

Usos do Laço For

Prof. Alberto Costa Neto
Programação em Python

Usos comuns de laços

1. Encontrar o maior/menor valor entre vários
2. Somar um conjunto de valores
3. Calcular a média de um conjunto de valores
4. Buscar por um valor entre outros valores
5. Filtrar um conjunto de valores

Criando laços “espertos”

- Você precisa **descobrir** ou **calcular** algo sobre os valores contidos em uma sequência.
- Durante a iteração, temos acesso a apenas um item da sequência.
- Geralmente usamos variáveis que **acumulam ou guardam os valores referentes a toda a sequência** e as atualizamos a cada iteração.

Ajustar os valores iniciais das variáveis

for item in sequencia:

Procurar algo ou fazer algo com cada item separadamente, atualizando as variáveis

Consultar as variáveis

Navegando por um conjunto

```
print('Antes')  
for item in [9, 41, 12, 3, 74, 11] :  
    print(item)  
print('Depois')
```

\$ python basicloop.py

Antes

9

41

12

3

74

11

Depois

Encontrando o maior valor

```
maior_ate_agora = -1
print('Antes', maior_ate_agora)
for num in [9, 41, 12, 3, 74, 11] :
    if num > maior_ate_agora :
        maior_ate_agora = num
        print(maior_ate_agora, num)

print('Depois', maior_ate_agora)
```

```
$ python maior.py
```

```
Antes -1
```

```
9 9
```

```
41 41
```

```
41 12
```

```
41 3
```

```
74 74
```

```
74 11
```

```
Depois 74
```

Definimos uma **variável** que contém o **maior valor visto até agora**. Se o valor corrente do **número que está sendo examinado** for maior, ele é o novo **maior valor visto até agora**.

Contando em um Laço

```
cont = 0
print('Antes', cont)
for item in [9, 41, 12, 3, 74, 11] :
    cont = cont + 1
    print(cont, item)
print('Depois', cont)
```

```
$ python lacocontar.py
```

```
Antes 0
```

```
1 9
```

```
2 41
```

```
3 12
```

```
4 3
```

```
5 74
```

```
6 11
```

```
Depois 6
```

Para **contar** quantas vezes o laço foi executado, introduzimos um **contador** que inicia-se em 0 e adicionamos 1 a cada iteração do laço.

Somando em um Laço

```
soma = 0
print('Antes', soma)
for item in [9, 41, 12, 3, 74, 11] :
    soma = soma + item
    print(soma, item)
print('Depois', soma)
```

```
$ python lacosomar.py
Antes 0
9 9
50 41
62 12
65 3
139 74
150 11
Depois 150
```

Para **somar** um **valor** encontrado em um laço, introduzimos uma **variável soma** que **inicia-se em 0** e adicionamos o **valor** à **variável soma** a cada iteração do laço.

Calculando a Média em um Laço

```
cont = 0
soma = 0
print('Antes', cont, soma)
for valor in [9, 41, 12, 3, 74, 11] :
    cont = cont + 1
    soma = soma + valor
    print(cont, soma, valor)
print('Depois', cont, soma, soma / cont)
```

```
$ python media.py
Antes 0 0
1 9 9
2 50 41
3 62 12
4 65 3
5 139 74
6 150 11
Depois 6 150 25.0
```

O cálculo da **média** é uma combinação dos padrões já mostrados para **contar** e **somar**. No final, **dividimos um pelo outro** e temos a média.

Filtrando em um Laço

```
print('Antes')
for valor in [9, 41, 12, 3, 74, 11] :
    print('Testando o valor', valor)
    if valor > 20:
        print(valor, 'maior que 20')
print('Depois')
```

```
$ python filtrar1.py
Antes
Testando o valor 9
Testando o valor 41
41 maior que 20
Testando o valor 12
Testando o valor 3
Testando o valor 74
74 maior que 20
Testando o valor 11
Depois
```

Usamos um comando `if` no `laço` para capturar / filtrar os valores que estamos buscando.

