Atividade Guiada
Engenharia de Software

Objetivos de Aprendizagem

Compreender o propósito e a aplicação dos padrões Adapter e Iterator.
Implementar ambos os padrões e aplicar boas práticas de Engenharia de Software: documentação, tratamento de erros e testes unitários.

Contexto do Problema

Uma equipe de Ciência de Dados precisa unificar o acesso a dados provenientes de diferentes fontes para análise posterior. Atualmente, há três origens distintas:
1. Um arquivo CSV local (dados.csv) contendo registros tabulares.
2. Uma API externa que retorna dados em formato JSON.
3. Um objeto interno de uma classe Python que armazena dados manualmente.

Essas fontes são incompatíveis entre si, cada uma tem sua própria forma de fornecer dados. Sua tarefa é criar uma interface unificada que permita iterar sobre os registros sem que o código principal precise saber de onde vêm os dados.

Etapas da Atividade

Parte 1: Compreensão e Planejamento

Leia a descrição, pesquise sobre os padrões e reflita:
- O que os padrões Adapter e Iterator resolvem?
- Onde eles se aplicam neste contexto?

Parte 2: Implementação do Iterator

Crie uma classe que implemente um Iterator genérico para percorrer registros.
Essa classe deve:
- Ter os métodos __iter__() e __next__() corretamente definidos;
- Ser capaz de iterar sobre uma coleção de dados (lista, dicionário, etc.);
- Tratar exceções como StopIteration de forma adequada.

Parte 3 — Implementação do Adapter

Implemente três adaptadores, um para cada tipo de fonte:
1. CSVAdapter: lê e converte dados do CSV.
2. APIAdapter: conecta-se a uma API simulada (pode ser um dicionário fixo).
3. ObjectAdapter: acessa dados de um objeto do domínio da aplicação.

Cada adaptador deve implementar uma interface comum (por exemplo, um método get_data() que retorna uma lista de registros).

O código principal deve ser capaz de consumir qualquer uma das fontes sem precisar mudar sua lógica.

Parte 4: Integração, Testes e Documentação

Implemente testes unitários para cada adaptador e para o iterator.
Trate erros previsíveis.
Adicione docstrings a todas as classes e métodos, descrevendo propósito, parâmetros e retorno.
Crie um módulo main.py que:
- Instancia um adaptador;
- Obtém os dados via get_data();
- Usa o Iterator para percorrê-los e exibir resultados.

Parte 5: Reflexão

Responda brevemente através de comentários no código:
- Quais seriam as consequências de não usar o Adapter?
- Como o Iterator melhora a legibilidade e manutenção do código?

Entrega Esperada

```
/atividade_adapter_iterator/
├── adapters.py       # Contém as classes CSVAdapter, APIAdapter, ObjectAdapter
├── iterator.py       # Classe DataIterator
├── main.py           # Código principal (driver code)
├── tests/            # Módulo de testes unitários
│   ├── test_adapters.py
│   └── test_iterator.py
└── README.md      # Instruções de execução e reflexão
```