✨ Dicionários em ```Python ✨

> O que são, Como Usar e Manipular Dicionários

Nesta aula, vamos aprender sobre dicionários em Python. Dicionários são uma estrutura de dados que permitem armazenar pares de chave-valor. Eles são ideais para representar informações que têm uma relação, como o nome de uma pessoa e sua idade, ou um produto e seu preço.

💡 O que é um Dicionário? 💡

Um dicionário é uma coleção não ordenada, mutável e indexada de elementos. Cada item em um dicionário possui uma chave e um valor. Eles são definidos usando chaves `{}`.

```
aluno = {
    "nome": "Ana",
    "idade": 16,
    "nota": 9.5
}
```

Aqui, aluno é um dicionário com três pares chave-valor:

- `"nome"` é a chave e `"Ana"` é o valor.
- `"idade"` é a chave e `16` é o valor.
- `"nota"` é a chave e `9.5` é o valor.

⚡ Acessando Valores em um Dicionário ⚡

Podemos acessar os valores dos dicionários usando suas chaves entre colchetes `[]`.

```
print(aluno["nome"])  # Exibe "Ana"
```

Também podemos usar o método `.get()`, que permite definir um valor padrão caso a chave não exista.

```
print(aluno.get("idade"))  # Exibe 16
print(aluno.get("turma", "Não especificado"))  # Exibe "Não especificado"
```

O método `.get()` é útil para evitar erros quando a chave não está presente.

⚡ Adicionando e Atualizando Valores ⚡

Dicionários são mutáveis, o que significa que podemos adicionar ou atualizar elementos.

Adicionando um novo par chave-valor:

```
aluno["turma"] = "3A"
print(aluno) #  A nova chave "turma" é adicionada
```

Atualizando um valor existente:

```
aluno["idade"] = 17
print(aluno)  # A idade agora é 17
```

⚡ Removendo Itens de um Dicionário ⚡

Podemos remover itens usando o método `pop()`, que remove o item da chave especificada e retorna o valor.

```
nota = aluno.pop("nota")
print(nota)  # Exibe 9.5
print(aluno)  # Exibe {'nome': 'Ana', 'idade': 17, 'turma': '3A'}
```

Também podemos usar a palavra-chave `del` para remover um par chave-valor.

```
del aluno["turma"]
print(aluno)  # Exibe {'nome': 'Ana', 'idade': 17}
```

⚡ Iterando Sobre um Dicionário ⚡

Podemos usar loops para iterar sobre as chaves ou os valores de um dicionário.

Iterar Sobre Chaves:

```
for chave in aluno:
    print(chave) # Exibirá todas as chaves
```

Iterar Sobre Valores:

```
for valor in aluno.values():
    print(valor) # Exibirá todos os valores
```

Iterar Sobre Chaves e Valores:

Podemos usar o método `.items()` para obter tanto as chaves quanto os valores.

```
for chave, valor in aluno.items():
    print(f"{chave}: {valor}")
```

Isso exibirá:

- **nome:** Ana
- **idade:** 17

⚡ Métodos Úteis de Dicionários ⚡

Python oferece muitos métodos úteis para trabalhar com dicionários:

- `keys()`: Retorna todas as chaves do dicionário.

```
chaves = aluno.keys()
print(chaves)  # Exibe dict_keys(['nome', 'idade'])
```

- `values()`: Retorna todos os valores do dicionário.

```
valores = aluno.values()
print(valores)  # Exibe dict_values(['Ana', 17])
```

- `items()`: Retorna todos os pares chave-valor.

```
itens = aluno.items()
print(itens)  # Exibe dict_items([('nome', 'Ana'), ('idade', 17)])
```

- `clear()`: Remove todos os itens do dicionário.

```
aluno.clear()
print(aluno)  # Exibe {}
```

⚡ Dicionários Aninhados ⚡

Podemos ter dicionários dentro de outros dicionários. Isso é útil para representar informações mais complexas.

```
escola = {
    "aluno1": {"nome": "Ana", "idade": 16},
    "aluno2": {"nome": "João", "idade": 17}
}
```

O dicionário `escola` contém dois dicionários: `aluno1` e `aluno2`, cada um com seus próprios pares chave-valor.

Para acessar o `nome` do `aluno1`:

```
print(escola["aluno1"]["nome"])  # Exibe "Ana"
```

📋 Resumo da Aula 📋

- **Dicionário:** Estrutura de dados composta por pares chave-valor, definida usando `{}`.
- **Acessar valores:** Use colchetes `[]` ou o método `.get()`.
- **Adicionar/Atualizar:** Use `dicionario[chave] = valor`.
- **Remover itens**: Use `pop()` ou `del`.
- **Iterar:** Use `for` para percorrer chaves, valores ou ambos.
- **Dicionários aninhados:** Permitem representar estruturas mais complexas.