PYTHON ACADEMY

FATIAMENTO (OU SLICING) DE LISTAS NO PYTHON

Manipular listas é um conhecimento extremamente importante para o Pythonista e nesse ebook você via aprender sobre o Slicing ou Fatiamento de Listas

PYTHONACADEMY.COM.BR

Salve salve Pythonista 👋

Neste artigo, vamos explorar o fatiamento de listas (slicing) em Python

O fatiamento é uma técnica muito útil para selecionar subconjuntos de elementos em uma lista de forma eficiente.

Vamos entender como funciona o fatiamento em Python, como utilizá-lo e qual a sua importância na programação.

O que é o Fatiamento de Listas?

O fatiamento de listas é uma operação que nos permite extrair uma porção específica de uma lista, criando uma nova lista com os elementos selecionados.

Essa operação pode ser útil em várias situações, como por exemplo quando precisamos processar apenas uma parte dos elementos de uma lista.

Em Python, o fatiamento é realizado utilizando a notação de colchetes `[]`, juntamente com índices que especificam o início e o fim do intervalo que queremos fatiar.

Por exemplo, considere a seguinte lista:

```
numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Para extrair uma parte dessa lista, podemos utilizar a sintaxe `numeros[inicio:fim]`.

Onde `inicio` é o índice do primeiro elemento que queremos incluir no fatiamento e `fim` é o índice do último elemento + 1.

Fatiamento Básico

Vamos começar entendendo o fatiamento básico de listas.

Considere a lista `numeros` definida acima. Se quisermos fatiar essa lista e obter apenas os elementos com índice 2, 3 e 4, podemos fazer da seguinte forma:

```
numeros_fatiados = numeros[2:5]
print(numeros_fatiados)
```

A saída será:

```
[3, 4, 5]
```

O índice 2 corresponde ao número 3, o índice 3 corresponde ao número 4 e o índice 4 corresponde ao número 5.

Note que o índice 5 é o “fim” e não é incluído no fatiamento.

Fatiamento com Índices Negativos

Uma característica interessante do fatiamento de listas em Python é a possibilidade de utilizar índices negativos.

Quando utilizamos índices negativos, contamos a partir do final da lista.

O último elemento tem índice -1, o penúltimo -2, e assim por diante.

Vejamos um exemplo para ilustrar essa ideia. Suponha que temos uma lista com os dias da semana e queremos fatiar essa lista de forma a obter apenas os últimos três dias:

```
dias_semana = ["segunda", "terça", "quarta", "quinta", "sexta", "sábado", "domingo"]
ultimos_tres_dias = dias_semana[-3:]
print(ultimos_tres_dias)
```

A saída será:

```
["sábado", "domingo"]
```

O índice -3 corresponde ao “sábado” e o índice -1 corresponde ao “domingo”.

Note que nesse caso não precisamos especificar o “fim”, pois queremos todos os elementos até o final da lista.

> 💡 *Estou desenvolvendo o **DevBook**, uma plataforma que usa IA para gerar ebooks técnicos profissionais. Depois de ler, dá uma passada no site!*

Fatiamento com Passo (*step*)

Além do início e fim, podemos utilizar um terceiro valor para especificar o passo do fatiamento.

O passo indica de quantos elementos em quantos elementos queremos selecionar da lista. Se não especificado, o passo é considerado como 1.

Para entender melhor essa ideia, vamos a um exemplo. Considere a lista:

```
pares = [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

Se quisermos fatiar essa lista de forma a obter apenas os números pares em posições ímpares, podemos fazer da seguinte forma:

```
pares_fatiados = pares[1::2]
print(pares_fatiados)
```

A saída será:

```
[2, 6, 10, 14, 18]
```

Nesse caso, utilizamos o `1` como valor do início para pular o primeiro elemento.

O `2` como passo, para selecionar apenas os elementos de posição ímpar e o omitimos o fim, para incluir todos os elementos até o final da lista.

Fatiamento de Strings

Além de ser aplicado em listas, o fatiamento também pode ser utilizado em strings.

O comportamento é semelhante ao das listas, onde cada caractere é tratado como um elemento.

