Python - Loops While

Normalmente, o fluxo de execução das etapas de um programa de computador vai do início ao fim. No entanto, em vez da próxima etapa, se o fluxo for redirecionado para qualquer etapa anterior, constitui um loop .

Python enquanto Loop

Uma instrução de loop **while** na linguagem de programação Python executa repetidamente uma instrução de destino, desde que uma determinada expressão booleana seja verdadeira.

Sintaxe do loop while

A sintaxe de um loop **while** na linguagem de programação Python é -

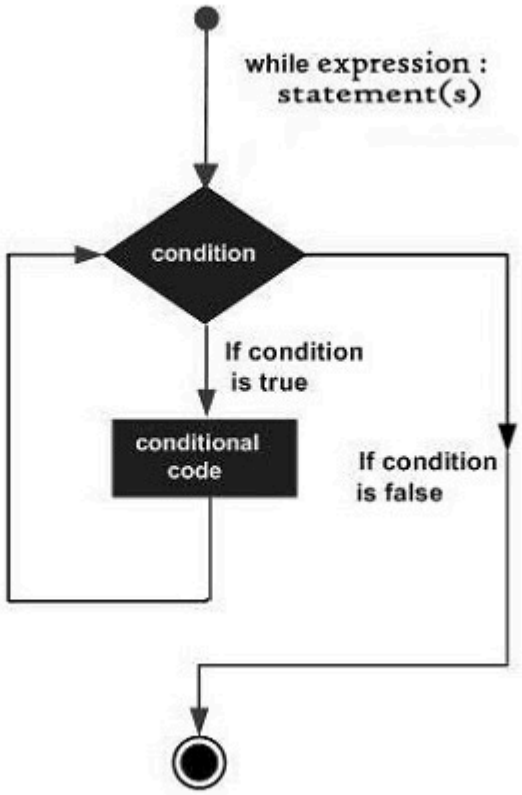
```
while expression:
   statement(s)
```

A palavra-chave **while** é seguida por uma expressão booleana e, em seguida, pelo símbolo de dois pontos, para iniciar um bloco recuado de instruções. Aqui, as declarações podem ser uma única declaração ou um bloco de declarações com recuo uniforme. A condição pode ser qualquer expressão e verdadeiro é qualquer valor diferente de zero. O loop itera enquanto a expressão booleana é verdadeira.

Assim que a expressão se tornar falsa, o controle do programa passa para a linha imediatamente após o loop.

Se não se tornar falso, o loop continuará em execução e não parará, a menos que seja interrompido à força. Esse loop é chamado de loop infinito, o que é indesejado em um programa de computador.

O diagrama de fluxo a seguir ilustra o loop **while** -



Exemplo 1

Em Python , todas as instruções recuadas pelo mesmo número de espaços de caracteres após uma construção de programação são consideradas parte de um único bloco de código. Python usa recuo como método de agrupamento de instruções.

```
count=0
while count<5:
   count+=1
   print ("Iteration no. {}".format(count))

print ("End of while loop")
```

Inicializamos a variável count como 0 e o loop é executado até "count<5". Em cada iteração, a contagem é incrementada e verificada. Se não forem 5, a próxima repetição ocorre. Dentro do bloco de loop, o valor instantâneo da contagem é impresso. Quando a condição **while** se torna falsa, o loop termina e a próxima instrução é executada, aqui está a mensagem de fim do loop **while .**

Saída

Ao ser executado, este código produzirá a seguinte saída -

```
Iteration no. 1
Iteration no. 2
Iteration no. 3
Iteration no. 4
```

```
Iteration no. 5
End of while loop
```

Exemplo 2

Aqui está outro exemplo de uso do loop **while** . Para cada iteração, o programa solicita a entrada do usuário e continua repetindo até que o usuário insira uma string não numérica. A função isnumeric() que retorna verdadeiro se a entrada for um número inteiro, falso caso contrário.

```
var='0'
while var.isnumeric()==True:
   var=input('enter a number..')
   if var.isnumeric()==True:
      print ("Your input", var)
print ("End of while loop")
```

Saída

Ao ser executado, este código produzirá a seguinte saída -

```
enter a number..10
Your input 10
enter a number..100
Your input 100
enter a number..543
Your input 543
enter a number..qwer
End of while loop
```

Python infinito enquanto loop

Um loop se torna um loop infinito se uma condição nunca se tornar FALSA. Você deve ter cuidado ao usar loops while devido à possibilidade de que essa condição nunca seja resolvida para um valor FALSE. Isso resulta em um loop que nunca termina. Esse tipo de loop é chamado de loop infinito.

