Loops

Los loops o ciclos permiten la ejecución de un determinado trozo de código de maneara repetitiva, evitando de esta manera el copiar la misma instrucción varias veces.

For loop

El for loop permite la iteración de elementos, Iteración es la repetición de un segmento de código dentro de un programa,

Sintaxis:

```
for <var> in <iterable>:
    <statement(s)>
```

En este tipo de iterador se necesita inicializar una variable que será la que almacenará el objeto a iterar.

Ejemplo:

```
lista = [1,2,3,4,5,6,7,8,9]
for i in lista:
    print(i)

# 1
# 2
# 3
# 4
# 5
# 6
# 7
# 8
# 9
```

Podemos iterar cualquer tipo de dato de tipo secuencia pero no iterar datos como números enteros o reales ya que estos no puede ser iterados.

La variable i: como se dan cuenta le hemos colocado el nombre de `i` a la variable que contiene el elemento a iterar. Esta variable puede llamarse de cualquier manera pero se suelen utilizar las letras i, j y k.

For range()

La función range nos permite seleccionar un rango de números sobre el cual se ejecutará nuestro ciclo for.

```
for i in range(10):
    print(i, end="-")
```

```
# 0-1-2-3-4-5-6-7-8-9-
```

Continue Continue es una palabra clave que se utiliza para finalizar la iteración actual en un for bucle (o un while bucle) y continúa con la siguiente iteración. Este programa itera una lista en caso de la condición ser verdadera se dejará de ejecutar el ciclo for, en caso de ser falsa se continuará ejecutando el ciclo for.

```
numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

for numero in numeros:
    if numero <= 1:
        print('1 es menor que 2')
        continue
else:
    print('Fin del ciclo for')

# 1 es menor que 2
# Fin del ciclo for
```

Break La palabra reservada break permite detener la ejecución del ciclo. Este programa lo que hace es detenerse si la condición es falsa y ejecutar el condicional else que contiene un mensaje.

```
numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

for numero in numeros:
    if 1 > 2:
        print('1 es mayor que 2')
        break
else:
    print('1 es menor que 2')

# OUTPUT
1 es menor que 2
```

For anidado

Se puede colocar un for dentro de otro for y este funcionará de la siguiente manera. Por cada iteración del primer for se realizarán las n iteraciones del segundo for y así susesivamente, veamos un ejemplo para entenderlo mejor.

```
palabras = ['Python', 'Javascript', 'Flask', 'React', 'Django']
for palabra in palabras:
    for letra in palabra:
        print(letra)

# P
```

```
# y
# t
# h
# o
# n
# J
# a
# v
# a
# s
# c
# r
# i
# p
# t
# F
# l
# a
# s
# k
# R
# e
# a
# c
# t
# D
# j
# a
# n
# g
# o
```

Como vemos por cada palabra que iteramos de la lista de palabras volvemos a iterarla para iterar sobre cada letra de cada palabra de la lista.

While loop

El concepto detrás de un ciclo while es simple: mientras una condición es verdadera -> Ejecuta el bloque de código. El bucle while comprueba la condición cada vez, y si devuelve True, ejecutará las instrucciones dentro del bucle.

```
x = 1

while x <= 10:
    print(x)
    x += 1
```

```
# 1
# 2
# 3
# 4
# 5
# 6
# 7
# 8
# 9
# 10
```

Continue Continue es una palabra clave que se utiliza para finalizar la iteración actual en un for bucle (o un while bucle) y continúa con la siguiente iteración. Este programa itera una lista en caso de la condición ser verdadera se dejará de ejecutar el ciclo for, en caso de ser falsa se continuará ejecutando el ciclo for Aquí hay otro escenario: digamos que desea omitir el bucle si se cumple una determinada condición. Sin embargo, desea continuar con las ejecuciones posteriores hasta que la condición principal while se vuelva falsa. En este programa el bucle se imprimirá de 1 al 10, excepto 5. Cuando x es 5, el resto de los comandos se omiten y el flujo de control vuelve al inicio del bucle while.

```
x = 1

while x <= 10:
    x += 1
    if x == 5:

        continue
    print(x)

# 2
# 3
# 4
# 6
# 7
# 8
# 9
# 10
# 11
```

Break En este programa saldremos del ciclo cuando la condición sea verdadera. En el programa, el bucle detendrá la ejecución cuando x sea 5, a pesar de que x sea mayor o igual que 1.

```
x = 1

while x <= 10:
    if x == 5:
```

```
        break
    print(x)
    x += 1

print("Fin del bucle")

# 1
# 2
# 3
# 4
# Fin del bucle
```

Do while

Un do while es básicamente un while invertido, en vez de comprobar si la condición es verdadera primero y luego ejecutar las instrucciones especificadas lo hace al revés, de manera que primero se ejecuta la acción y luego se comprueba la condición del ciclo. En Python no existe el ciclo do while pero si se puede emular fácilmente.

```
n = 0
while True:
    print(n)
    if n < 10:
        n += 1
    else:
        break
```