

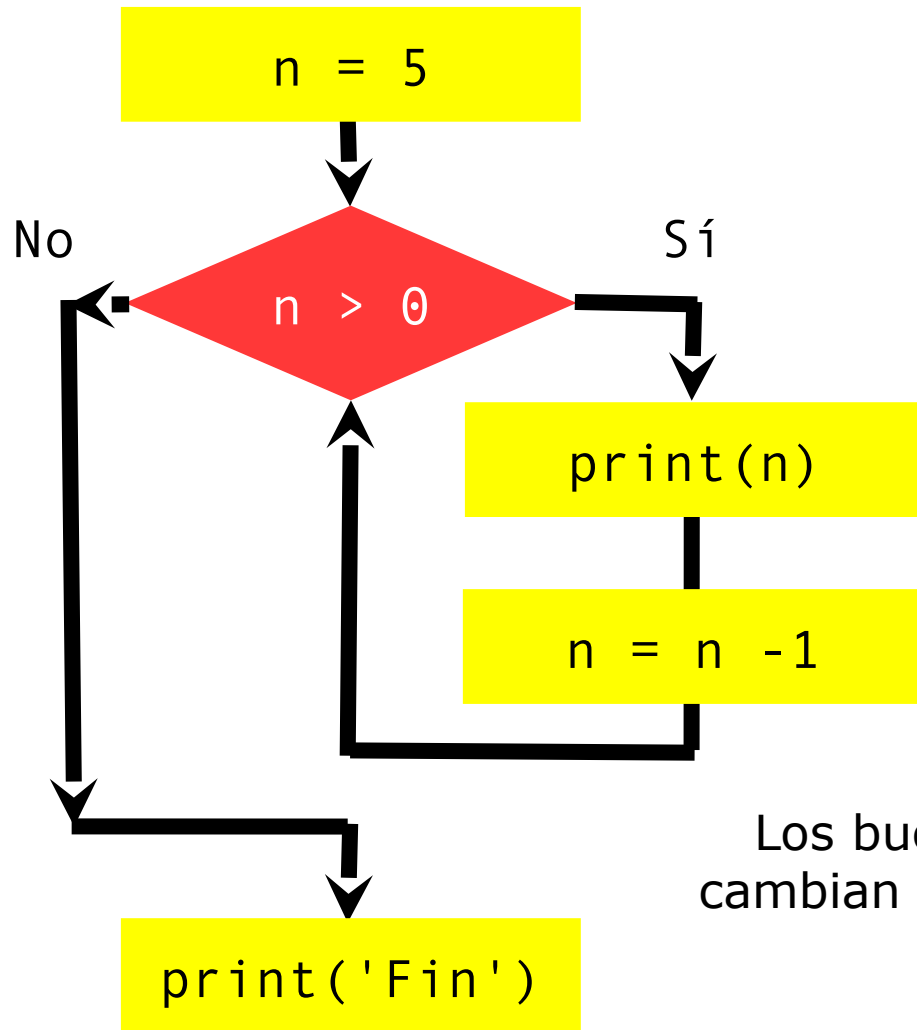
Unidad 3: Estructuras de Control

Instrucciones Iterativas

Iteraciones o Lazos de repetición

- ▶ Iterar se refiere a la repetición de un conjunto de instrucciones.
- ▶ Existen dos instrucciones para implementar iteraciones: `while` y `for`.
- ▶ `while`
 - ▶ Repite un bloque de instrucciones siempre y cuando una condición sea cierta
- ▶ `for`
 - ▶ Repite un bloque de instrucciones un número de veces específico

