Conteo en un Bucle

```
zork = 0
print('Antes', zork)
for objeto in [9, 41, 12, 3, 74, 15] :
    zork = zork + 1
    print(zork, objeto)
print('Después', zork)
```

```
$ python countloop.py
Antes 0
1 9
2 41
3 12
4 3
5 74
6 15
Después 6
```

Para contar cuántas veces ejecutamos un bucle, introducimos una variable de conteo que comience en 0 y le sumamos uno cada vez a través del bucle.

Suma en un Bucle

```
zork = 0
print('Antes', zork)
for objeto in [9, 41, 12, 3, 74, 15]
:
    zork = zork + objeto
    print(zork, objeto)
print('Después', zork)
```

```
$ python countloop.py
Antes 0
9 9
50 41
62 12
65 3
139 74
154 15
Después 154
```

Para sumar un valor que encontramos en un bucle, introducimos una variable de suma que comience en 0 y le sumamos el valor a la suma cada vez a través del bucle.

Sacar el Promedio en un Bucle

```
conteo = 0
suma = 0
print('Antes', conteo, suma)
for valor in [9, 41, 12, 3, 74, 15] :
    conteo = conteo + 1
    suma = suma + valor
    print(conteo, suma, valor)
print('Después', conteo, suma, suma /
conteo)
```

```
$ python averageloop.py
Antes 0 0
1 9 9
2 50 41
3 62 12
4 65 3
5 139 74
6 154 15
Después 6 154 25
```

Un promedio solo combina los patrones de conteo (count) y suma (sum) y divide cuando el bucle ha terminado.

Filtrar en un Bucle

```
print('Antes')
for valor in [9, 41, 12, 3, 74, 15] :
    if valor > 20:
        print 'Mayor Número',valor
print('Después')
```

```
$ python search1.py
Antes
Mayor número  41
Mayor número 74
Después
```

Utilizamos un enunciado hipotético “if” en el bucle para captar / filtrar los valores que estamos buscando.

Búsqueda Utilizando una Variable Booleana

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

```
$ python search1.py
Antes False (Falsa)
False (Falsa) 9
False (Falsa) 41
False (Falsa) 12
True (Falsa) 3
True (Falsa) 74
True (Falsa) 15
Después True (Verdadera)
```

Si solo deseamos buscar y saber si un valor fue hallado (found), utilizamos una variable que comience como False (Falsa) y se vuelva True (Verdadera) tan pronto como encontramos (find) lo que estamos buscando.

Cómo Encontrar el Menor Valor

```
mayor_hasta_ahora = -1
print('Antes', mayor_hasta_ahora)
for the_num in [9, 41, 12, 3, 74, 15] :
    if the_num > mayor_hasta_ahora :
 mayor_hasta_ahora = the_num
    print(mayor_hasta_ahora, the_num)

print('Después', mayor_hasta_ahora)
```

```
$ python largest.py
Antes -1
9  9
41  41
41 12
41  3
74  74
74  15
Después 74
```

¿Cómo cambiaríamos esto para hacer que encuentre el menor valor de la lista?

Cómo Encontrar el Menor Valor

```
menor_hasta_ahora = -1
print('Antes', menor_hasta_ahora)
for the_num in [9, 41, 12, 3, 74, 15] :
    if the_num < menor_hasta_ahora :
 menor_hasta_ahora = the_num
    print(menor_hasta_ahora, the_num)

print('Después', menor_hasta_ahora)
```

Cambiamos el nombre de la variable por menor valor hasta ahora (smallest_so_far) y cambiamos > por <

Cómo Encontrar el Menor Valor

```
menor_hasta_ahora = -1
print('Antes', menor_hasta_ahora)
for the_num in [9, 41, 12, 3, 74, 15] :
    if the_num < menor_hasta_ahora :
 menor_hasta_ahora = the_num
    print(menor_hasta_ahora, the_num)

print('Después', menor_hasta_ahora)
```

```
$ python smallbad.py
Antes -1
-1  9
-1  41
-1 12
-1  3
-1  74
-1  15
Después -1
```

Cambiamos el nombre de la variable por menor valor hasta ahora (smallest_so_far) y cambiamos > por <

Cómo Encontrar el Menor Valor

```
menor = Ninguno
print('Antes')
for valor in [9, 41, 12, 3, 74, 15] :
    if menor is Ninguno:
 menor = valor
    elif valor < menor :
 menor = valor
    print(menor, valor)
print('Después', menor)
```

```
$ python smallest.py
Antes
9 9
9 41
9 12
3 3
3 74
3 15
Después 3
```

Aún tenemos una variable que es menor valor (smallest) hasta ahora. La primera vez en el bucle menor valor es Ninguno, entonces tomamos el primer valor como menor valor.

Los Operadores “is” e “is not”

```
menor = Ninguno
print('Antes')
for valor in [3, 41, 12, 9, 74, 15] :
    if menor is Ninguno:
 menor = valor
    elif valor < menor :
 menor = valor
    print menor, valor

print('Después', menor)
```

- Python tiene un operador is (es) que puede ser utilizado en expresiones lógicas
- Implica que “es el mismo que”
- Similar a, pero más fuerte que ==
- is not (no es) también es un operador lógico

Síntesis

- Bucle While (indefinido)
- Bucles infinitos
- Uso de Break
- Uso de Continue
- Bucle For (definido)
- Variables de iteración
- Lenguajes de bucle
- Mayor o menor

Agradecimientos / Colaboraciones

Desarrollo inicial: Charles Severance, Facultad de Información de la Universidad de Michigan

… Ingrese nuevos colaboradores y traductores aquí

...