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.

```
3
4   # Abrir o arquivo JSON corrompido - codigo retirado de uma videoaula do YouTube
5   def ler_JSONcorrompido():
6       with open('broken-database.json', 'r', encoding='utf8') as f:
7           return json.load(f)
8
```

A chamada essa função é feita por esse comando:

```
# 1. Recuperação dos dados originais do banco de dados
# A. Ler o arquivo Json do banco de dados corrompido
dicionarioJSON = ler_JSONcorrompido()
```

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

```
9
10  # Criar um novo JSON com os erros corrigidos - codigo retirado de uma videoaula do YouTube
11  def salvar_JSONcorrigido(dicionario):
12      with open('saida.json', 'w', encoding='utf8') as f:
13          return json.dump(dicionario, f, ensure_ascii=False, sort_keys=False, indent=4, separators=(',', ": "))
14
```

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.

```
15
16  # Abrir o arquivo JSON corrigido para validação - codigo retirado de uma videoaula do YouTube
17  def ler_JSONcorrigido():
18      with open('saida.json', 'r', encoding='utf8') as f:
19          return json.load(f)
20
```

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**

```
22  # Função para transformar str em float (PREÇOS)
23  def corrige_precos(dicionario):
24      for x in range(10):
25          dicionario[x]['price'] = float(dicionario[x]['price'])
26
27      return dicionario[x]['price']
```

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.

```
 98
 99   # C. Corrigir preços
100   corrige_precos(dicionarioJSON)
101
```

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:

```
29
30  # Função para substituir as letras erradas (NOMES)
31  def corrige_nomes(dicionario):
32      for x in range(10):
33          dicionario[x]['name'] = (dicionario[x]['name'].replace("æ", "a"))
34          dicionario[x]['name'] = (dicionario[x]['name'].replace("ç", "c"))
35          dicionario[x]['name'] = (dicionario[x]['name'].replace("ø", "o"))
36          dicionario[x]['name'] = (dicionario[x]['name'].replace("ß", "b"))
37
38      return dicionario[x]['name']
39
```

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

```
95
96   # B. Corrigir nomes
97   corrige_nomes(dicionarioJSON)
98
```

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.**

```
40
41  # Função para adicionar o campo 'quantity' (QUANTIDADE)
42  def corrige_qtd(dicionario):
43      for x in range(10):
44          if 'quantity' in dicionario[x]:
45              pass
46          else:
47              dicionario[x]["quantity"] = 0
48
49      return dicionario[x]['quantity']
50
```

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.

```
101
102  # D. Corrigir quantidades
103  corrige_qtd(dicionarioJSON)
104
```

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

Nomes:

```
for x in range(10):
    print(dicionarioJSON[x]['name'])
```

```
D:\Users\edgar.DESKTOP-VF7CTFN\OneDrive\Python\PS\PY TEMP>D:/Users/edgar.DESKTOP-V
thon.exe "d:/Users/edgar.DESKTOP-VF7CTFN/OneDrive/Python/PS/PY TEMP/resolucao.py"
Conjunto de Panelas antiaderentes com 05 Peças Paris
Lava & Seca 10,2 Kg Samsung Eco bubble branca com 09 Programas de Lavagem
Refrigerador bottom Freezer Electrolux de 02 Portas Frost Free com 598 Litros
Fogão de Piso Electrolux de 04 bocas, Mesa de Vidro Prata
Forno Micro-ondas Panasonic com capacidade de 21 Litros branco
Smart TV 4K Sony LED 65" 4K X-Reality Pro, UpScalling, Motionflow XR 240 e Wi-F
Home Theater LG com blu-ray 3D, 5.1 canais e 1000W
Kit Gamer acer - Notebook + Headset + Mouse
Monitor 29 LG FHD Ultrawide com 1000:1 de contraste
Mouse Gamer Predator cestus 510 Fox Preto
```

Preços:

```
for x in range(10):
    print(type(dicionarioJSON[x]['price']))
```

```
D:\Users\edgar.DESKTO
thon.exe "d:/Users/ed
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
<class 'float'>
```

Quantidade:

```
for x in range(10):
    print(dicionarioJSON[x]['quantity'])
```

```
D:\Users\edgar.DESKTO
thon.exe "d:/Users/ed
21
57
12
37
13
0
80
0
18
0
```

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

```
104
105    # E. Exportar um arquivo JSON com o banco corrigido
106    salvar_JSONcorrigido(dicionarioJSON)
107
```

```
 9
10  # Criar um novo JSON com os erros corrigidos - codigo retirado de uma videoaula do YouTube
11  def salvar_JSONcorrigido(dicionario):
12      with open('saida.json', 'w', encoding='utf8') as f:
13          return json.dump(dicionario, f, ensure_ascii=False, sort_keys=False, indent=4, separators=(',', ": "))
14
```

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.

```
# 2. Validação do banco de dados corrigido
# Input o banco de dados corrigido da questão 1
validarCorrecao = ler_JSONcorrigido()
```

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.

```
113  # A. Função que imprime a lista de nomos de produtos, ordenados por categoria
114  validarCorrecao.sort(key=ordena_produtos)
115
116  print("\n------ PRODUTOS NO ESTOQUE ------\n")
117  for x in range(10):
118      print(validarCorrecao[x]['category'], " - ", validarCorrecao[x]['name'])
119
```