Lazos de repetición



#Programa

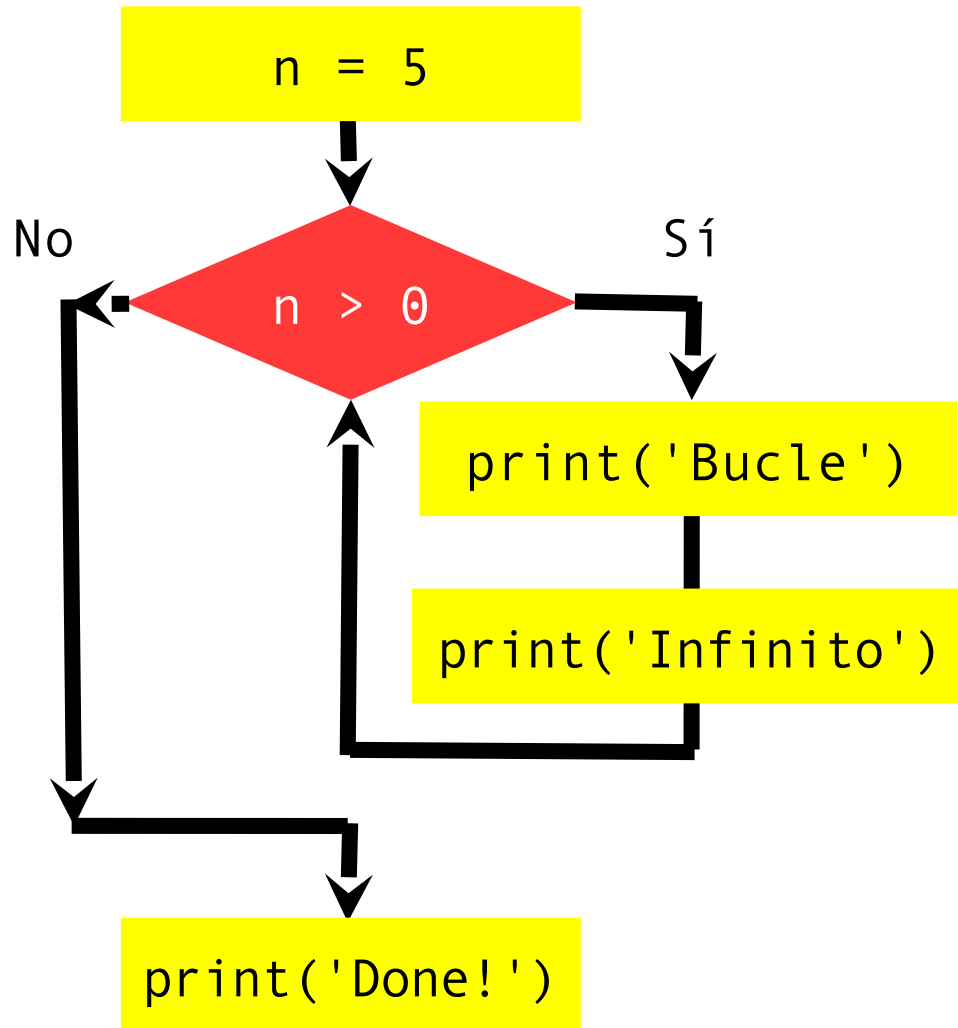
```
n = 5
while n > 0 :
    print(n)
    n = n - 1
print('Fin!')
print(n)
```

Salida

```
5
4
3
2
1
Fin!
0
```

Los bucles (lazos de repetición) tienen **variables de control** que cambian cada vez que se recorre el bucle. A menudo esas variables de control recorren una secuencia de números.

