Aplicando listas à solução de problemas

Agora que você já sabe as operações básicas de criação, edição, combinação e acesso em listas, e entende como buscar e ordenar elementos, está na hora de analisar alguns exemplos dentro do ambiente Jupyter Notebook, para ter uma compreensão melhor de como todas essas possiblidades podem ser usadas para resolver problemas reais. Vamos lá?

Quando você usa listas, além de todas as operações como inserir, remover e alterar elementos, uma das coisas mais simples e úteis que pode ser feita é visitar todos os componentes do começo ao fim, aplicando alguma operação sobre eles. Por exemplo, se você tiver uma lista das previsões de temperatura para cada dia da próxima semana, pode gerar um breve relatório, informando se o dia será confortável ou quente, supondo que nenhum dia vai chegar a ser frio. Para isso, primeiro, defina uma lista de temperaturas. Para servir de exemplo, temos a linha 1, onde estão definidas as temperaturas 23,4◦C, 25,2◦C, 20.0◦C, 22.1◦C, 27.5◦C, 26.2◦C e 25.8◦C. Cada uma representa um dia da semana.

Depois de definida, para percorrer essa lista, adicione um laço for com a variável temp, assumindo o valor de cada temperatura. Dentro do for, use um if, para distinguir entre um dia quente (acima de 25 graus) e um dia confortável. Caso temp for maior que 25, imprima a mensagem dia quente, seguida da temperatura. Do contrário, imprima dia confortável, também seguido da temperatura. O resultado que você pode verificar mostra 7 linhas, uma para cada temperatura, do começo ao fim da lista, em que cada temperatura é definida como dia confortável ou dia quente.

A saída exata pode variar se você usar valores diferentes dos exemplificados na lista de temperatura que foi usada, mas, de qualquer forma, ao final, irá apresentar uma linha para cada temperatura com uma mensagem que classifica o dia correspondente, como quente ou confortável. Bem, nesse momento que você já compreendeu como percorrer uma lista, entenda o cálculo da soma de alguns dos seus elementos.

Em casos simples, quando você quer somar a lista inteira, basta usar a função sum. Entretanto, quando nem todos os números devem ser somados, a função sum não irá servir. Por exemplo, no código a seguir, há duas listas nas linhas 1 e 2: a primeira é uma lista de gastos para uma festa, com gastos = [15.00, 57.50, 83.25, 11.75, 2.0, 52.50, 97.75, 72.25]; e a segunda é uma lista de valores lógicos, do mesmo tamanho da primeira, em que cada elemento indica se o gasto da posição correspondente é essencial ou não. Dessa maneira, a lista é representada por essencial = [True, False, False, True, True, True, False, True].

Para calcular as somas de gastos essenciais e não essenciais, deve-se definir duas variáveis separadas, como nas linhas 4 e 5: gastos_essencial e gastos_não_essencial. Ambas com valor inicial 0. Em seguida, adicione um laço com um número de passos igual ao tamanho da lista de gastos, ou seja, len(gastos). Portanto, a variável i será usada para indexar ambas as listas de maneira coordenada. Assim, a linha de código deve ser for i in range(len(gastos)).

Depois, adicione um comando if, para verificar se o gasto é essencial ou não, usando a lista essencial indexada na posição i para isso. De acordo com o resultado, some o gasto correspondente, dado por gastos[i] à variável correta. Com isso, cada variável acumula a soma apenas dos valores apropriados.

Por fim, você pode usar um par de comandos print ao final para verificar o resultado. Ao executar este programa, com estes valores, o resultado será o par de linhas, que dizem, "Gasto essencial: 153.5" e "Gasto não essencial: 238.5". Bem simples, não acha?

Esta é uma forma de distribuir informação em listas de mesmo tamanho, em que cada lista guarda uma parte da informação e uma lista pode ser usada para fazer uma triagem da outra. Neste caso, gastos é uma lista de valores monetários e essencial outra lista que diz se o valor é essencial ou não. Acompanhe a seguir, um outro exemplo de problema, em que é exigido uma lista flexível.

Então, alguns programas de computador processam dados obtidos, automaticamente, da internet ou de sensores, mas muitos são totalmente governados pela interação com o usuário. Por exemplo, é razoável que um programa, para criar e manter uma lista de compras, só adicione e remova itens, quando você o pede para fazer isso.

A partir dessas informações, será abordada uma técnica simples para criar uma lista de tamanho flexível: a cada rodada, o usuário responde primeiro se quer adicionar novos itens; quando disser que sim, o novo item é lido e adicionado; quando disser que não, as repetições terminam.

Para isso, crie, na linha 1, uma lista vazia para guardar os itens, escrevendo assim itens = []. Em seguida, a linha "cont = input(Continuar? (S/N)').upper()", que implementa uma pergunta com resposta Sim ou Não, para ser feita ao usuário. O método upper é aplicado à string para transformá-la em caixa alta, padronizando a resposta, caso letras minúsculas sejam digitadas. Após isso, o programa entra no laço while se a primeira resposta for "S".

Na continuação, tem-se as linhas 5 a 7, onde o programa lê o novo item, usando a função input e o adiciona à lista itens, mas apenas se ele não for uma string vazia. Por último, o programa repete a pergunta da linha 2, para que o laço while da linha 2 saiba se deve continuar ou não, e, ao final do laço while, imprima a lista construída.

