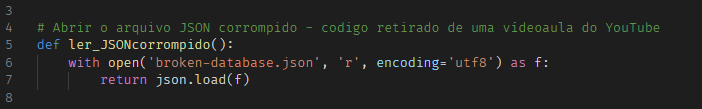
**Candidato:** Edgar Augusto Steffen – **Projeto Classificatório Processo seletivo – TI**

A linguagem de programação escolhida para a realização desse projeto foi a Python 3.8.5, o motivo da escolha foi um pouco pessoal e por insistência de um amigo que dizia que Python é uma linguagem fácil de se aprender e que um código escrito em 15 linhas no C poderia ser escrito em 3 no Python. Por curiosidade em aprender como funciona que resolvi usar essa linguagem.

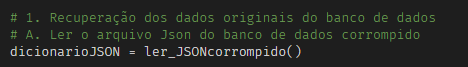
Grande parte do que foi usado para realização desse teste, eu vi exemplos no site W3Schools e antes de colocar cada função no código, eu fazia um teste em um arquivo separado. O divertido foi aprender Python em 4 dias para realização disso.

**Referente as funcionalidades do código:**

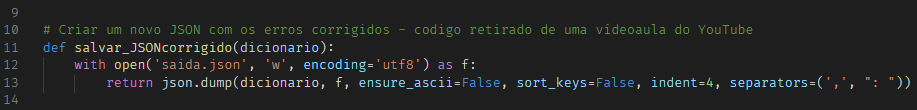
As três primeiras funções, logo do começo do arquivo, servem para importar o arquivo **BROKEN-DATABASE.JSON** para que seja feita toda tratativa dos erros.



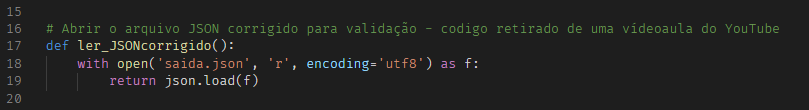
A chamada essa função é feita por esse comando:



Após toda a tratativa, é utilizada uma função para exportar para um arquivo chamado **SAIDA.JSON**



É usado mais uma vez uma função de import, para usar o arquivo **SAIDA.JSON** como input das funções de ordenação e função para a soma dos valores dos itens dos produtos.

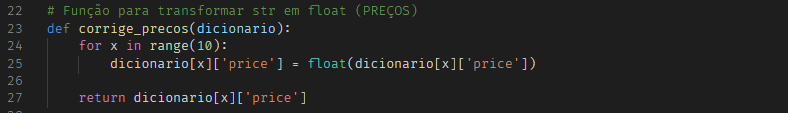


Esses comandos para importação/exportação foram feitos com base na videoaula no canal Igor Lemões

* Importar: <https://www.youtube.com/watch?v=pHOoCzvfDHY>
* Exportar: <https://www.youtube.com/watch?v=p3cH8i0ON48>

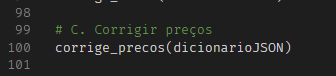
Arquivos JSON importados para o Python, são transformados em listas do Python.

A próxima função é para corrigir os valores de certos produtos que estavam como **TEXTO**

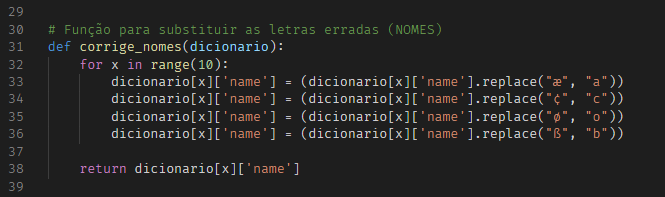
****

A maneira feita para correção dos valores foi até que simples. Usando um comando **FOR** para percorrer toda a lista criada, onde a cada passagem os valores no atributo “**price**” eram convertidos para **FLOAT**.

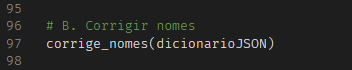
A chamada dessa função é feita pelo comando, retornando a lista com todos os valores do atributo “**price**” arrumados.



Para a correção dos nomes dos produtos, onde algumas letras foram substituídas por certos caracteres, foi utilizado uma ideia parecida com a função de correção de valores:



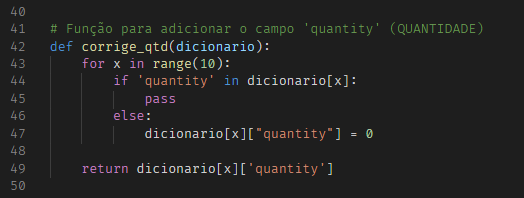
USado um **FOR** para percorrer toda a lista e utilizado um **replace**, onde cada letra errada erra substituída pela letra correta.



Ao final era retornado a lista com todos os nomes corrigidos.

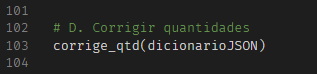
Olha o **FOR** novamente!

Para os produtos que a quantidade estavam com 0 e o atributo “**quantity**” não aparecia foi utilizado um **FOR** junto com um **IF.**

****

Nesse comando **IF** era checado se o atributo “**quantity**” estava presente, caso o **IF** encontrasse esse atributo ele encerrava o **IF** sem passar pelo **ELSE**, caso não houvesse o atributo “**quantity**” era adicionado ao produto com valor 0.

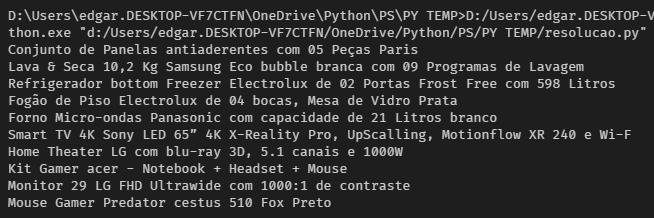
E retornava ao comando que chamou essa função, a lista com as quantidades corrigidas.



