listas, Tuplas e Dicionários

# 1. Introdução

As estruturas de dados são ferramentas fundamentais em qualquer linguagem de programação, permitindo organizar e manipular conjuntos de valores. Em Python, as três principais estruturas de dados são as **listas**, **tuplas** e **dicionários**. Cada uma tem suas características e usos específicos.

# 2. Listas

## 2.1 O que são listas?

Uma **lista** é uma coleção de itens que podem ser modificados após a criação. Cada item em uma lista tem uma posição, também chamada de índice, que começa em 0.

```
minha_lista = [1, 2, 3, 4, 5]
```

## 2.2 Características principais

- **Mutáveis**: Podemos alterar os elementos de uma lista.
- **Ordenadas**: A posição dos elementos é mantida.
- **Aceitam tipos variados**: Uma lista pode conter diferentes tipos de dados.

```
lista_mista = [1, "Olá", 3.5, True]
```

## 2.3 Operações comuns

### 2.3.1 Acessar elementos

Podemos acessar elementos de uma lista usando seus índices.

```
primeiro_elemento = minha_lista[0]  # Acessa o primeiro item
```

### 2.3.2 Modificar elementos

Podemos alterar um valor em uma posição específica.

```
minha_lista[1] = 10  # Muda o segundo item para 10
```

### 2.3.3 Adicionar elementos

- `append()`: Adiciona um elemento ao final da lista.
- `insert()`: Adiciona um elemento em uma posição específica.

```
minha_lista.append(6)  # Adiciona 6 ao final
minha_lista.insert(2, 9)  # Adiciona 9 na posição 2
```

#### 2.3.4 Remover elementos

- `remove()`: Remove o primeiro item com o valor especificado.
- `pop()`: Remove um item pelo índice.

```
minha_lista.remove(9)  # Remove o valor 9
minha_lista.pop(0)  # Remove o primeiro item
```

### 2.4 Iteração sobre listas

Você pode usar um loop `for` para percorrer os elementos de uma lista.

```
for item in minha_lista:
    print(item)
```

## 3. Tuplas

### 3.1 O que são tuplas?

Uma **tupla** é similar a uma lista, mas é **imutável**, o que significa que, uma vez criada, não podemos modificar seus elementos. (OBS: Raramente usamos tuplas)

```
minha_tupla = (1, 2, 3, 4, 5)
```

### 3.2 Características principais

- **Imutáveis**: Não podemos alterar, adicionar ou remover elementos após a criação.
- **Ordenadas**: A posição dos elementos é mantida.
- **Aceitam tipos variados**: Assim como as listas, as tuplas podem conter diferentes tipos de dados.

```
tupla_mista = (1, "Olá", 3.5, True)
```

### 3.3 Operações comuns

#### 3.3.1 Acessar elementos

O acesso a elementos funciona da mesma forma que nas listas, usando índices.

```
primeiro_elemento = minha_tupla[0]
```

#### 3.3.2 Métodos das tuplas

Tuplas possuem menos métodos que listas devido à sua imutabilidade. No entanto, você pode usar métodos como `count()` e `index()`.

```
minha_tupla.count(2)  # Conta quantas vezes o valor 2 aparece
minha_tupla.index(3)  # Encontra a posição do valor 3
```

## 4. Dicionários

### 4.1 O que são dicionários?

Um **dicionário** é uma coleção de pares **chave-valor**. Ao contrário de listas e tuplas, que usam índices numéricos, os dicionários usam **chaves** para acessar seus valores.

```
meu_dicionario = {"nome": "João", "idade": 25, "cidade": "São Paulo"}
```

## 4.2 Características principais

- **Mutáveis**: Podemos alterar, adicionar ou remover pares chave-valor.
- **Desordenados**: Até o Python 3.7, os dicionários não mantinham a ordem dos elementos. A partir do Python 3.7, a ordem de inserção é mantida.
- **Chaves únicas**: Cada chave em um dicionário deve ser única.

## 4.3 Operações comuns

### 4.3.1 Acessar valores

Podemos acessar um valor usando sua chave.

```
nome = meu_dicionario["nome"]  # Acessa o valor associado à chave 'nome'
```

### 4.3.2 Modificar e adicionar pares chave-valor

Podemos alterar o valor de uma chave ou adicionar uma nova chave.

```
meu_dicionario["idade"] = 26  # Modifica o valor da chave 'idade'
meu_dicionario["profissão"] = "Engenheiro"  # Adiciona um novo par chave-valor
```

### 4.3.3 Remover pares chave-valor

- `pop()`: Remove e retorna o valor associado a uma chave.
- `del`: Remove um par chave-valor.

```
meu_dicionario.pop("cidade")  # Remove e retorna o valor associado à chave 'cidade'
del meu_dicionario["profissão"]  # Remove o par 'profissão'
```

## 4.4 Iteração sobre dicionários

Você pode percorrer as chaves, os valores ou ambos.

```
# Percorre as chaves
for chave in meu_dicionario:
    print(chave)

# Percorre os valores
for valor in meu_dicionario.values():
    print(valor)

# Percorre os pares chave-valor
for chave, valor in meu_dicionario.items():
    print(f"{chave}: {valor}")
```

# 5. Exercícios Práticos

Obs: Após a conclusão do exercício, criem um 'zip' com os códigos e eviem em meu email: `mauriciobenjamin700@gmail.com`.

Caso não saibam o que é `zip`, pesquisem usando os computadores do laboratório. Dúvidas entrem em contato comigo via email ou whatsapp (89 988025705)

### 5.1 Listas

1. Crie uma lista de 5 números e altere o valor do terceiro número.
2. Adicione um novo número no final da lista e remova o segundo número.

### 5.2 Tuplas

1. Crie uma tupla com 4 elementos. Tente alterar o valor do segundo elemento e observe o erro.
2. Use o método `count()` para contar quantas vezes um valor aparece na tupla.

### 5.3 Dicionários

1. Crie um dicionário que armazene o nome, idade e cidade de uma pessoa. Adicione uma nova chave chamada "profissão".

2. Remova a chave "cidade" do dicionário e mostre o dicionário atualizado.

## Aviso

A prova será realizada no dia 23-10-2024 ou no dia 30-10-2024, então se preparem!