**04 - Operações com listas**

Agora que conhecemos as listas do Python, é interessante aprendermos a manipular as informações dentro delas. Como foi dito anteriormente, as listas são mutáveis, ou seja, é possível realizar operações sobre os seus dados, por exemplo adicionando ou retirando itens. Quanto mais recursos um cientista de dados tem para manipular as informações, melhores e mais eficientes serão os seus programas.

Na seção **3.2 Operações com listas** temos, inicialmente, um [link para a documentação do Python](https://docs.python.org/3.6/library/stdtypes.html#common-sequence-operations) onde você encontrará todas operações possíveis com listas. Aqui abordaremos algumas das principais operações utilizadas no dia-a-dia de um cientista de dados.

Primeiro aprenderemos a identificar se um item está dentro de uma lista. Mas qual a utilidade disso? Imagine que temos a seguinte lista de Acessorios de um veículo:

Acessorios = ['Rodas de liga', 'Travas elétricas', 'Piloto automático', 'Bancos de couro', 'Ar condicionado', 'Sensor de estacionamento', 'Sensor crepuscular', 'Sensor de chuva']

AcessoriosCOPIAR CÓDIGO

Estamos interessados em rodar um modelo de previsão de preços somente para carros que tenham determinada característica, por exemplo as "Rodas de liga". Se identificarmos que a lista contém determinado item, poderemos selecionar aquele veículo. Observe que nessa seção encontramos a instrução **x in A**, onde x representa aquilo que estamos buscando, A representa a lista onde buscaremos e in é o comando Python que fará essa execução. Veja o exemplo:

'Rodas de liga' in AcessoriosCOPIAR CÓDIGO

Aqui é como se estivéssemos perguntando "o item Rodas de liga está presente na lista Acessorios?". Ao executarmos esse código, o Python nos retorna um booleano True - ou seja, o item 'Rodas de liga' está sim presente em Acessorios. Vamos refazer o teste, agora para o item "4 x 4", que não está contido na lista Acessorios.

'4 x 4' in AcessoriosCOPIAR CÓDIGO

Dessa vez o retorno será um booleano False, algo que já era esperado. Também podemos fazer essa pergunta "ao contrário" usando a instrução not:]

'Rodas de liga' not in AcessoriosCOPIAR CÓDIGO

O retorno desse código será False, já que "Rodas de liga" está presente em acessórios. Se fizermos o mesmo com "4 x 4", o retorno será verdadeiro.

'4 x 4' not in AcessoriosCOPIAR CÓDIGO

True

Prosseguindo, já aprendemos que a funcionalidade do sinal + pode variar de acordo com o tipo de dado com que estamos trabalhando - quando passamos valores numéricos, fazemos uma soma; quando passamos strings, fazemos a concatenação. No caso das listas também é feita a concatenação, juntando os elementos das duas listas.

No nosso exemplo, a lista Acessorios foi divida em duas listas chamadas A e B, cada uma contendo quatro itens.

A = ['Rodas de liga', 'Travas elétricas', 'Piloto automático', 'Bancos de couro']

B = ['Ar condicionado', 'Sensor de estacionamento', 'Sensor crepuscular', 'Sensor de chuva']COPIAR CÓDIGO

Ao executarmos A + B, será feita a concatenação das listas, resultando em uma nova lista de oito itens.

A + BCOPIAR CÓDIGO

['Rodas de liga', 'Travas elétricas', 'Piloto automático', 'Bancos de couro', 'Ar condicionado', 'Sensor de estacionamento', 'Sensor crepuscular', 'Sensor de chuva']

Outra informação interessante de descobrirmos é o tamanho de uma lista. No momento estamos lidando com poucos itens, sendo possível contá-los "a olho". Entretanto, em data science acabamos trabalhando com conjuntos de inúmeros dados. A função len(), intrínseca do Python, nos permite descobrir facilmente o tamanho de uma lista:

len(Acessorios)COPIAR CÓDIGO

8

Essas são algumas das operações mais importantes e recorrentes no trabalho com data science, e no próximo vídeo aprenderemos a fazer seleções nas listas.