Computação I

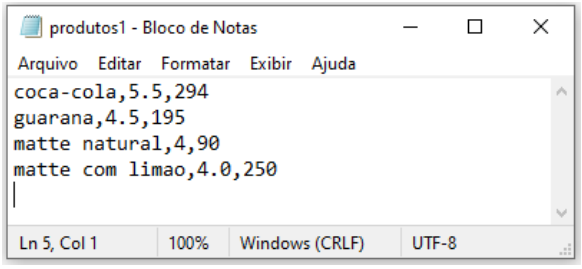
Lista de exercícios 7 – Interação com o usuário em um programa mais completo, dicionários, módulos e arquivos
Atenção! Leia as instruções antes de fazer a lista! - Data de entrega: 29/06/2022
Por favor, não deixe de respeitar a padronização do nome do arquivo e dos nomes das funções, pois isso nos ajuda na correção.
A menos que esteja explicitamente pedido na questão, não utilize nenhum método ou função já existente do Python exceto pelo print, type, len, range, input, append, format, upper, lower, split (de forma simplificada, função é tudo o que precisa ser chamado com parênteses, e método é tudo o que precisa ser chamado com parênteses, mas que está associado a uma variável, isto é, variavel.metodo(args), por exemplo, o len é uma função e o upper e o lower são métodos). As funções de transformação de tipo int, float e str são permitidas nessa lista. Importe somente o módulo datetime. Não é necessário testar se os dados passados através do input e se os valores dos arquivos são válidos a menos que pedido na questão (considere que os valores passados por input podem ser convertidos para inteiro, quando necessário, ou float, quando necessário, que os arquivos necessariamente seguirão a padronização, e que sempre haverá um pulo de linha na última linha do arquivo). **Atenção para a diferença entre o return e o print.** O return permite que o valor seja utilizado fora da função, enquanto o print só imprime na tela. Somente a função principal do seu código deve terminar com o número 1.

Para ajudar na correção, é muito importante verificar, no arquivo de testes, os exemplos de funcionamento do código. Sua função deve funcionar exatamente como os exemplos. Não adicione nenhum input além dos pedidos.

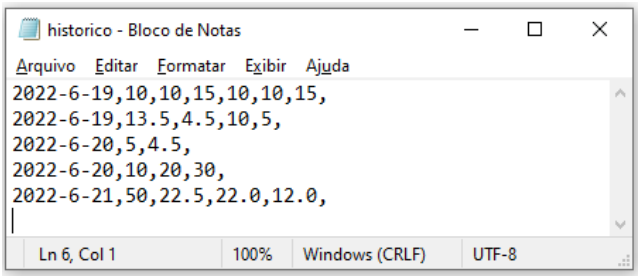
O método split, do tipo string, pode ajudar muito nas questões abaixo. Ele funciona da seguinte forma: considerando uma variável do tipo string **s**, quando utilizamos o split aplicado ao **s** a sintaxe é **s.split(separador)**, onde separador é uma outra string. O split retorna uma lista de strings com pedaços de **s** entre a string separadora. Isto é, se **s = Fernanda.Oliveira**, então **s.split('.')** retorna a lista **['Fernanda' , 'Oliveira']** (note que o separador em si não aparece no valor de retorno). Veja alguns exemplos realizados no shell do Python:

```
>>> s = 'Fernanda.Oliveira'
>>> nomes = s.split('.')
>>> nomes
['Fernanda', 'Oliveira']
>>> frase = 'Ola! Meu nome eh Fernanda'
>>> palavras = frase.split(' ')
>>> palavras
['Ola!', 'Meu', 'nome', 'eh', 'Fernanda']
>>> # Veja que a variavel com a string original nao muda:
>>> frase
'Ola! Meu nome eh Fernanda'
>>> # Quando o separador estah na ultima posicao da string, aparece uma
string vazia na ultima posicao da lista resultante:
>>> teste1 = frase.split('a')
>>> teste1
['Ol', '! Meu nome eh Fern', 'nd', '']
>>> # Quando o separador nao existe na string, o resultado eh uma lista
com a string completa:
>>> teste2 = frase.split('A')
>>> teste2
['Ola! Meu nome eh Fernanda']
>>># O exemplo abaixo pode ajudar nas questoes da lista
>>>linha = 'matte natural,4,90'
>>>info = linha.split(',')
>>>info
   ['matte natural', '4', '90']
>>>linha = '2022-6-21,50,22.5,22.0,12.0,\n'
>>>info = linha.split(',')
>>>info
   ['2022-6-21', '50', '22.5', '22.0', '12.0', '\n']
```

1. Escreva um programa em Python para ajudar no gerenciamento de uma loja. Seu código deve importar o módulo datetime, pois ele será usado para que um histórico de vendas do dia seja cadastrado. O programa deve estar dividido em diversas funções, conforme pedido abaixo. A função principal deve receber por argumento uma string com o nome de um arquivo com as informações dos produtos; e uma outra string com o nome de um arquivo com o histórico de vendas. O arquivo de produtos contém, em cada linha, o nome do produto, o preço, e a quantidade de itens em estoque. Essas informações encontram-se separados por vírgula, como no exemplo:



