**02 - Instrução for**

Nosso assunto agora são as **estruturas de repetição** e **condicionais** em Python, começando pela instrução **for**. Ela permite que façamos iterações dentro de coleções do Python, que são as listas . Também existem outras formas de criar iteradores para utilizar em estruturas de repetição, como o range que conheceremos mais adiante.

Abaixo temos o formato padrão da estrutura for. Nela temos a instrução for em si seguida de uma variável que representará cada item sendo varrido na coleção, uma instrução in que indica onde queremos iterar e uma coleção - no caso, a nossa lista. Note que a linha termina com dois pontos (:) e que as instruções a serem executadas no laço são indentadas em relação a ela, indicando o seu pertencimento.

for <variável> in <coleção>:

<instruções>COPIAR CÓDIGO

No nosso exemplo continuaremos trabalhando com a lista Acessorios.

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

AcessoriosCOPIAR CÓDIGO

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

Nosso objetivo por enquanto é fazermos uma varredura e acessarmos cada item dessa lista. A ideia também é que seja possível executarmos algum tipo de teste ou condição para esses itens, como o preenchimento de uma lista, mas não trabalharemos nisso especificamente no momento.

Sendo assim, faremos um for e chamaremos a variável a ser utilizada de item. Em seguida, indicaremos que a varredura deverá ocorrer na lista Acessorios usando a palavra reservada in, terminando a linha com dois pontos (:). Pressionando "Enter", o Colab identificará que estamos entrando em um laço for e adicionará a indentação automaticamente. Se estiver usando uma IDE que não faz isso, bastará pressionar "Tab" na linha seguinte.

for item in Acessorios:

COPIAR CÓDIGO

Por enquanto simplesmente imprimiremos cada item da lista passando essa variável para a função intrínseca print().

for item in Acessorios:

print(item)COPIAR CÓDIGO

Com isso, acessaremos cada item da lista e os imprimiremos no console do Colab:

Rodas de liga Travas elétricas Piloto automático Bancos de couro Ar condicionado Sensor de estacionamento Sensor crepuscular Sensor de chuva

No futuro falaremos também de loops alinhados e trabalharemos com as instruções if e else. No momento conversaremos sobre uma construção de listas conhecida como *list comprehensions*, além de conhecermos outra *built-in function* do Python, a range(). As referências para a documentação sobre cada um desses assuntos está no nosso notebook. As *list comprehensions* podem parecer confusas no começo, mas com o tempo você pegará o jeito.

O range() é um método que gera um iterador do tamanho passado por parâmetro. Ou seja, se passarmos o valor 10, geraremos uma "lista" iterável com dez elementos.

range(10)COPIAR CÓDIGO

Ao executarmos, receberemos:

range(0, 10)

Isso significa que temos um *range* que se inicia no índice 0 e tem o tamanho 10 - ou seja, dez elementos. Para analisarmos isso, usaremos outra forma de criar listam em Python, a função list(). Ao passarmos o range(10) como parâmetro dessa função, nos será devolvida uma lista com dez elementos.

list(range(10))COPIAR CÓDIGO

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

É basicamente esse tipo de lista que é iterada quando usamos o range() dentro de um for. Em nosso teste, chamaremos cada item encontrado pelo iterador de i e imprimiremos esse valor ao quadrado com print(i \*\* 2).

for i in range(10):

print(i \*\* 2)COPIAR CÓDIGO

Com isso, faremos a varredura, executaremos a operação e imprimiremos o seu resultado:

0 1 4 9 16 25 36 49 64 81

Preste atenção nesses pontos pois, nos exercícios, começaremos a misturar diferentes conceitos já abordados ao longo do curso - algo que continuará a ser feito no próximo treinamento. Ou seja, devemos entender esses conhecimentos como cumulativos e estudá-los com afinco. Se você tiver alguma dificuldade, pode criar um novo notebook Python e realizar alguns testes ou procurar o fórum do curso para obter auxílio.

Agora queremos armazenar as informações que acabamos de criar dentro de uma lista, algo que é possível por meio do método append(). Para isso, iniciaremos uma lista vazia quadrado e em seguida usaremos o for para iterar em um range(10) chamando cada valor encontrado de i. Dentro do iterador, obteremos o quadrado de i com i \*\* 2 e adicionaremos o resultado à lista quadrado usando o método append(). Por fim, sairemos do *loop* e executaremos somente quadrado para obtermos o seu valor.

quadrado = []

for i in range(10):

quadrado.append(i \*\* 2)

quadradoCOPIAR CÓDIGO

Como retorno, teremos uma lista de todos os valores elevados ao quadrado:

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

Voltando ao assunto das *list comprehensions*, sabemos que elas são uma forma concisa de criar listas no Python. Por exemplo, a lista que acabamos de obter pode ser criada usando uma única linha de código:

[i \*\* 2 for i in range(10)]COPIAR CÓDIGO

Aqui estamos abrindo colchetes e, dentro deles, passando a instrução que deve ser executada - nesse caso, i \*\* 2. Em seguida, usaremos o iterador for indicando o nome da variável (i) e onde desejamos iterar (range(10)). O resultado é uma lista idêntica à obtida anteriormente:

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

Com o tempo esse tipo de estrutura se tornará mais natural, portanto não se preocupe se parecer confuso por enquanto. No próximo vídeo conversaremos sobre *loops* alinhados.