Estructuras de Control: if-else

Contenido

- Expresiones Booleanas
- if
- if - else
- if - elif - else
- if + if

Expresiones Booleanas

Una expresión booleana es aquella que al evaluarse devuelve un valor booleano (True-1, False-0).

true y false no son expresiones booleanas

Operadores Relacionales

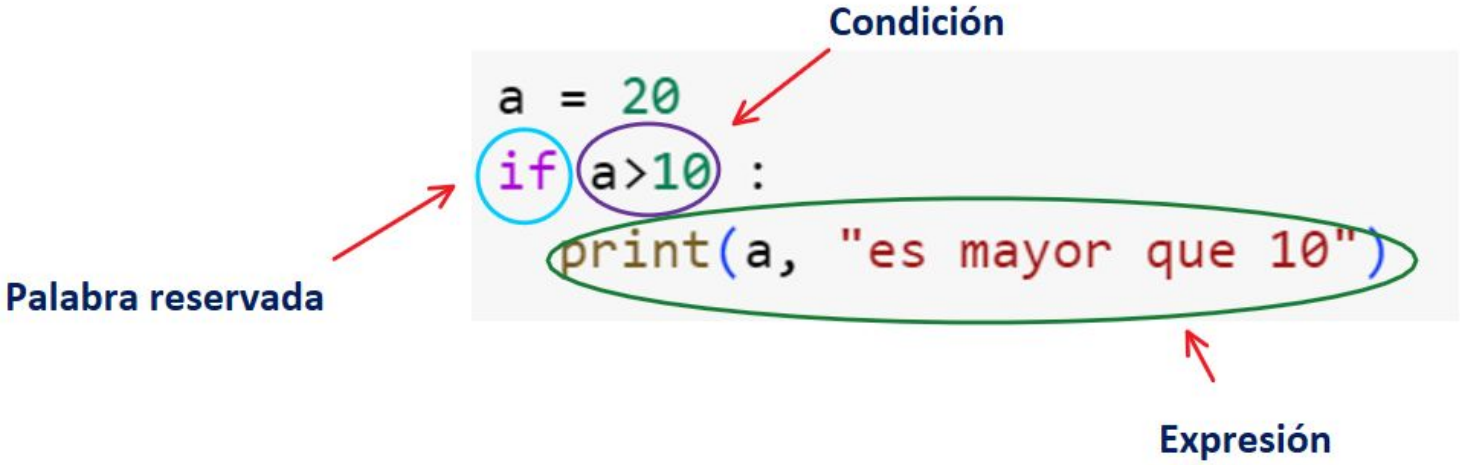
OPERADOR	DESCRIPCIÓN	USO
>	Devuelve True si el operador de la izquierda es mayor que el operador de la derecha	12 > 3 devuelve True
<	Devuelve True si el operador de la derecha es mayor que el operador de la izquierda	12 < 3 devuelve False
==	Devuelve True si ambos operandos son iguales	12 == 3 devuelve False
>=	Devuelve True si el operador de la izquierda es mayor o igual que el operador de la derecha	12 >= 3 devuelve True
<=	Devuelve True si el operador de la derecha es mayor o igual que el operador de la izquierda	12 <= 3 devuelve False
!=	Devuelve True si ambos operandos no son iguales	12 != 3 devuelve True

