Projeto Classificatório – Rocky

Autor: Pedro Tortello

Data: 21/08/2020

Foi escolhida a linguagem Python por se tratar de uma linguagem de sintaxe simples e de fácil compreensão. Outro fator foi a familiaridade que possuo com a linguagem.

Procurei utilizar boas práticas como o conceito de DRY (Don't Repeat Yourself) e o padrão de estilo PEP-8 (https://www.python.org/dev/peps/pep-0008/). Além disso utilizei a língua inglesa para todo o código, com exceção dos comentários.

Utilizei como referência principal a documentação do Python 3.8 (https://docs.python.org/3/).

A solução escrita no arquivo **resolucao.py** tem por objetivo corrigir os dados de entrada fornecidos pelo arquivo **broken-database.json**, validar esses dados e gerar o arquivo **saida.json** com os dados corrigidos. Além disso ele imprime na tela os testes de validação conforme solicitado no problema.

O arquivo **resolucao.py** contém comentários indicando as principais funções para cada trecho do código.

Os dados do arquivo **broken-database.json** são carregados para a memória e corrigidos no loop principal indicado pelo comentário "# Recuperando os dados corrompidos".

Foi criada a função *name_fixer* com a intenção de trazer uma abstração ao loop principal de recuperação de dados, facilitando seu entendimento. A função verifica cada caractere do argumento *name* contra o dicionário *modified*, buscando e substituindo os caracteres corrompidos. Esta função retorna uma string já corrigida.

Na saída dos dados foi necessário tratar o código para evitar bugs. Esse tratamento consiste no argumento *ensure_ascii=False* da função *json.dump*. Esse argumento garante que a acentuação de palavras como "Fogão" ou "Acessórios" sejam preservadas.

Para o primeiro teste de validação, a ordenação dos produtos por categoria e por id foi feita com a função *sorted*, built-in do Python, e a função *itemgetter* da biblioteca *operator*, também built-in. A impressão em tela foi formatada através de um template para facilitar a visualização.

No segundo teste de validação, através de *try* os valores são somados em sua respectiva categoria no dicionário *totals*. Caso a categoria ainda não exista no dicionário, ela será adicionada através do *except*.

Os trechos de código não comentados neste documento foram considerados mais simples e/ou autoexplicativos.