



PYTHON
ACADEMY



O PACOTE COLLECTIONS DO PYTHON

Nesse ebook você vai aprender a utilizar as estruturas de dados presentes no famoso pacote collections do Python

PYTHONACADEMY.COM.BR

Gere ebooks como este com



em <https://ebookr.ai>

Crie ebooks profissionais incríveis em minutos com IA



Chega de formatar texto no Google Docs, Word ou ferramentas que só te fazem perder tempo...

E deixe que nossa IA faça o trabalho pesado!



Capas gerados por IA



Infográficos feitos por IA



Edite em Markdown em Tempo Real



Adicione Banners Promocionais

TESTE AGORA

PRIMEIRO CAPÍTULO 100% GRÁTIS

Salve salve Pythonista!

As estruturas de dados são elementos fundamentais na programação e desempenham um papel crucial na eficiência e organização do código.

A linguagem de programação Python possui uma biblioteca chamada `Collections`, que traz um conjunto de estruturas de dados adicionais além das já presentes na biblioteca padrão.

Neste artigo, abordaremos as estruturas de dados mais utilizadas do pacote `Collections` do Python e como elas podem auxiliar no desenvolvimento de aplicações mais eficientes e flexíveis.

Antes de explorarmos as estruturas de dados disponíveis no pacote `Collections`, é importante destacar que essa biblioteca oferece alternativas otimizadas para as estruturas padrão do Python, como listas e dicionários.

Além disso, as estruturas do `Collections` também são altamente eficientes na manipulação e armazenamento de dados.

A Collection `defaultdict`

Uma das estruturas de dados mais utilizadas do pacote `Collections` é o `defaultdict`.

Diferente do dicionário tradicional, o `defaultdict` aceita uma função que será chamada quando um valor não existir.

Por exemplo, se tentarmos acessar um elemento que não está presente no dicionário, em vez de retornar um `KeyError`, o `defaultdict` irá criar o elemento utilizando a função especificada.

```
from collections import defaultdict

dicionario = defaultdict(int)
dicionario['chave'] += 1
```

Nesse exemplo, o dicionário `dicionario` utiliza a função `int` como padrão para os valores que não foram inicializados previamente.

Caso a chave não exista, o valor será inicializado como 0.

Dessa forma, podemos incrementar o valor de `dicionario['chave']` mesmo sem ter atribuído um valor inicial a ele.

A Collection Counter

Outra estrutura de dados interessante é o `Counter`, que permite contar elementos em uma lista ou qualquer objeto iterável.

Esse contador é implementado como um dicionário de contadores, onde cada elemento é mapeado para uma contagem.

```
from collections import Counter

lista = ['a', 'b', 'c', 'a', 'b', 'a']
contador = Counter(lista)
print(contador)
```

A saída será:

```
Counter({'a': 3, 'b': 2, 'c': 1})
```

O `Counter` também oferece métodos úteis, como a possibilidade de encontrar os elementos mais comuns, remover ou adicionar elementos.

A Collection `deque`

Um tipo de estrutura de dados frequente na programação é a fila, que segue o princípio **FIFO** (*First In, First Out*).

O pacote `Collections` oferece a classe `deque`, que permite criar uma fila com todas as operações de inserção e remoção eficientes.

```
from collections import deque

fila = deque()
fila.appendleft(1)
fila.appendleft(2)
fila.appendleft(3)
print(fila)
```

A saída será:

```
deque([3, 2, 1])
```

Perceba que utilizamos o método `appendleft` para inserir os elementos na fila, simulando o comportamento FIFO.

Essa função é mais eficiente em uma fila do que o método `append`, que insere o elemento no final.

A Collection namedtuple

Outra estrutura de dados muito útil é o `namedtuple`, um subtipo de tupla que atribui nomes fixos aos elementos.

Com o `namedtuple`, podemos criar uma classe de tupla personalizada sem a necessidade de criar uma classe completa.

```
from collections import namedtuple

Pessoa = namedtuple('Pessoa', ['nome', 'idade'])
p1 = Pessoa('João', 25)
print(p1.nome) # João
print(p1.idade) # 25
```

Aqui, criamos um `namedtuple` chamado `Pessoa` com os campos `nome` e `idade`.