Não haverá duas linhas com o mesmo nome de produto. O arquivo de histórico contém, a cada linha, uma data no formato ano-mês-dia, separados por hífen, e o valor de cada compra realizada naquele dia. A data e o valor de cada compra devem separados por vírgula. A mesma data pode aparecer em mais de uma linha diferente (como se o programa tivesse sido aberto e fechado a cada troca de turno, por exemplo), como no exemplo:



A função principal, cujo nome deve terminar com o número 1 (atenção, nenhuma outra função pode terminar com o número 1), deve, logo no começo, fazer a leitura do arquivo de produtos e criar dois dicionários: o primeiro deve conter como chaves os nomes dos produtos e como valores, floats que representam os preços; e o segundo deve conter como chaves os nomes dos produtos e como valores, inteiros que representam a quantidade dos itens no estoque. A função principal deve conter uma tupla de valores das compras. Essa tupla deve ser definida como uma tupla vazia no início da função principal. Em seguida, a função principal deve mostrar um menu com as opções: (1) Realizar uma venda; (2) Alterar preço de um produto; (3) Alterar a quantidade de itens de um produto; (4) Adicionar um novo produto; (5) Imprimir relatório de vendas de um dia; (6) Sair. Todas as seis opções devem, necessariamente, ser implementadas em funções que são chamadas pela função principal. O programa só deve ser encerrado quando a opção 6 for digitada. Para cada opção é esperado o seguinte comportamento:

Opção 1: interage com o usuário para saber o nome do produto comprado. Se o usuário digitar um produto que não existe (deve reconhecer o nome do produto independente das minúsculas e maiúsculas), informe-o, e pergunte se deseja comprar algum outro produto (s/n). Se o usuário digitar ‘s’, então repete a interação. Se o usuário digitar ‘n’, calcula e imprime o preço total da compra. Interage para perguntar a quantidade. Verifique se a quantidade comprada é maior que a quantidade no estoque. Se for maior, informe ao usuário. E volte a perguntar se deseja comprar algum outro produto (s/n). Se o usuário digitar ‘s’, então repete a interação desde o começo. Uma vez encerrada a compra calcule e imprima o preço total. Além disso, o dicionário de nomes de produtos e de quantidade no estoque deve ser atualizado, e o preço total da compra deve ser incluído no final da tupla com o histórico de valores de compras.

Opção 2: interage com o usuário para perguntar o nome do produto cujo preço deve ser alterado. Se o produto não existir, informe ao usuário que é necessário primeiro adicionar o produto e volte ao menu principal. Se o produto existir (deve reconhecer o nome do produto independente das minúsculas e maiúsculas), peça o preço atualizado e atualize o dicionário de produtos e preços.

Opção 3: interage com o usuário para perguntar o nome do produto cuja quantidade deve ser alterada. Se o produto não existir, informe ao usuário que é necessário primeiro adicionar o produto e volte ao menu principal. Se o produto existir (deve reconhecer o nome do produto independente das minúsculas e maiúsculas), peça a quantidade atualizada e atualize o dicionário de produtos e quantidades.

Opção 4: interage com o usuário para perguntar o nome do produto que deseja adicionar. Se o produto já for um produto que existe no dicionário (deve reconhecer o nome do produto independente das minúsculas e maiúsculas) informe ao usuário e volte ao menu principal. Se o produto não existir, peça seu preço e quantidade (em inputs separados), e adicione o novo produto a ambos dicionários. Os produtos adicionados nos dicionários devem estar escritos somente com letras minúsculas.

Opção 5: Interage com o usuário para pedir o dia, mês e ano desejados (em inputs separados), abre o arquivo de histórico, imprime cada linha com a data pedida, calcula e imprime o total de vendas da data pedida considerando as informações de todas as linhas encontradas. (Não esqueça de fechar o arquivo)

Opção 6: apaga as informações que existiam no arquivo de produtos (abrindo-o com ‘w’) e escreve as novas informações, contidas nos dicionários, mantendo a padronização que foi definida para esse arquivo. Escreve, no final do arquivo com histórico de vendas (abra o arquivo de forma a permitir a escrita no final, sem apagar o que já estava escrito), a data atual (utilize o módulo datetime para descobrir a data atual) no formato ano-mês-dia e os valores das compras guardados na tupla. Certifique-se de que cada informação presente na linha (data e cada valor de compra do dia) esteja separada por vírgulas, de forma a manter a padronização do arquivo. Retorna os dicionários (os tipos dos dados nos dicionários devem ser respeitados).

Obs.: Veja que, como as vendas só são salvas ao sair do programa, se a opção 5 for rodada e o dia atual for pedido, as vendas feitas enquanto o programa está aberto não serão impressas. A opção 5 só imprime o que já foi salvo.