Operadores lógicos

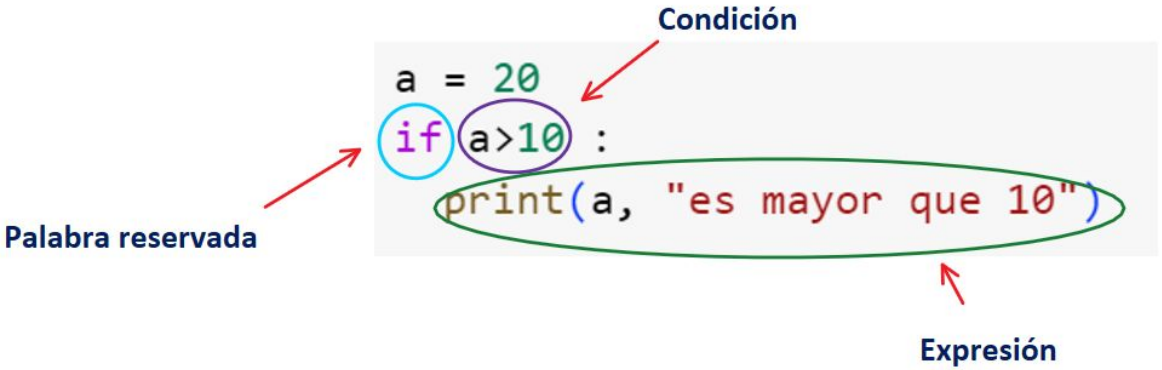
OPERADOR	DESCRIPCIÓN	USO
and	Devuelve True si ambos operandos son True	a and b
or	Devuelve True si alguno de los operandos es True	a or b
not	Devuelve True si alguno de los operandos False	not a

Sentencia Condicional - if

Una **sentencia if** en Python esencialmente dice: "Si la expresión evaluada, resulta ser verdadera (True), entonces ejecuta una vez el código en la expresión, caso contrario (la expresión es falsa), entonces no ejecutes el código

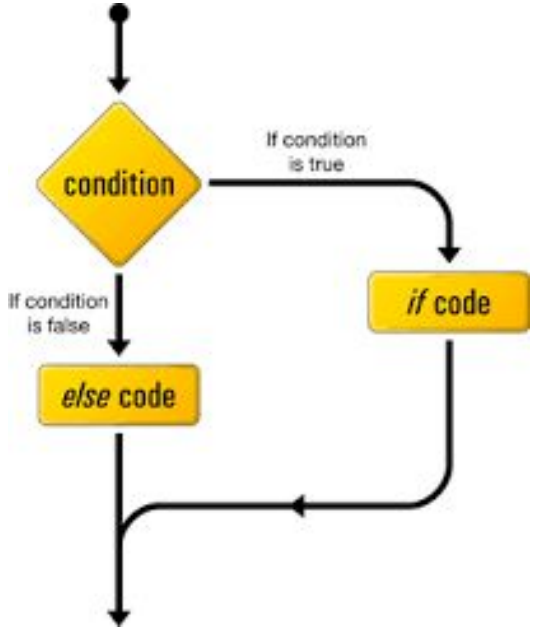


Sentencia Condicional - if



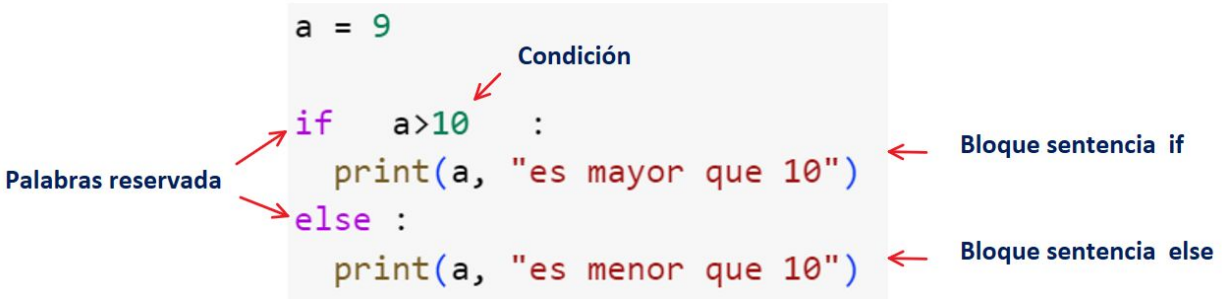
- Palabra reservada : **if**, y termina la línea con **:**
- Expresión booleana: condición (**a>10**)
- Expresión: Puede ser 1 o más líneas
- Indentación

Sentencia Condicional - if else




```
a = 9
if a>10 :
  print(a, "es mayor que 10")
else :
  print(a, "es menor que 10")
```