Um loop infinito pode ser útil na programação cliente/servidor, onde o servidor precisa ser executado continuamente para que os programas clientes possam se comunicar com ele como e quando necessário.

Exemplo

Vamos dar um exemplo para entender como funciona o loop infinito em Python -

```
var = 1
while var == 1 : # This constructs an infinite loop
   num = int(input("Enter a number :"))
   print ("You entered: ", num)
print ("Good bye!")
```

Saída

Ao ser executado, este código produzirá a seguinte saída -

> *O exemplo acima entra em loop infinito e você precisa usar CTRL+C para sair do programa.*

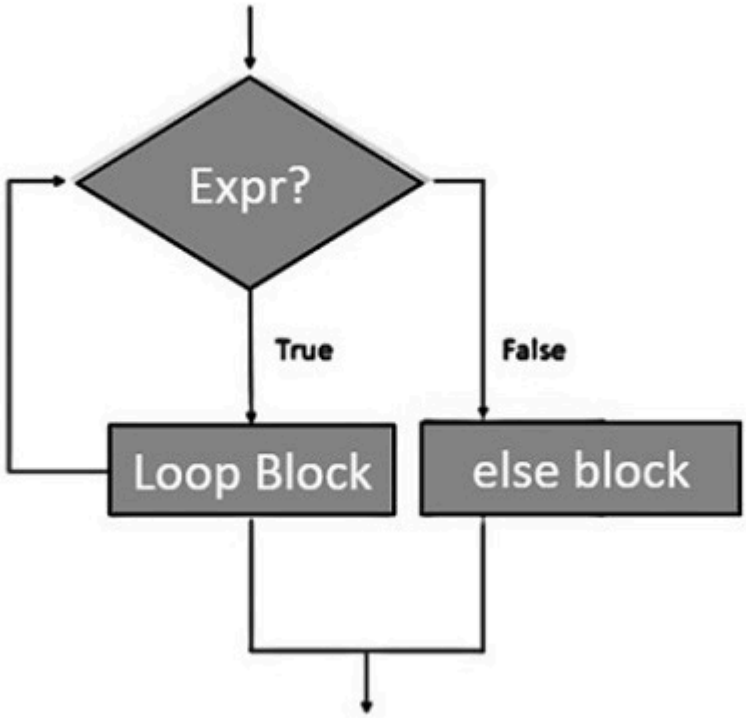
```
Enter a number :20
You entered: 20
Enter a number :29
You entered: 29
Enter a number :3
You entered: 3
Enter a number :11
You entered: 11
Enter a number :22
You entered: 22
Enter a number :Traceback (most recent call last):
  File "examples\test.py", line 5, in
    num = int(input("Enter a number :"))
KeyboardInterrupt
```

Loop while-else do Python

Python suporta ter uma instrução **else** associada a uma instrução de loop **while** .

Se a instrução **else** for usada com um loop **while** , a instrução **else** será executada quando a condição se tornar falsa antes que o controle mude para a linha principal de execução.

O diagrama de fluxo a seguir mostra como usar **else** com a instrução **while** -



Exemplo

O exemplo a seguir ilustra a combinação de uma instrução else com uma instrução while. Até que a contagem seja inferior a 5, a contagem de iterações é impressa. À medida que se torna 5, a instrução print no bloco else é executada, antes que o controle seja passado para a próxima instrução no programa principal.

```
count=0
while count<5:
   count+=1
   print ("Iteration no. {}".format(count))
else:
   print ("While loop over. Now in else block")
print ("End of while loop")
```

Saída

Ao ser executado, este código produzirá a seguinte **saída** -

```
Iteration no. 1
Iteration no. 2
Iteration no. 3
Iteration no. 4
Iteration no. 5
While loop over. Now in else block
End of while loop
```