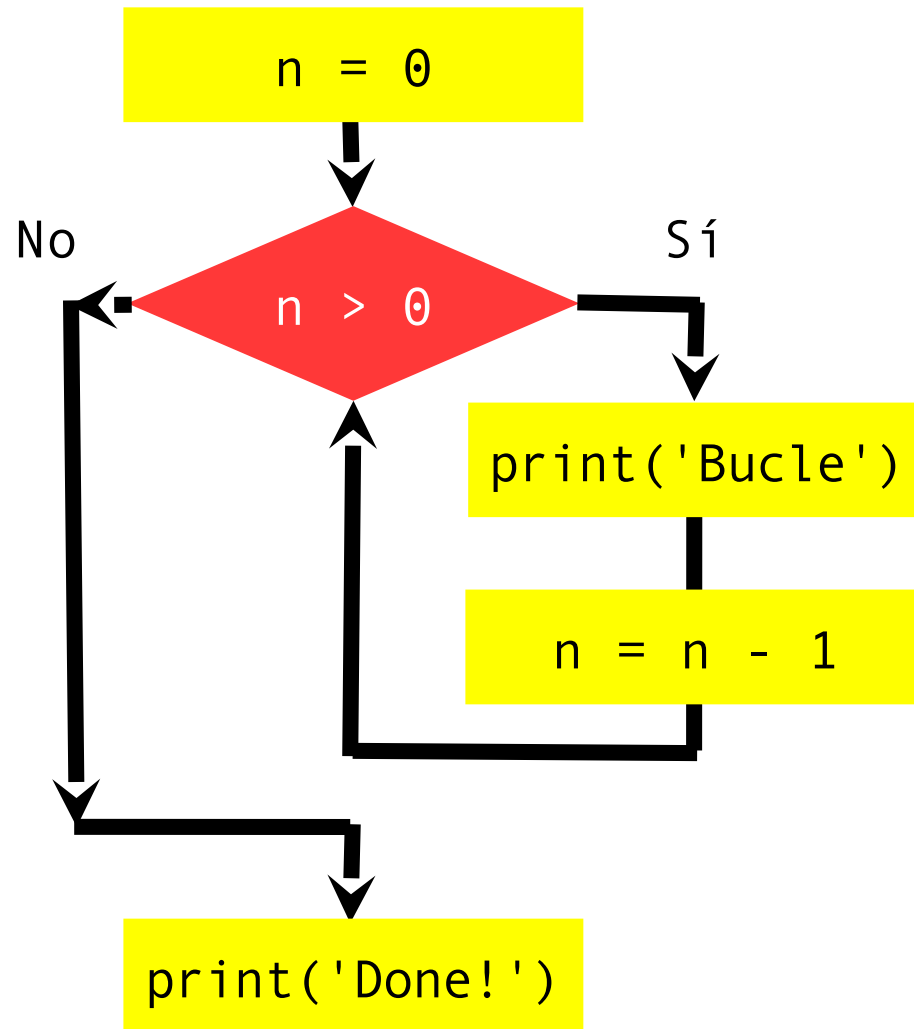
Lazos Infinitos



```
n = 5
while n > 0 :
    print('Bucle')
    print('Infinito')
print ('Done!')
```

¿Qué está mal en este bucle?

Otro bucle




```
n = 0
while n > 0 :
    print('Bucle')
    n = n - 1
print('Done!')
```

¿Qué hace este bucle?

Saliendo de un bucle

- ▶ La instrucción **break** finaliza el bucle actual y salta al siguiente instrucción al bucle
- ▶ Es como una prueba que puede aparecer en cualquier lugar del cuerpo del bucle

```
while True:
    line = input('> ')
    if line == 'done':
        break
    print(line)
print('Fin!')
```