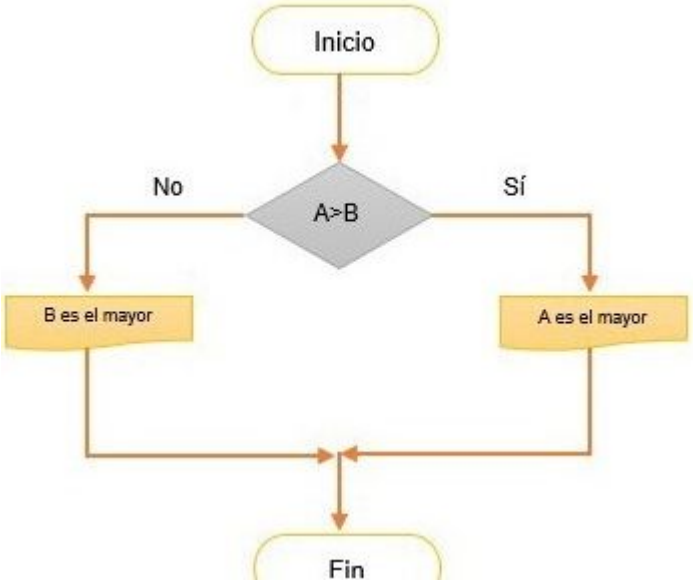
Sentencia Condicional - if else



- La **expresión booleana** es a>10.
- Palabras reservadas **if** y **else** y los dos puntos (**:**) al final.
- Bloque de sentencias del if y del else están indentados.
- Else no requiere una expresión booleana.

Sentencia Condicional - if else

Ejemplo: Se tiene dos números indicar cual es mayor o menor?

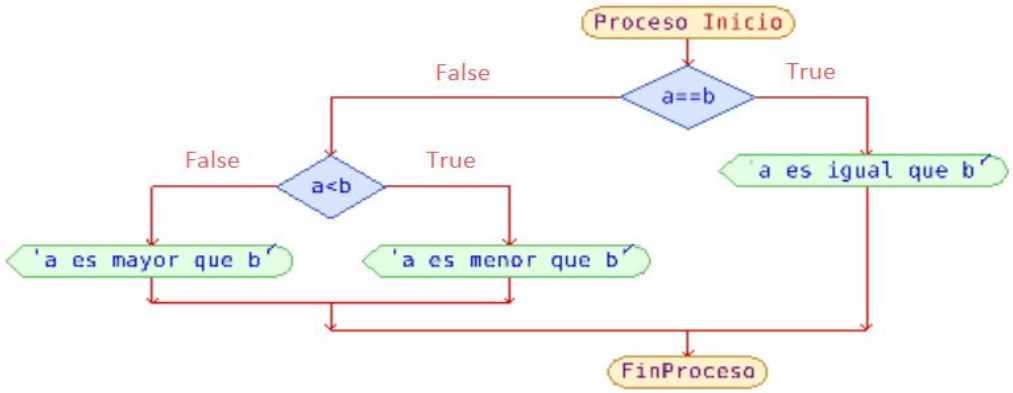



```
1 a = 9
2 b = 14
3 if   a>b   :
4   print(a, "es mayor que ",b)
5 else :
6   print(a, "es menor que ", b)
7
```

```
9 es menor que  14
```

Sentencia Condicional - if elif else

Ejemplo: Se tiene dos números indicar quien es mayor, menor o igual?

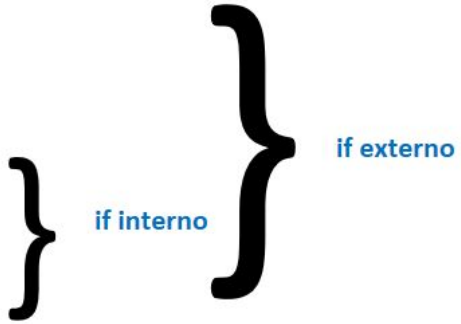



```
1 a = 9
2 b = 14
3 if a==b :
4   print(a, "es igual que ",b)
5 else:
6   if a<b:
7     print(b, "es mayor que ", a)
8   else:
9     print(a, "es mayor que ", b)
```

```
14 es mayor que  9
```

