

ASIMOV

While

A instrução **while** em Python é uma das formas mais gerais de executar iterações. Uma instrução **while** executará repetidamente uma única declaração ou grupo de instruções, desde que a condição seja verdadeira. A razão pela qual é chamado de "loop" é porque as instruções de código são roteadas repetidamente até que a condição não seja mais atendida.

O formato geral de um loop while é:

while teste: declaração de código else: declarações de código final

Olhemos alguns simples do while em ação.

In [1]:

```
x = 0

while x < 10:
    print('x is currently: ', x)
    print(' x is still less than 10, adding 1 to x')
    x += 1
```

```
x is currently: 0
x is still less than 10, adding 1 to x
x is currently: 1
x is still less than 10, adding 1 to x
x is currently: 2
x is still less than 10, adding 1 to x
x is currently: 3
x is still less than 10, adding 1 to x
x is currently: 4
x is still less than 10, adding 1 to x
x is currently: 5
x is still less than 10, adding 1 to x
x is currently: 6
x is still less than 10, adding 1 to x
x is currently: 7
x is still less than 10, adding 1 to x
x is currently: 8
x is still less than 10, adding 1 to x
x is currently: 9
x is still less than 10, adding 1 to x
```

Observe quantas vezes as declarações de impressão ocorreram e como o while continuou até a condição True deixasse de ser verdadeira, que ocorreu após `x == 10`. É importante notar que, uma vez que isso ocorreu, o código parou. Vamos ver como podemos adicionar uma outra afirmação:

In [2]:

```
x = 0

while x < 10:
    print('x is currently: ',x)
```

```

    print(' x is still less than 10, adding 1 to x')
    x+=1

else:
    print('All Done!')

```

```

x is currently: 0
x is still less than 10, adding 1 to x
x is currently: 1
x is still less than 10, adding 1 to x
x is currently: 2
x is still less than 10, adding 1 to x
x is currently: 3
x is still less than 10, adding 1 to x
x is currently: 4
x is still less than 10, adding 1 to x
x is currently: 5
x is still less than 10, adding 1 to x
x is currently: 6
x is still less than 10, adding 1 to x
x is currently: 7
x is still less than 10, adding 1 to x
x is currently: 8
x is still less than 10, adding 1 to x
x is currently: 9
x is still less than 10, adding 1 to x
All Done!

```

break, continue, pass

Podemos usar as declarações `break`, `continue` e `pass` em nossos loops para adicionar funcionalidades adicionais para vários casos. As três declarações são definidas por:

`break`: Para o loop `continue`: Vai para o próximo loop `pass`: Não faz nada

Pensando nas declarações **`break`** e **`continue`**, o formato geral do loop `while` se parece com isto:

`while test: código` `if test: break` `if test: continue` `else:`

Vamos ver alguns exemplos!

In [3]:

```

x = 0

while x < 10:
    print('x is currently: ', x)
    print(' x is still less than 10, adding 1 to x')
    x+=1
    if x ==3:
        print('x==3')
    else:
        print('continuing...')
        continue

```

```

x is currently: 0
x is still less than 10, adding 1 to x
continuing...
x is currently: 1
x is still less than 10, adding 1 to x
continuing...
x is currently: 2
x is still less than 10, adding 1 to x
x==3

```

```

x is currently: 3
  x is still less than 10, adding 1 to x
continuing...
x is currently: 4
  x is still less than 10, adding 1 to x
continuing...
x is currently: 5
  x is still less than 10, adding 1 to x
continuing...
x is currently: 6
  x is still less than 10, adding 1 to x
continuing...
x is currently: 7
  x is still less than 10, adding 1 to x
continuing...
x is currently: 8
  x is still less than 10, adding 1 to x
continuing...
x is currently: 9
  x is still less than 10, adding 1 to x
continuing...

```

Observe como temos uma declaração impressa quando `x == 3` e continuamos imprimindo enquanto continuamos através do `while`. Vamos fazer uma pausa uma vez `x == 3` e ver se o resultado faz sentido:

In [4]:

```

x = 0

while x < 10:
    print('x is currently: ',x)
    print(' x is still less than 10, adding 1 to x')
    x+=1
    if x ==3:
        print('Breaking because x==3')
        break
    else:
        print('continuing...')
        continue

```

```

x is currently: 0
  x is still less than 10, adding 1 to x
continuing...
x is currently: 1
  x is still less than 10, adding 1 to x
continuing...
x is currently: 2
  x is still less than 10, adding 1 to x
Breaking because x==3

```

Observe a declaração `else` não foi alcançada e a continuação nunca foi impressa! Após esses exemplos breves e simples, você deve se sentir confortável ao usar as instruções em seu código.

Uma observação importante! É possível criar um ciclo de execução infinita com instruções `while`. Por exemplo:

In []:

```

# NÃO RODE ESTE CÓDIGO!
while True:
    print('Uh Oh infinite Loop!')

```