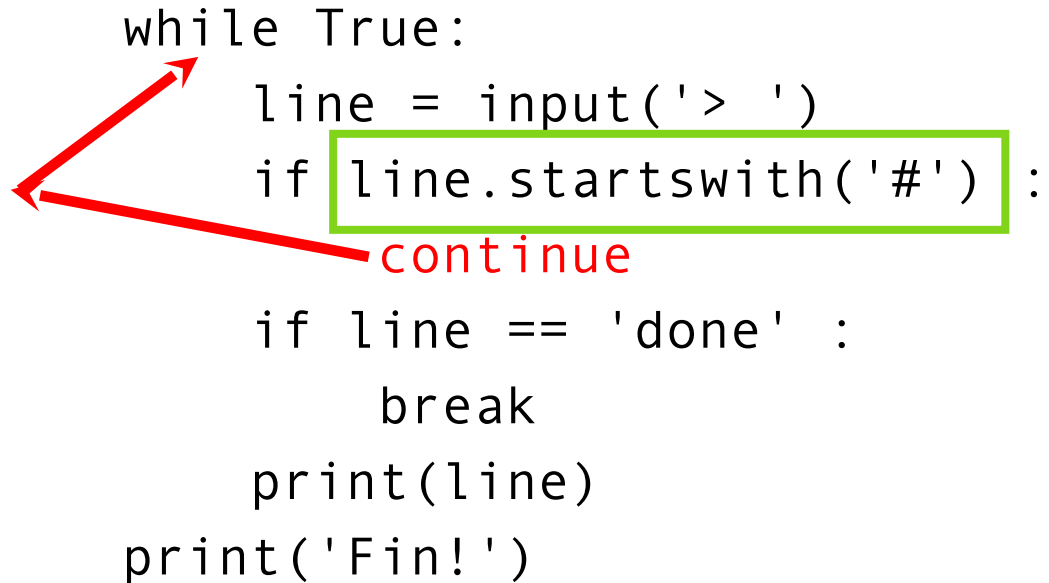


```
> hola aquí
hola aquí
> otro mensaje
otro mensaje
> done
Fin!
```

Terminar una iteración con continue

- La instrucción `continue` finaliza la iteración actual, salta al inicio del bucle y comienza la siguiente iteración

```
while True:
    line = input('> ')
    if line.startswith('#') :
        continue
    if line == 'done' :
        break
    print(line)
print('Fin!')
```



```
> hola a todos
hola a todos
> # no se imprime
> imprimir esto!
imprimir esto!
> done
Fin!
```

Bucles condicionales

- ▶ Los bucles **while** son llamados "bucles condicionales" porque **continúan hasta que una condición lógica se vuelve falsa** (False)
- ▶ En los ciclos que hemos visto hasta ahora es muy fácil determinar si terminan o se vuelven "bucles infinitos"
- ▶ En ocasiones es más difícil determinar si un bucle terminará

Bucles definidos

- ▶ A menudo tenemos una lista de items o las líneas en un archivo - en efecto un conjunto finito de cosas
- ▶ Podemos escribir un bucle que recorra el ciclo una vez por cada uno de los ítems de la lista usando la estructura `for`
- ▶ Estos bucles son llamados “bucles de repetición fija” o “bucles definidos” porque se ejecutan un número exacto de veces
- ▶ Decimos que los bucles definidos `iteran` a través de los elementos de un conjunto

Un bucle definido simple

#Programa

for *i* in *lista* [9, 7, 5, 3, 1] :
 print(*i*)
print()
print('Fin!')

Salida

9
7
5
3
1


Fin!

Un lazo definido con cadenas

#Programa

```
friends = ['Joseph', 'Glenn', 'Sally']  
for friend in friends :  
    print('Happy New Year: ' + friend)  
print()  
print('Done!')
```

Salida



```
Happy New Year: Joseph  
Happy New Year: Glenn  
Happy New Year: Sally  
  
Done!
```

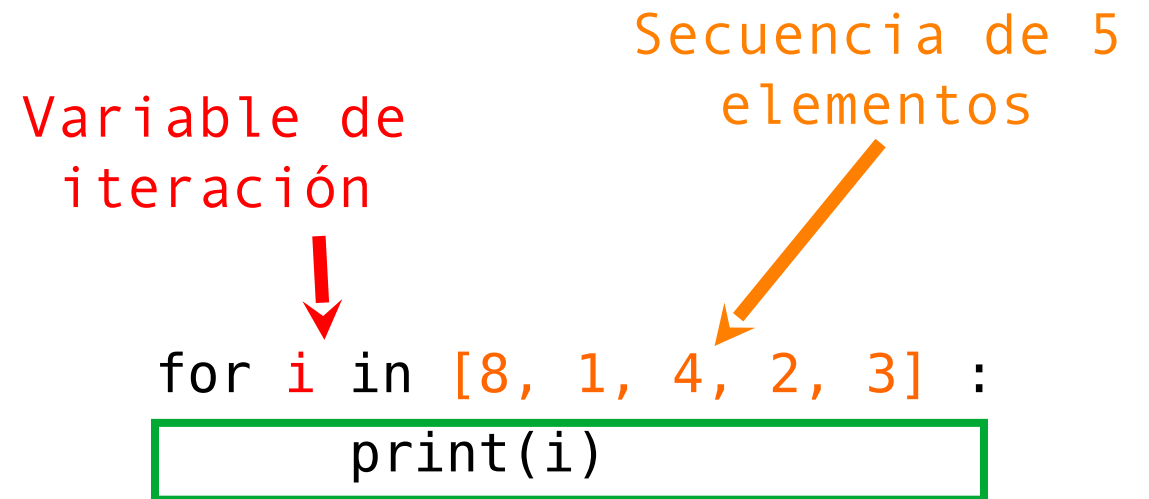
Una mirada a `in`...