Feito isso, ao executar esse programa, você poderá realizar testes. Ao responder "S" à pergunta inicial “Continuar?”, o programa pedirá um item, digitando "Feijão" e Enter, a primeira pergunta é repetida; respondendo "S" mais três vezes e informando os itens "Arroz", "Batata" e "Cenoura" respectivamente. Por último, se responder "N" o programa encerra, imprimindo uma lista com uma sequência de itens que foram adicionados, a partir da resposta “S”, que, nesse caso, será uma lista com feijão, arroz, batata e cenoura. Pronto, você pôde entender como desenvolver esse programa e praticar, para criar sua própria lista de produtos.

Agora que você já aprendeu como percorrer uma lista, usar listas paralelas, para particionar informação, e construir uma lista interativamente, está na hora de combinar estas

técnicas com operações de busca para desenvolver um programa que controla uma lista de compras com diferentes quantidades para cada produto.

Para isso, defina, primeiramente, duas listas paralelas nas linhas 1 e 2: uma para os nomes dos produtos, nesse caso, usaremos os produtos aveia, sabão em pó, manteiga e cebola; e outra para as quantidades, contando que aqui será exatamente [1, 1, 1, 1], onde cada número representa, respectivamente, as quantidades de cada produto da lista anterior.

Em seguida, escreva uma string multi-linha, com menu = """\Escolha uma opção: (1) Alterar quantidade (2) Sair""". Esta mensagem, pedindo para que o usuário escolha entre alterar quantidade ou sair, será exibida a cada rodada para que ele decida o que fazer.

Depois de apresentar o menu e obter a resposta do usuário com a função input, converta a resposta com int e adicione um laço while, que só termina quando a opção 2 do menu é selecionada.

Após isso, faça uma linha para definir o que deve ser feito caso a opção 1, "alterar quantidade", seja selecionada, usando if. Com input, será perguntado ao usuário qual produto deve ser alterado. Também inclua, no código, um if adicional para verificar se o produto digitado pelo usuário realmente existe na lista produtos, protegendo contra erros de digitação.

Já a posição do produto pode ser determinada pelo método index em pos = produtos.index(prod), o que permite também saber a quantidade atual do produto indicado, localizada na variável quantidades[pos].

Após imprimir a quantidade atual para o usuário, pergunte a nova quantidade, usando input e int, já que é um número inteiro, conforme consta na linha nova_quant = int(input('Nova quantidade? ')).

Dando sequência, use um if para verificar se a resposta é 0. Neste caso, será removido o produto e sua quantidade das duas listas, com o método pop. Caso contrário, é só atualizar a lista quantidades nos índices pos, e, assim, a lista produtos não precisará ser alterada. Dessa forma, as duas mantêm sempre o mesmo tamanho e os itens paralelos.

Nesse momento, já é possível fechar o if da linha 13 que contém if prod in produtos: com aquilo que acontece, caso o produto informado não seja encontrado. Nesse caso é só imprimir uma mensagem de erro, como a informação no código “Ops, a lista não contém”, para informar, ao usuário, que o produto digitado não está na lista.

Além disso, existe a possibilidade de fechar o if da linha 10 com um else, informando ao usuário, que ele colocou uma opção inválida, caso seja inserido algo diferente de 1 ou 2, já que o menu só tem essas duas opções. E, fora do else, mas ainda dentro do while, será repetida a pergunta da linha 9, para que o laço while, na linha 10, saiba se deve continuar após a próxima resposta do usuário.

Tudo bem até aqui? Se estiver, você já entendeu o mais importante. Agora, basta adicionar um for para imprimir os produtos finais e suas quantidades. Usando a variável i para percorrer os

índices válidos, dados pela função range(len(produtos)). Assim será possível recuperar o produto e sua quantidade das listas respectivas na posição i e imprimir estes dados.

Vamos testar o programa?

Inicialmente, você terá a exibição do menu, e será necessário escolher entre alterar quantidade e sair. Assim, imaginemos que você escolheu a primeira opção e, em seguida, informou o produto “aveia” para a pergunta "Qual produto?". Como este produto existe, o programa diz "aveia tem 1 unidade" e é questionado se você deseje alterar essa quantidade. Então, você pode responder com o número 2, caso deseja informar ao programa, que existe duas unidades de aveia.

Alterada a quantidade de aveia, o programa torna a te fazer a pergunta inicial para escolher entre a opção 1 – Alterar quantidade; ou 2- Sair. Você poderá repetir o processo, entrando com "1", para a opção; "manteiga", para o produto; e "3", para a nova quantidade. Mais uma vez, escolherá "1" como opção, mas, agora, informará "detergente", que não está na lista produtos. Consequentemente, o programa vai imprimir "Ops, a lista não contém detergente". Então, ele tornará a te fazer a pergunta inicial, e, dessa vez, você escolherá 2 para sair e encerrar o programa.

Por fim, são impressas as quantidades finais de cada produtos que existiam na lista. Um por linha, de acordo com as modificações que foram realizadas nas quantidades. Então será exibido:

2 unidades de aveia

1 de sabão em pó

3 de manteiga

1 de cebola.

Até aqui, você pôde perceber diversas formas de utilizar listas, como percorrer seus elementos, no exemplo das temperaturas; usar uma lista para particionar outra e acumular seus valores seletivamente, demonstrado, no exemplo dos gastos para a festa; além de entender como construir uma lista, a partir da interação com usuários no exemplo da lista de compras; e, por fim, alterar listas pareadas, usando busca e interação com os usuários, neste exemplo final.

Estas são algumas das técnicas que você pode combinar com outras coisas que já sabe para lidar com situações corriqueiras de programação. Não se esqueça de revisar todo o conteúdo e tentar praticar o que aprendeu, para aprofundar seus conhecimentos na linguagem.

Bons estudos e até mais!