Loops

Vamos a um exemplo, suponha que você tenha que apresentar no console os itens em uma lista de compras de supermercado, um de cada vez, com o que conhecemos até agora uma das alternativas seria assim:

```
lista_compras = ["Arroz", "Feijão", "Carne Seca", "Tomate"]
print(lista_compras[0])
print(lista_compras[1])
print(lista_compras[2])
print(lista_compras[3])
```

Agora imagine que vamos hospedar um banquete com 30 opções diferentes de pratos, a nossa lista teria muito mais do que 4 itens e o trabalho de listar um por um começa a ficar chato e muito trabalhoso, além disso, se fosse necessário adicionar um item da lista a qualquer momento, seria necessário adicionar também mais uma linha de código para lidar com isso.

Para ajudar neste tipo de situação, podemos recorrer as estruturas de repetição presentes em python, neste curso vamos abordar duas, `while` e `for`.
Vamos começar com o while. Para quem tem experiência na língua inglesa já pode inferir que algo será feito **enquanto** alguma condição for verdadeira, e é exatamente isso que acontece. Para utilizar o while, precisamos de uma condição, que pode ser escrita da mesma forma que as expressões utilizadas no comando if, e enquanto o resultado da expressão for verdadeiro, o próximo bloco de código será executado.

```
i = 3
while i > 0:
        print(i)
        i = i - 1

# 3
# 2
# 1
```

O exemplo acima vai exibir uma contagem regressiva que começa em três, depois vai para dois e por último termina em um.

Existem duas palavras reservadas que podem ser utilizadas tanto em loops do tipo while quanto em for, uma serve para interromper o loop por qualquer motivo, enquanto a outra pula uma etapa de execução, passando para a próxima iteração, vamos conhecer as palavras `break` e `continue`.

Vamos ver um exemplo de como usar break para interromper o loop caso uma condição seja atingida.

```
carros = ["Palio", "Uno", "Toro", "Bravo"]
i = 0
while i < len(carros):
        if carros[i] == "Toro":
                break
        print(carros[i])
        i += 1

# "Palio"
# "Uno"
```

Neste exemplo, quando i for igual a 2, a condição if será verdadeira, e então o loop será interrompido pela instrução break.

Ok, sabemos como interromper loops caso uma condição seja verdadeira, mas existem casos onde é preciso apenas pular aquela rodada, e seguir para a próxima iteração, nestes casos, podemos utilizar a palavra continue;

```
carros = ["Palio", "Uno", "Toro", "Bravo"]
i = 0
while i < len(carros):
        if carros[i] == "Toro":
                i += 1
                continue
        print(carros[i])
        i += 1

# "Palio"
# "Uno"
# "Bravo"
```

O for possui uma estrutura muito parecida com o while, mas é muito mais versátil, e por esse motivo é muito mais comum também. Como o próprio nome diz, for executa um bloco de código para cada valor dentro de um iterável, mas o que são iteráveis? Bom, strings e listas são iteráveis que nós já conhecemos.

```
carros = ["Palio", "Uno", "Toro", "Bravo"]
for carro in carros:
        print(carro)
```

Note que é bem mais simples utilizar for do que while, não precisamos utilizar uma variável de índice para acessar os valores da lista, isso ocorre porque no exemplo, a variável carro recebe os valores da lista carros, um de cada vez. Mas se por algum motivo você precisar utilizar o índice, segue aqui um exemplo:

```
for i in range(0, len(carros)):
        print(carro)
```

Um ponto importante, é permitido o uso de break e continue em loops do tipo for, e eles funcionam da mesma forma que em loops do tipo while.

Uma dica legal, é como podemos realizar loops dentro de loops, vamos a um exemplo, no qual a variável carro é uma lista de listas.

```
marcas_carros = [["Palio", "Uno"], ["Golf", "Gol", "Jetta"], ["Celta", "Cruze"]]
for marca in marcas_carros:
        for carro in marca:
                print(carro)
```