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

```
51
52  # Função para ordenaçao dos produtos primeiro por categoria
53  def ordena_produtos(categoria):
54      return categoria['category']
55
```

- 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.

```
------ PRODUTOS NO ESTOQUE ------

Acessórios  -  Mouse Gamer Predator cestus 510 Fox Preto
Eletrodomésticos  -  Lava & Seca 10,2 Kg Samsung Eco bubble branca com 09 Programas de Lavagem
Eletrodomésticos  -  Refrigerador bottom Freezer Electrolux de 02 Portas Frost Free com 598 Litros
Eletrodomésticos  -  Fogão de Piso Electrolux de 04 bocas, Mesa de Vidro Prata
Eletrodomésticos  -  Forno Micro-ondas Panasonic com capacidade de 21 Litros branco
Eletrônicos  -  Smart TV 4K Sony LED 65" 4K X-Reality Pro, UpScalling, Motionflow XR 240 e Wi-F
Eletrônicos  -  Home Theater LG com blu-ray 3D, 5.1 canais e 1000W
Eletrônicos  -  Kit Gamer acer - Notebook + Headset + Mouse
Eletrônicos  -  Monitor 29 LG FHD Ultrawide com 1000:1 de contraste
Panelas  -  Conjunto de Panelas antiaderentes com 05 Peças Paris
```

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.

```
119
120   # # B. Uma função que calcula qual é o valor total do estoque por categoria
121   print("\n------ VALOR DO ESTOQUE ------\n")
122   total_produtos(validarCorrecao)
123
```

```
56
57  # Função para o valor total dos produtos por categoria - usando o arquivo saida.JSON
58  def total_produtos(valorProdutos):
59      totalEletroDo = 0
60      totalEletrico = 0
61      totalPanelas = 0
62      totalAcessorios = 0
63      totalEstoque = 0
64
65      for x in range(10):
66          if valorProdutos[x]['category'] == "Eletrodomésticos":
67              totalEletroDo = totalEletroDo + \
68                  (valorProdutos[x]['quantity'] * valorProdutos[x]['price'])
69
70          if valorProdutos[x]['category'] == "Eletrônicos":
71              totalEletrico = totalEletrico + \
72                  (valorProdutos[x]['quantity'] * valorProdutos[x]['price'])
73
74          if valorProdutos[x]['category'] == "Panelas":
75              totalPanelas = totalPanelas + \
76                  (valorProdutos[x]['quantity'] * valorProdutos[x]['price'])
77
78          if valorProdutos[x]['category'] == "Acessórios":
79              totalAcessorios = totalAcessorios + \
80                  (valorProdutos[x]['quantity'] * valorProdutos[x]['price'])
81
82      totalEstoque = totalAcessorios + totalEletrico + totalEletroDo + totalPanelas
83
84      print("Valor total de ACESSÓRIOS - R$ {:.2f}".format(totalAcessorios))
85      print("Valor total de ELETRODOMÉSTICOS - R$ {:.2f}".format(totalEletroDo))
86      print("Valor total de ELETRÔNICOS - R$ {:.2f}".format(totalEletrico))
87      print("Valor total de PANELAS - R$ {:.2f}".format(totalPanelas))
88
89      print("\nValor total de ESTOQUE - R$ {:.2f}".format(totalEstoque))
90
```

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:

```
83
84      print("Valor total de ACESSÓRIOS - R$ {:.2f}".format(totalAcessorios))
85      print("Valor total de ELETRODOMÉSTICOS - R$ {:.2f}".format(totalEletroDo))
86      print("Valor total de ELETRÔNICOS - R$ {:.2f}".format(totalEletrico))
87      print("Valor total de PANELAS - R$ {:.2f}".format(totalPanelas))
88
89      print("\nValor total de ESTOQUE - R$ {:.2f}".format(totalEstoque))
90
```

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

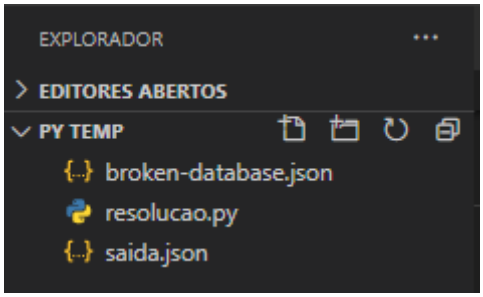
Um adendo, também é mostrado qual o valor total do estoque (somando o valor de todas as categorias).

```
------ VALOR DO ESTOQUE ------

Valor total de ACESSÓRIOS - R$ 0.00
Valor total de ELETRODOMÉSTICOS - R$ 315752.67
Valor total de ELETRÔNICOS - R$ 203989.20
Valor total de PANELAS - R$ 4049.64

Valor total de ESTOQUE - R$ 523791.51
```

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.

```
Traceback (most recent call last):
  File "d:/Users/edgar.DESKTOP-VF7CTFN/OneDrive/Python/PS/PY TEMP/resolucao.py", line 94, in <module>
    dicionarioJSON = ler_JSONcorrompido()
  File "d:/Users/edgar.DESKTOP-VF7CTFN/OneDrive/Python/PS/PY TEMP/resolucao.py", line 6, in ler_JSONcorrompido
    with open('broken-database.json', 'r', encoding='utf8') as f:
FileNotFoundError: [Errno 2] No such file or directory: 'broken-database.json'
```