SIMULADO 4

Leia todo o enunciado antes de começar a solução.

Considere os arquivos **'tabela.txt' e 'info.txt'**, descritos a seguir.

O arquivo **'tabela.txt'** contém valores em ponto flutuante (parte decimal separada por ponto), não negativos, no formato de 4 colunas e um número arbitrário de linhas (igual para todas as colunas). Os dados contidos em cada linha são separados por um caractere de espaço.

O arquivo **'info.txt**' contém 1 coluna de valores em ponto flutuante (parte decimal separada por ponto), com um número arbitrário de linhas. O padrão esperado dos dados é que não haja valores negativos.

Elabore um programa em Python que execute as seguintes tarefas:

a) O programa deve verificar se os valores no arquivo **'info.txt**' são não negativos. Caso não atendam o padrão esperado, deve ser escrita uma mensagem de alerta na tela, e a execução do programa deve ser encerrada. Use uma função (definida no programa) para executar tal verificação.

b) No caso da execução continuar, deve ser gerado um novo arquivo chamado '**saida.txt'** contendo 3 colunas separadas por espaço, com as seguintes informações:

i) primeira coluna: valores da terceira coluna do arquivo **'tabela.txt'**, em ordem crescente, com 6 casas decimais;

ii) segunda coluna: valores contidos no arquivo **'info.txt**', em ordem crescente, com 6 casas decimais. Note: esta coluna deve ter o mesmo número de linhas que a primeira. Caso o arquivo 'info' possua uma quantidade de valores insuficiente, complete com null. Caso possua mais, selecione apenas o necessário;

iii) terceira coluna: valor do erro absoluto entre as duas colunas anteriores, com 6 casas decimais. Use uma função (definida no programa) para calcular o erro. Quando o erro não puder ser calculado, escreva null.

Regras de implementação e considerações úteis:

1. O programa deve implementar tratamento de erro na abertura de arquivos. O programa deve escrever uma mensagem na tela em caso de erro na leitura ou escrita de arquivos, **indicando o nome do arquivo com problema.**
2. Os arquivos de entrada não devem ser alterados (nem o nome nem o conteúdo).
3. Considere que os arquivos de entrada possuem a formatação correta de acordo com a definição do enunciado.
4. O programa deve gerar somente o arquivo de saída solicitado, **mantendo o nome e a extensão**.
5. Considere que os arquivos manipulados encontram-se no mesmo diretório de execução do código.
6. Apenas bibliotecas padronizadas da linguagem são permitidas.
7. Encontram-se em anexo exemplos de arquivos de entrada e o arquivo de saída correspondente, de modo a facilitar a verificação do resultado. Mas, o programa não deve ser elaborado de forma particularizada para o exemplo fornecido.
8. O programa não deve solicitar nenhuma digitação ao usuário.

9. O que não estiver explicitado no enunciado é de escolha do programador.

   Dica: Os dados obtidos dos arquivos de entrada **são mais facilmente manipulados** por meio de **arrays NumPy**.