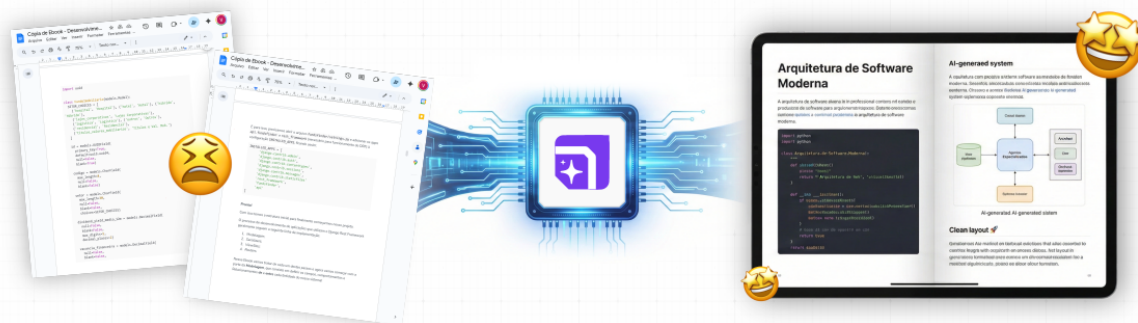
Em seguida, instanciamos esse tipo de dado e atribuímos valores aos seus campos.

Por fim, podemos acessar esses campos como atributos da instância.



Criei o Ebookr.ai, uma plataforma que usa IA para gerar ebooks profissionais sobre qualquer tema — com capa gerada por IA, infográficos automáticos e exportação em PDF. Confere!


Crie Ebooks profissionais incríveis em minutos com IA



Chega de formatar texto no Google Docs, Word ou ferramentas que só te fazem perder tempo...


... e deixe que nossa IA faça o trabalho pesado!

TESTE AGORA! PRIMEIRO CAPÍTULO 100% GRÁTIS [↗](#)

 Capas gerados por IA

 Adicione Banners Promocionais

 Edite em Markdown em Tempo Real

 Infográficos feitos por IA

A Collection **OrderedDict**

Por fim, temos o **OrderedDict**, uma variante do dicionário padrão que mantém a ordem dos elementos de acordo com a inserção.

Enquanto o dicionário padrão do Python não garante uma ordem específica, o **OrderedDict** lembra a ordem de inserção dos elementos.

```
from collections import OrderedDict

dicionario = OrderedDict()
dicionario['chave1'] = 1
dicionario['chave2'] = 2
print(dicionario)
```

A saída será:

```
OrderedDict([('chave1', 1), ('chave2', 2)])
```

A ordem dos elementos no `OrderedDict` é preservada de acordo com a ordem de inserção.

Isso é útil em diversas situações em que é necessário iterar sobre os itens do dicionário de acordo com a ordem de inserção.

Conclusão

O pacote `Collections` do Python oferece uma gama de estruturas de dados adicionais às já presentes na biblioteca padrão, trazendo alternativas otimizadas para listas, dicionários e outros tipos de dados.

Neste artigo, apresentamos algumas das estruturas de dados mais utilizadas, como o `defaultdict`, `Counter`, `deque`, `namedtuple` e `OrderedDict`.

Ao utilizar essas estruturas de dados, é possível melhorar a eficiência do código e deixar o código mais legível e organizado. Com o `defaultdict`, podemos evitar o uso de condicionais para verificar a existência de chaves em dicionários.

O `Counter` facilita a contagem de elementos em listas e objetos iteráveis.

O `deque` oferece uma implementação otimizada de uma fila.

O `namedtuple` permite a criação de classes de tuplas personalizadas de forma mais simples.

E o `OrderedDict` mantém a ordem dos elementos inseridos em um dicionário, facilitando a iteração sobre eles.

Portanto, conhecer e utilizar as estruturas de dados disponíveis no pacote `Collections` do Python é fundamental para programadores que desejam desenvolver aplicações mais eficientes e organizadas.

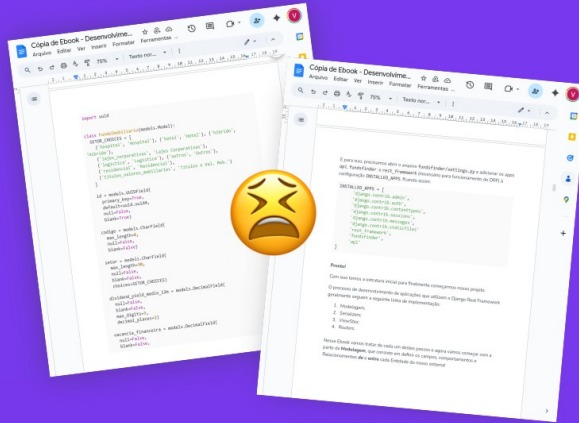
Experimente essas estruturas de dados e aproveite ao máximo seus recursos.

Não se esqueça de conferir!



Ebookr

Crie Ebooks profissionais em minutos com IA



Chega de formatar código no Google Docs ou Word



Capas gerados por IA



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