Busca com o auxílio de uma variável

Booleana

```
encontrou = False
print('Antes', encontrou)
for valor in [9, 41, 12, 3, 74, 15] :
    if valor == 3 :
        encontrou = True
    print(encontrou, valor)
print('Depois', encontrou)
```

```
$ python busca1.py
Antes False
False 9
False 41
False 12
True 3
True 74
True 15
Depois True
```

Se queremos buscar para **saber se um valor foi encontrado**, usamos uma **variável** que é inicialmente **False** e recebe **True** quando o valor que está sendo buscado é **encontrado**

Busca com o auxílio de uma variável Booleana (com Break)

```
encontrou = False
print('Antes', encontrou)
for valor in [9, 41, 12, 3, 74, 15] :
    if valor == 3 :
        encontrou = True
        break
    print(encontrou, valor)
print('Depois', encontrou)
```

```
$ python busca1.py
```

```
Antes False
```

```
False 9
```

```
False 41
```

```
False 12
```

```
Depois True
```

Quando o valor foi encontrado, qual é o sentido de continuar procurando?

Podemos usar o comando break para interromper o laço for

Como achar o menor valor?

```
maior_ate_agora = -1
print('Antes', maior_ate_agora)
for num in [9, 41, 12, 3, 74, 11] :
    if num > maior_ate_agora :
        maior_ate_agora = num
    print(maior_ate_agora, num)

print('Depois', maior_ate_agora)
```

\$ python maior.py

Antes -1

9 9

41 41

41 12

41 3

74 74

74 11

Depois 74

O que precisamos fazer para encontrar o menor valor?

Já sabemos encontrar o maior!

Achando o menor valor?

```
menor_ate_agora = -1
print('Antes', menor_ate_agora)
for num in [9, 41, 12, 3, 74, 11] :
    if num < menor_ate_agora :
        menor_ate_agora = num
    print(menor_ate_agora, num)

print('Depois', menor_ate_agora)
```

- 1) Mudamos o nome da variável
 - 2) Trocamos o operador relacional de > para <
- Está pronto?

Achando o menor valor?

```
menor_ate_agora = -1
print('Antes', menor_ate_agora)
for num in [9, 41, 12, 3, 74, 11] :
    if num < menor_ate_agora :
        menor_ate_agora = num
    print(menor_ate_agora, num)

print('Depois', menor_ate_agora)
```

```
$ python menor_bug.py
Antes -1
-1 9
-1 41
-1 12
-1 3
-1 74
-1 11
Depois -1
```

Por que não funcionou?

Achando o menor valor

```
menor_ate_agora = None
print('Antes')
for valor in [9, 41, 12, 3, 74, 11] :
    if menor_ate_agora is None :
        menor_ate_agora = valor
    elif valor < menor_ate_agora :
        menor_ate_agora = valor
    print(menor_ate_agora, valor)
print('Depois', menor_ate_agora)
```

```
$ python menor.py
```

```
Antes
```

```
9 9
```

```
9 41
```

```
9 12
```

```
3 3
```

```
3 74
```

```
3 11
```

```
Depois 3
```

Continuamos com a variável `menor_ate_agora`. Na primeira iteração do laço o valor de `menor_ate_agora` é `None`, logo na primeira iteração o primeiro valor tornar-se-á o `menor_ate_agora`.

Os operadores “is” e “is not”

```
menor_ate_agora = None
print('Antes')
for valor in [9, 41, 12, 3, 74, 11]:
    if menor_ate_agora is None :
        menor_ate_agora = valor
    elif valor < menor_ate_agora :
        menor_ate_agora = valor
    print(menor_ate_agora, valor)
print('Depois', menor_ate_agora)
```

- O operador **is** que pode ser usado em expressões lógicas
- Implica em “é o mesmo que”
- Similar a, porém mais forte que o operador **==** porque testa a identidade dos operandos
- **is not** também é um operador lógico