Iteradores e Geradores

Guilherme Arthur de Carvalho

Analista de sistemas

https://linktr.ee/decarvalhogui

Objetivo Geral

Vamos aprender sobre iteradores e geradores em Python. Esses são conceitos poderosos que nos permitem trabalhar com sequências de maneira eficiente.

Pré-requisitos

- ✓ Python e VSCode

Conteúdo

- ❑ Iteradores
- ❑ Geradores

Iteradores

Iteradores e Geradores

Introdução

Em Python, um iterador é um objeto que contém um número contável de valores que podem ser iterados, o que significa que você pode percorrer todos os valores. O protocolo do iterador é uma maneira do Python fazer a iteração de um objeto, que consiste em dois métodos especiais "__iter__()" e "__next__()"

Ler arquivos grandes

- Economizar memória evitando carregar todas as linhas do arquivo.
- Iterar linha a linha do arquivo.

Exemplo de código:

```python
class FileIterator:
    def __init__(self, filename):
        self.file = open(filename)

    def __iter__(self):
        return self

    def __next__(self):
        line = self.file.readline()
        if line ≠ '':
            return line
        else:
            self.file.close()
            raise StopIteration

# Uso do FileIterator
for line in FileIterator('large_file.txt'):
    print(line)

```

Geradores

Iteradores e Geradores

O que são geradores?

São tipos especiais de iteradores, ao contrário das listas ou outros iteráveis, não armazenam todos os seus valores na memória.

São definidos usando funções regulares, mas, ao invés de retornar valores usando "return", utilizam "yield".

Características de geradores

- Uma vez que um item gerado é consumido, ele é esquecido e não pode ser acessado novamente.
- O estado interno de um gerador é mantido entre chamadas.
- A execução de um gerador é pausada na declaração "yield" e retomada daí na próxima vez que ele for chamado.

Recuperando dados de uma API

- Solicitar dados por páginas (paginação).
- Fornecer um produto por vez entre as chamadas.
- Quando todos os produtos de uma página forem retornados, verificar se existem novas páginas.
- Tratar o consumo da API como uma lista Python.

Exemplo de código:

```python
import requests

def fetch_products(api_url, max_pages=100):
    page = 1
    while page ≤ max_pages:
        response = requests.get(f"{api_url}?page={page}")
        data = response.json()
        for product in data['products']:
            yield product
        if 'next_page' not in data:
            break
        page += 1

# Uso do gerador
for product in fetch_products("http://api.example.com/products"):
    print(product['name'])
```

Dúvidas?

> Fórum/Artigos - https://web.dio.me/articles

Exemplo de código:

Insira sua imagem dentro deste espaço
(retire o retângulo azul, ele deverá ser utilizado somente para referência)