- ▶ La variable de iteración "`itera`" a través de la secuencia (conjunto ordenado)
- ▶ El `bloque de instrucciones` es ejecutado una vez por cada valor en la secuencia (`in`)
- ▶ La `variable de iteración` se mueve a través de todos los valores en la secuencia

Variable de iteración

Secuencia de 5 elementos

```
for i in [8, 1, 4, 2, 3] :  
    print(i)
```



Generar una secuencia con `range`...

- ▶ Es posible construir una secuencia de números con la llamada a la función `range()`
- ▶ A través de esta secuencia se puede iterar un bucle definido por los items de la secuencia.
- ▶ `range()` permite generar una secuencia, ascendente o descendente y también no consecutiva

```
for i in range(5) :  
    print(i)
```

```
for i in range(1, 6) :  
    print(i)
```

```
for i in range(1, 11, 2) :  
    print(i)
```

```
for i in range(5, 0, -1) :  
    print(i)
```

Generar secuencias

- La función `range()` devuelve un objeto `range(ini, fin)`, que combinándolo con la función `list()` podemos observar la secuencia generada desde la consola de Python.

```
>>>list(range(5))  
[ 0, 1, 2, 3, 4 ]  
  
>>>list(range(1, 6))  
[ 1, 2, 3, 4, 5 ]  
  
>>>list(range(1, 11, 2))  
[ 1, 3, 5, 7, 9 ]  
  
>>>list(range(5, 0, -1))  
[ 5, 4, 3, 2, 1 ]
```

Costo de un mensaje

```
cont = 1
while True:
    print('linea {}'.format(cont))
    linea = input()
    linea = linea.lower()
    if linea == 'end':
        break

    lista = linea.split(' ')
    costo_linea = 0
    for palabra in lista:
        ...
    print("Costo: {}".format(costo_linea))
    cont = cont + 1

print('Fin')
```

Costo de un mensaje

```
cont = 1
while True:
    print('linea {}'.format(cont))
    linea = input()
    linea = linea.lower()
    if linea == 'end':
        break

    lista = linea.split(' ')
    costo_linea = 0
    for palabra in lista:
        ...
    print("Costo: {}".format(costo_linea))
    cont = cont + 1

print('Fin')
```


Costo de un mensaje

```
cont = 1
while True:
    print('linea {}'.format(cont))
    linea = input()
    linea = linea.lower()
    if linea == 'end':
        break

    lista = linea.split(' ')
    costo_linea = 0
    for palabra in lista:
        ...
    print("Costo: {}".format(costo_linea))
    cont = cont + 1

print('Fin')
```

Costo de un mensaje

```
cont = 1
print('linea {}'.format(cont))
linea = input()
linea = linea.lower()
while linea != 'end' :
    lista = linea.split(' ')
    costo_linea = 0
    for palabra in lista:
        ...
    print("Costo: {}".format(costo_linea))

    print('linea {}'.format(cont))
    linea = input()
    linea = linea.lower()
    cont = cont + 1

print('Fin')
```