Após passar por essas três funções, todos os erros foram corrigidos. Para checar, foi necessário apenas dar um print um cada atributo:

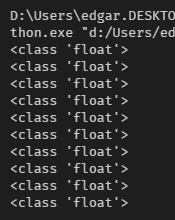
Nomes:





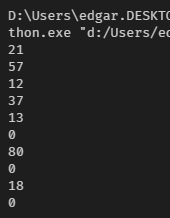
Preços:



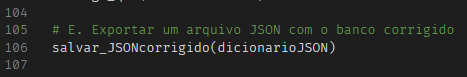


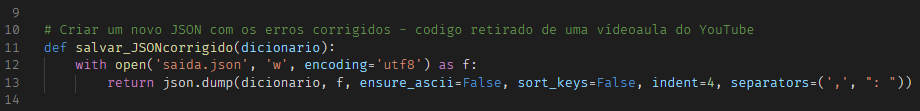
Quantidade:





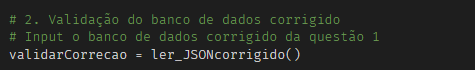
Logo após essa checagem rápida, foi necessário exportar essas informações para o arquivo **SAIDA.JSON**



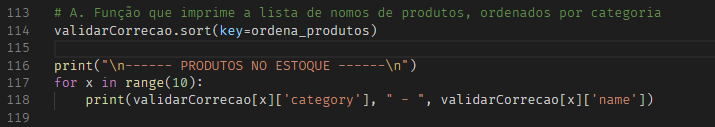


**Referente as validações do arquivo JSON gerado:**

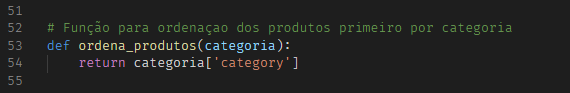
Para a validação dos dados gerados pelas funções anteriores, primeiro foi necessário importar o arquivo **SAIDA.JSON** para que pudesse ser usado com input.



A primeira função que é executada logo após a importação do JSON para a variável **validarCorrecao** é a que organiza as categorias por ordem alfabética e logo após é impresso no terminal os produtos já em ordem por categoria.

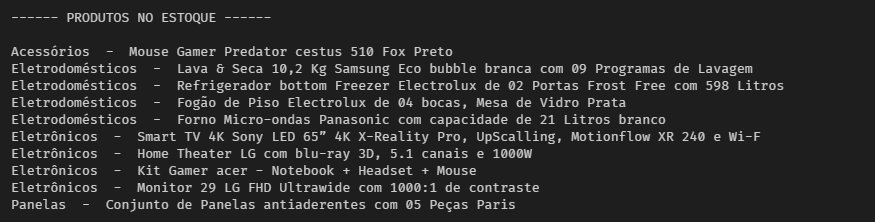


Essa função foi criada com base num exemplo retirado do W3School, para ordenação de listas:

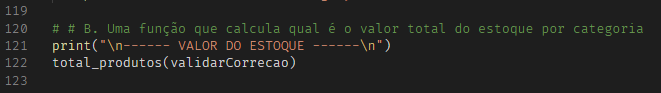


* Referência usada: <https://www.w3schools.com/python/trypython.asp?filename=demo_ref_list_sort5>

Sendo sincero, eu não consegui achar ou elaborar uma maneira para o que foi pedido no teste, que além de ordenar por categoria, ordenar também pelo ID dos produtos. Assim sendo, ficou organizado apenas por categoria.



E a última função utilizada para validação dos dados, é a função para soma dos produtos por categoria levando em conta a quantidade disponível dos produtos.

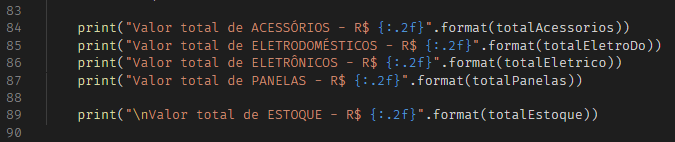




Explicando melhor o código:

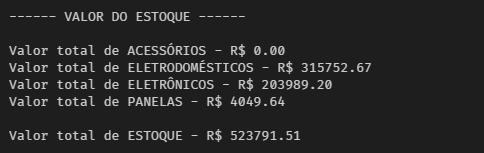
É recebido pela função a lista que foi importado do arquivo **SAIDA.JSON**. É usado um **FOR** junto com um **IF** para percorrer a lista recebida, onde em cada **IF** é checado qual o dado que consta no atributo “**cateogory**”, logo após pega o valor do “**price**” e multiplica pela “**quantity**” onde o resultado é armazenado numa variável criada localmente nessa função.

As últimas linhas:

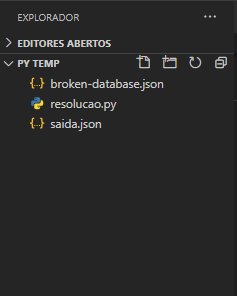


Escrevem no console o resultado dos valores do estoque por cada categoria.

Um adendo, também é mostrado qual o valor total do estoque (somando o valor de todas as categorias).



Arquivos usados e gerados nesse projeto.



Um adendo, eu só consegui fazer com que o projeto importe os arquivos JSON para o código caso a **pasta do projeto** nessa aberta. Se os arquivos forem abertos separadamente, um erro aparecerá.

