**06 - Iterando em tuplas**

Já aprendemos a fazer iterações em listas, e as iterações em tuplas não são diferentes. Nesse vídeo trabalharemos um recurso que nos auxiliará quando estivermos usando funções, principalmente as que retornam mais de um valor, e também com alguns tipos de iterações.

Começaremos inicializando a tupla nomes\_carros.

nomes\_carros = ('Jetta Variant', 'Passat', 'Crossfox', 'DS5')

nomes\_carrosCOPIAR CÓDIGO

Para iterarmos por seus itens, usaremos a instrução for, atribuiremos um nome válido a cada item encontrado (por exemplo item) e continuaremos com in seguido do nome da tupla (nomes\_carros). Não podemos nos esquecer de finalizar a linha com dois pontos (:) e da indentação na linha seguinte. Nesse exemplo simplesmente imprimiremos cada um dos item encontrados usando a função print().

for item in nomes\_carros:

print(item)COPIAR CÓDIGO

Jetta Variant Passat Crossfox DS5

Conseguimos varrer cada um dos itens e imprimi-los. O próximo conceito que iremos abordar é o desempacotamento de tuplas, o recurso que citamos anteriormente. Basicamente, é possível atribuirmos cada valor dentro de uma tupla a uma variável específico. Nesse primeiro exemplo, usaremos as variáveis carro\_1, carro\_2, carro\_3 e carro\_4 separadas por vírgula. Essa sequência receberá o conteúdo de nomes\_carros.

carro\_1, carro\_2, carro\_3, carro\_4 = nomes\_carrosCOPIAR CÓDIGO

Como temos quatro carros e quatro variáveis, cada uma delas receberá um dos valores na tupla, na ordem que definimos. Podemos verificar isso mostrando o conteúdo das nossas variáveis:

carro\_1COPIAR CÓDIGO

'Jetta Variant'

carro\_2COPIAR CÓDIGO

'Passat'

carro\_3COPIAR CÓDIGO

'Crossfox'

carro\_4COPIAR CÓDIGO

'DS5'

Esse é o desempacotamento de tuplas, algo útil, por exemplo, quando uma função nos retorna mais de um valor dentro de uma tupla e queremos utilizá-los de formas variadas no nosso programa, separando-os em variáveis diferentes.

Prosseguiremos para outra situação interessante: imagine que uma função nos retorna quatro valores, mas queremos utilizar somente dois deles. Nesse caso, queremos somente o segundo e o último valor da tupla nomes\_carros (a primeira posição e a terceira). Para isso, podemos usar o recurso do *underscore* (ou "*underline*"), da mesma forma que vimos nas listas.

\_, A, \_, B = nomes\_carrosCOPIAR CÓDIGO

Aqui usamos as variáveis A e B para recebermos os valores das posições desejadas, e os *underscores* para delimitar as posições que não desejamos salvar em variáveis específicas, ignorando os seus valores.

ACOPIAR CÓDIGO

'Passat'

BCOPIAR CÓDIGO

'DS5'

Em outra situação, queremos somente um valor da nossa tupla, mas nenhum outro. Imagine se tivéssemos uma tupla com 60 itens e só gostaríamos de salvar o segundo. Com a sintaxe que acabamos de aprender, seria bastante trabalhoso recuperarmos esse valor.

Para simplificar, podemos ignorar o primeiro valor com *underscore*, receber o segundo com uma variável (por exemplo C) e finalizar com asterisco seguido de *underscore* (\*\_). Dessa forma ignoraremos todos os valores posteriores.

\_, C, \*\_ = nomes\_carrosCOPIAR CÓDIGO

Com isso, recuperaremos somente o valor da primeira posição (ou o segundo item).

CCOPIAR CÓDIGO

Passat

Existe também uma nova função nativa do Python, chamada zip(), que cria um iterador com tuplas (de maneira semelhante à função range()), e vamos entender como ela funciona. Para nossos testes, inicializaremos duas listas: carros, contendo os modelos de alguns veículos; e valores, com seus preços.

carros = ['Jetta Variant', 'Passat', 'Crossfox', 'DS5']

carrosCOPIAR CÓDIGO

valores = [88078.64, 106161.94, 72832.16, 124549.07]

valoresCOPIAR CÓDIGO

Chamaremos então a função zip() passando como parâmetros nossas duas listas.

zip(carros, valores)COPIAR CÓDIGO

Teremos um retorno que significa simplesmente que nosso iterador foi criado com sucesso.

zip at 0x7fb74e671dc8

Com o auxílio da função list(), podemos transformar esse iterador em uma lista de modo a visualizá-lo.

list(zip(carros, valores))COPIAR CÓDIGO

[('Jetta Variant', 88078.64), ('Passat', 106161.94), ('Crossfox', 72832.16), ('DS5', 124549.07)]

Repare que, interessantemente, nosso iterador pegou o primeiro elemento de cada lista e criou uma tupla; depois o segundo elemento de cada lista, criando uma segunda tupla; e assim sucessivamente até o final da lista. Isso nos traz a possibilidade de, por exemplo, iterar o nome e o valor de um veículo ao mesmo tempo.

Faremos isso usando a instrução for que irá iterar pelo resultado de zip(carros, valores). Por enquanto apenas imprimiremos as tuplas resultantes.

for item in zip(carros, valores):

print(item)COPIAR CÓDIGO

('Jetta Variant', 88078.64) ('Passat', 106161.94) ('Crossfox', 72832.16) ('DS5', 124549.07)

Acabamos de aprender a fazer o desempacotamento de tuplas, e podemos fazer isso também com o for. Aqui, ao invés de item, passaremos dois nomes separados por vírgula: carro e valor. Também passaremos a imprimir ambas as variáveis (lembrando que o print() aceita mais de um argumento).

for carro, valor in zip(carros, valores):

print(carro, valor)COPIAR CÓDIGO

Com isso, conseguiremos separar cada carro e valor individualmente em todas as tuplas desse iterador.

Jetta Variant 88078.64 Passat 106161.94 Crossfox 72832.16 DS5 124549.07

Isso nos traz diversas outras possibilidades de análise. Por exemplo, podemos incluir uma condicional if que imprime somente os carros cujo valor é maior do que 100000.

for carro, valor in zip(carros, valores):

if(valor > 100000):

print(carro)COPIAR CÓDIGO

Como resultado, teremos apenas os dois carros que atendem a essa condição.

Passat DS5

No próximo capítulo avançaremos para um novo conteúdo: os dicionários, outro tipo de mapeamento utilizado no Python!