Sentencia Condicional - if elif else

```
1 a = 9
2 b = 14
3 if a==b :
4   print(a, "es igual que ",b)
5 else:
6   if a<b:
7     print(b, "es mayor que ", a)
8   else:
9     print(a, "es mayor que ", b)
```



Sentencia Condicional - if elif else

```
1 a = 9
2 b = 14
3 if a==b :
4   print(a, "es igual que ",b)
5 else:
6   if a<b:
7     print(b, "es mayor que ", a)
8   else:
9     print(a, "es mayor que ", b)
```

```
1 a = 9
2 b = 14
3 if a==b :
4   print(a, "es igual que ",b)
5 elif a<b:
6   print(b, "es mayor que ", a)
7 else:
8   print(a, "es mayor que ", b)
```

- **elif** es una abreviación de else if.
- Tanto a if como a elif tienen expresiones booleanas.

if + if

Ejemplo: Indicar si un número está entre 0 y 10.

```
1 a = 9
2 if a>= 0 :
3   if a<=10 :
4     print(a, "esta entre 0 y 10")
```

```
9 esta entre 0 y 10
```

```
1 a = 9
2 if a>=0 and a<=10 :
3   print(a, "esta entre 0 y 10")
```

```
9 esta entre 0 y 10
```

```
1 a = 9
2 if  0<= a <=10 :
3   print(a, "esta entre 0 y 10")
```

```
9 esta entre 0 y 10
```

Contenido del módulo

- **Introducción a Python**
- Variables y operaciones
- Listas y cadenas
- Estructuras del control
- Funciones y módulos

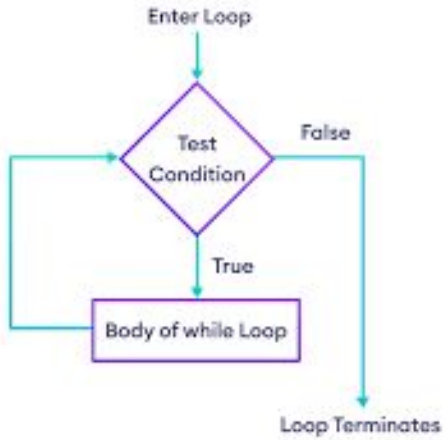
While

Contenido

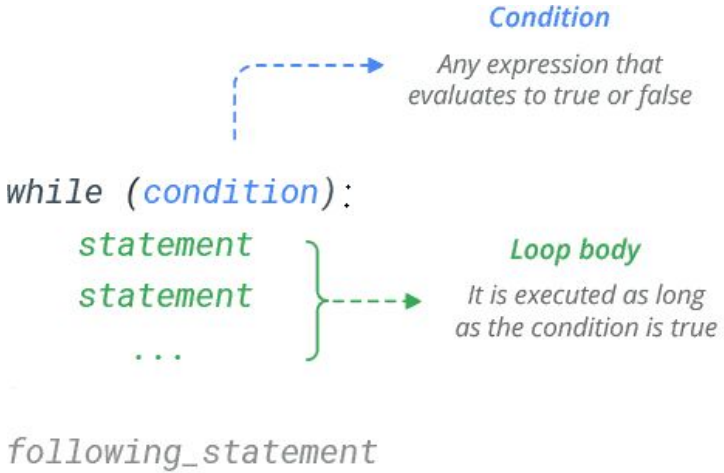
- Concepto
- Ejemplos
- While con listas
- While + …
- Creación de tablas
- While anidado
- Break y Continue

Concepto

- El **bucle while** es otra estructura de control de flujo, concretamente lo que hace es **repetir** un código mientras dure una determinada condición.
- Al entrar en while, se evalúa la condición, si es verdadera el contenido del bucle se ejecutará, pasada la ejecución volverá a comprobar la condición, si es verdad vuelve a entrar en el bucle (repetirá este ciclo), cuando la condición deje de cumplirse (falsa) sale del while y continúa su ejecución.