Vejamos um exemplo para ilustrar essa ideia. Considere a seguinte string:

```
mensagem = "Olá, Mundo!"
```

Se quisermos fatiar essa string de forma a obter apenas a palavra “Mundo”, podemos fazer da seguinte forma:

```
mensagem_fatiada = mensagem[5:10]
print(mensagem_fatiada)
```

A saída será:

```
"Mundo"
```

É importante ressaltar que o fatiamento produz uma nova lista ou string, contendo uma cópia dos elementos selecionados da lista original.

Modificar a lista resultante do fatiamento não afetará a lista original.

Conclusão

O fatiamento de listas (slicing) é uma técnica poderosa em Python, que nos permite extrair subconjuntos de elementos de forma rápida e eficiente.

Neste artigo, vimos como utilizar o fatiamento básico, utilizando índices positivos e negativos, e também como utilizar o passo para selecionar elementos em intervalos específicos.

Além disso, vimos que o fatiamento também pode ser aplicado em strings.

Com essa técnica, podemos manipular facilmente listas e strings em Python, selecionando apenas os elementos necessários para nossas operações.

Portanto, seu conhecimento é essencial para qualquer programador Python que trabalhe com manipulação de dados em listas ou strings.

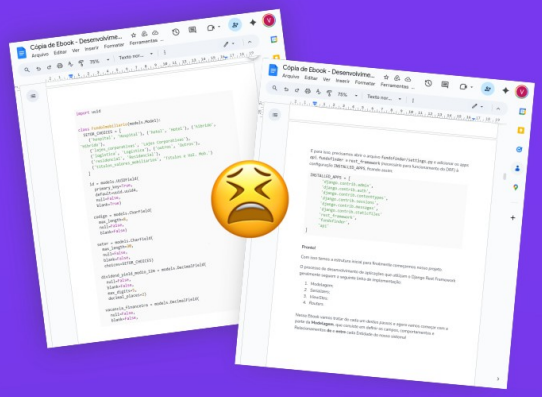
Nos vemos na próxima, Pythonista 👋

Não se esqueça de conferir!

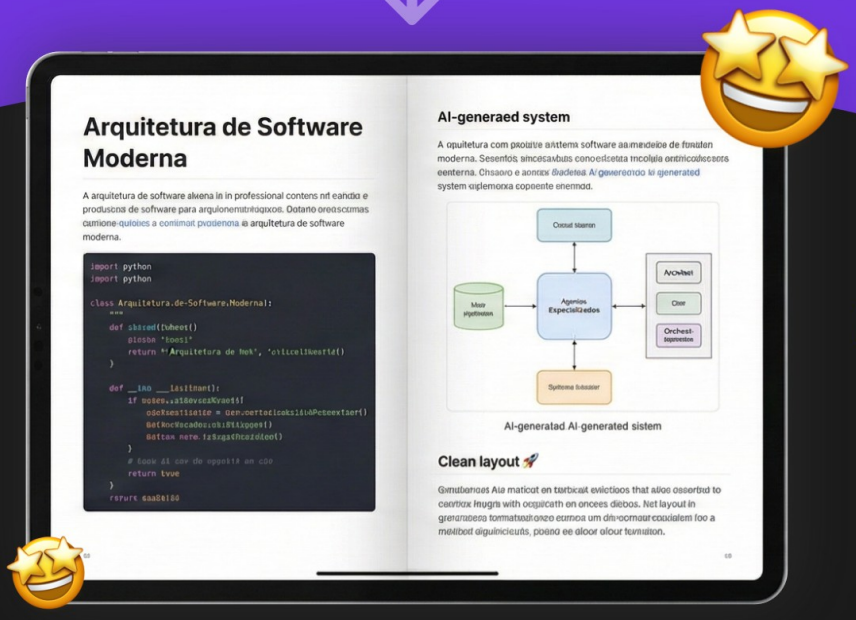
DevBook

Crie Ebooks técnicos em minutos com IA

Conheça a 1ª IA Especializada na criação de Ebooks **com código**!



Chega de formatar código no Google Docs



Syntax Highlight

Infográficos feitos por IA

Adicione Banners Promocionais

Deixe que nossa IA faça o trabalho pesado

Edite em Markdown em Tempo Real

TESTE AGORA

PRIMEIRO CAPÍTULO 100% GRÁTIS