100 XP

Sobre os loops "while"

5 minutos

Quando você escreve o código, um desafio comum é fazer com que ele execute uma tarefa um número desconhecido de vezes. Nesta unidade, você deseja permitir que um usuário insira uma lista de nomes de planeta. Infelizmente, você não sabe quantos nomes são inseridos pelo usuário. Para dar suporte a um loop que executa um número desconhecido de vezes, você pode usar o loop `while`.

O loop `while` executa uma operação *enquanto* determinada condição permanecer verdadeira. É possível usar um loop `while` para o seguinte:

- Verificar se há outra linha em um arquivo.
- Verificar se um sinalizador foi definido.
- Verificar se um usuário terminou de inserir valores.
- Verifique se algo mais foi alterado para indicar que o código pode parar de executar a operação.

> **Importante**
>
> A coisa mais importante a ser lembrada ao criar loops `while` é que a condição precisa ser alterada em algum momento. Se a condição for sempre verdadeira, o Python continuará a executar seu código até que o programa falhe.

A sintaxe do loop `while` é semelhante à da instrução `if`. Você fornece uma condição e o código que deseja executar enquanto a condição for verdadeira.

O loop `while` tem três partes importantes:

- A palavra-chave `while`, seguida por um espaço.
- A condição que você testa. Se a condição for verdadeira, o código dentro do loop `while` será executado.
- O código que você deseja executar para cada iteração, recuado com espaço em branco aninhado. Por exemplo:

```
Python
```

```
while <condition>:
    # code here
```

Veja a seguir como criar um código para solicitar que os usuários insiram valores e permitir que eles insiram a opção *Concluído* ao terminarem de inserir os valores desejados. No exemplo, a entrada de usuário é a condição que é testada na parte superior do loop `while`.

Python

```
user_input = ''

while user_input.lower() != 'done':
    user_input = input('Enter a new value, or done when done')
```

Observe que você está usando `input` o para solicitar dados aos usuários. Cada vez que os usuários inserem um novo valor, eles estão alterando a condição, o que significa que o loop `while` será encerrado quando eles inserirem a opção *concluído*.

> **Observação**
>
> Em nosso exemplo, usamos `lower` para converter a entrada em minúsculas, o que permite uma comparação que não diferencia maiúsculas de minúsculas.

Você pode usar a cadeia de caracteres inserida recentemente como faria com qualquer outra cadeia de caracteres capturada com `input`. Se você quiser adicioná-la a uma lista, poderá usar um código semelhante ao seguinte exemplo:

Python

```
# Create the variable for user input
user_input = ''
# Create the list to store the values
inputs = []

# The while loop
while user_input.lower() != 'done':
    # Check if there's a value in user_input
    if user_input:
        # Store the value in the list
        inputs.append(user_input)
    # Prompt for a new value
    user_input = input('Enter a new value, or done when done')
```

Observe a instrução `if` dentro do loop `while`. Esta instrução testa um valor de cadeia de caracteres dentro de `user_input`. Se o loop `while` estiver em execução pela primeira vez, não haverá nenhum valor, portanto, não haverá nada para armazenar em `inputs`. Depois de ser executado pela primeira vez, `user_input` sempre mantém o valor que o usuário inseriu. Como `while` está testando para garantir que o valor não seja igual a `done` (a palavra que o usuário insere para sair do aplicativo), você sabe que o valor atual deve ser adicionado à lista.

> **Observação**
>
> Você pode estar familiarizado com outras linguagens de programação que dão suporte ao loop `do`, que permite executar um teste na parte inferior do loop. O Python não fornece o loop `do`.

Unidade seguinte: Exercício – Criar um loop 'While'

Continuar >