**02 - Built-in Functions**

Nessa seção do treinamento conversaremos sobre funções. As funções são unidades de código reutilizáveis que realizam tarefas específicas. Elas podem receber *inputs*, que são as entradas (também chamadas de parâmetros), e podem retornar resultados.

Começaremos explorando as funções nativas do Python (*built-in functions*), ou seja, funções integradas da linguagem que estão sempre acessíveis, sem a necessidade de criação ou importação. Nós, inclusive, utilizamos várias delas neste treinamento e no anterior, como type(), print(), zip(), len() e assim por diante.

O objetivo aqui não é esgotarmos todas as funções nativas da linguagem, afinal muitas delas não têm utilidade prática para nosso trabalho como cientistas de dados. A ideia é, dependendo das necessidades, procurarmos por funções que nos ajudem a resolver determinados problemas. Se não existirem, poderemos criar as nossas próprias funçoes, algo que será tema dos próximos vídeos.

Caso queira, você pode consultar a [documentação do Python a respeito de funções](https://docs.python.org/3.6/library/functions.html) para se aprofundar nesse tema.

Para começarmos os testes, inicializaremos novamente um dicionário dados contendo alguns pares chave-valor.

dados = {'Jetta Variant': 88078.64, 'Passat': 106161.94, 'Crossfox': 72832.16}

dadosCOPIAR CÓDIGO

{'Crossfox': 72832.16, 'Jetta Variant': 88078.64, 'Passat': 106161.94}

Queremos obter a somatória dos valores dos veículos desse conjunto, algo que pode ser obtido de algumas formas mais complexas e de outras mais simples. Começaremos pelas mais complexas: criaremos uma lista vazia valores e iniciaremos um iterador for chamando cada item encontrado em dados.values() de valor e adicionando tais valores na lista por meio da função append(), que já conhecemos no curso anterior.

valores = []

for valor in dados.values():

valores.append(valor)

valoresCOPIAR CÓDIGO

Desse modo obteremos todos os valores dentro de uma única lista, o que nos permitirá somá-los.

[88078.64, 106161.94, 72832.16]

Uma forma um pouco enxuta de obtermos a somatória é criando uma variável soma que se iniciará com 0, iterando pelos valores com dados.values() e, ao invés de adicioná-los a uma lista, somá-los diretamente à variável criada anteriormente com soma += valor.

soma = 0

for valor in dados.values():

soma += valor

somaCOPIAR CÓDIGO

267072.74

Dentre as *built-in functions* do Python, existe uma que pode tornar nosso código ainda mais simples: sum(), que é justamente "soma" em inglês. Na [documentação da função sum()](https://docs.python.org/3.6/library/functions.html#sum) verificaremos que a função pode receber dois parâmetros: um iterador *iterable* e um *start*. Este último aparece entre colchetes, o que significa duas coisas: que o parâmetro possui um valor padrão e que ele é dispensável - ou seja, que não é necessário fornecê-lo à função para que ela nos retorne um valor.

Antes de utilizarmos essa função, repare na devolução da chamada list(dados.values()):

list(dados.values())COPIAR CÓDIGO

[88078.64, 106161.94, 72832.16]

Aqui conseguimos obter uma lista com todos os valores do nosso dicionário, da mesma maneira que anteriormente, mas utilizando um código bem mais simples e sucinto. A soma (sum()) operará da mesma forma, bastando passarmos como parâmetro o nosso dados.values()

sum(dados.values())COPIAR CÓDIGO

267072.74

Nosso resultado é a soma de todos os valores do dicionário dados, executada de maneira bastante resumida. O parâmetro *start* dessa função significa que podemos iniciar a soma já com algum valor. Isso pode ser feito simplesmente passando o valor desejado após uma vírgula.

sum(dados.values(), 1000000)COPIAR CÓDIGO

Assim, a nossa soma será adicionada ao valor inicial - nesse caso, um milhão.

1267072.7399999998

Quando precisamos entender determinadas funcionalidades, é interessante consultarmos a documentação. O Python possui também uma função nativa help() para qual podemos passar, como parâmetro, a função sobre a qual queremos mais informações - por exemplo, print.

help(print)COPIAR CÓDIGO

Isso nos devolverá a "ajuda" da função, que é uma espécie de documentação resumida.

Help on built-in function print in module builtins: print(...) print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.

Optional keyword arguments:

file: a file-like object (stream); defaults to the current sys.stdout.

sep: string inserted between values, default a space.

end: string appended after the last value, default a newline.

flush: whether to forcibly flush the stream.COPIAR CÓDIGO

Outra forma de acessarmos esse conteúdo no Colab é passarmos, depois do nome da função, uma interrogação.

print?COPIAR CÓDIGO

Isso abrirá uma aba do próprio Colab com as informações da função.

Docstring: print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False) Prints the values to a stream, or to sys.stdout by default. Optional keyword arguments: file: a file-like object (stream); defaults to the current sys.stdout. sep: string inserted between values, default a space. end: string appended after the last value, default a newline. flush: whether to forcibly flush the stream. Type: builtin\_function\_or\_method

Sempre que precisar de uma funcionalidade, primeiramente analise a documentação do Python, pois é possível que ela já exista. Caso contrário, você poderá criará suas próprias funções, algo que aprenderemos nos próximos vídeos.