Concepto




```
1 a = 2
2 while (a<5):
3   print("Estoy en el while")
4   a +=1
5 print("Sali del while")
```

Ejemplos

Ejemplo: Escribir 5 veces Ciencia de la Computación

```
1 a = 0
2 while (a<5):
3   print("Ciencia de la computación")
4   a +=1
5
```

```
Ciencia de la computación
Ciencia de la computación
Ciencia de la computación
Ciencia de la computación
Ciencia de la computación
```

Ejemplo

Ejemplo: Imprimir números del 1 al 9.

```
1 a = 1
2 while (a<10):
3   print(a)
4   a +=1
5
```

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

Ejemplo

Ejemplo: Escribir números de 5 al 1 y finalmente Despegue!

```
1 n=5
2 while n > 0:
3     print (n)
4     n = n-1
5 print ("¡Despegue!")
6
```

```
5
4
3
2
1
¡Despegue!
```

While con listas

El bucle while nos permite recorrer una lista mientras se cumpla una condición específica.

```
1 dia = 0
2 semana = ['Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes', 'Sabado', 'Domingo']
3 while dia < 7:
4     print("Hoy es " + semana[dia])
5     dia += 1
```

```
Hoy es Lunes
Hoy es Martes
Hoy es Miércoles
Hoy es Jueves
Hoy es Viernes
Hoy es Sabado
Hoy es Domingo
```

While con listas

```
1 frutas = ["manzana", "banana", "naranja", "uva"]
2 indice = 0
3 while indice < len(frutas):
4     print(frutas[indice])
5     indice += 1
```

```
manzana
banana
naranja
uva
```

While + …

En algunos casos esto no es tan fácil de asegurar el número de veces que se ejecutará el while

```
1 n=10
2 while n != 1:
3     print(n)
4     if n%2 == 0:
5         n = n/2
6     else:
7         n = n*3+1
8
```

```
10
5.0
16.0
8.0
4.0
2.0
```

Creación de tablas

Ejemplo: Crear la tabla de multiplicar del 2

```
1 n = 1
2 while n <= 10:
3     print('2 x',n,'=',2*n)
4     n = n + 1
5
```

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

Creación de tablas

Ejemplo: Crear una tabla con los primeros 10 cuadrados

```
1 import math
2 n = 1
3 while n <= 10:
4     print (n, '\t', 'elevado al 2 \t=\t', math.pow(n,2))
5     n = n + 1
6
```

```
1       elevado al 2    =       1.0
2       elevado al 2    =       4.0
3       elevado al 2    =       9.0
4       elevado al 2    =       16.0
5       elevado al 2    =       25.0
6       elevado al 2    =       36.0
7       elevado al 2    =       49.0
8       elevado al 2    =       64.0
9       elevado al 2    =       81.0
10      elevado al 2    =       100.0
```

While anidado

Ejemplo: Imprimir los primeros 5 multiplos de los 5 primeros números.

```
1 i = 1
2 while i <= 5:
3     print (i*1, end='\t')
4
5 prir
6
```

```
1 i = 1
2 while i <= 5:
3     print (i*2, end='\t')
4
5 print
6
```

```
1 i = 1
2 while i <= 5:
3     print (i*3, end='\t')
4     i = i + 1
5 print()
6
```

1

2

3 6 9 12 15

While anidado

Ejemplo: Imprimir los primeros 5 multiplos de los 5 primeros números.

```
1 a = 1
2 while a <= 5:
3     i = 1
4     while i <= 5:
5         print (i*a, end='\t')
6         i = i + 1
7     print()
8     a = a + 1
9
```

```
1       2       3       4       5
2       4       6       8       10
3       6       9       12      15
4       8       12      16      20
5       10      15      20      25
```

Break y Continue

- La sentencia **continue** se usa para omitir la ejecución del resto del código dentro de un bucle solo para la iteración actual. El bucle no termina pero continúa con la siguiente iteración.

```
1 i = 0
2 while i <= 5:
3     i +=1
4     print(i)
5     if i == 3:
6         continue
7     print(i)
8
```

```
1
1
2
2
3
4
4
5
5
6
6
```

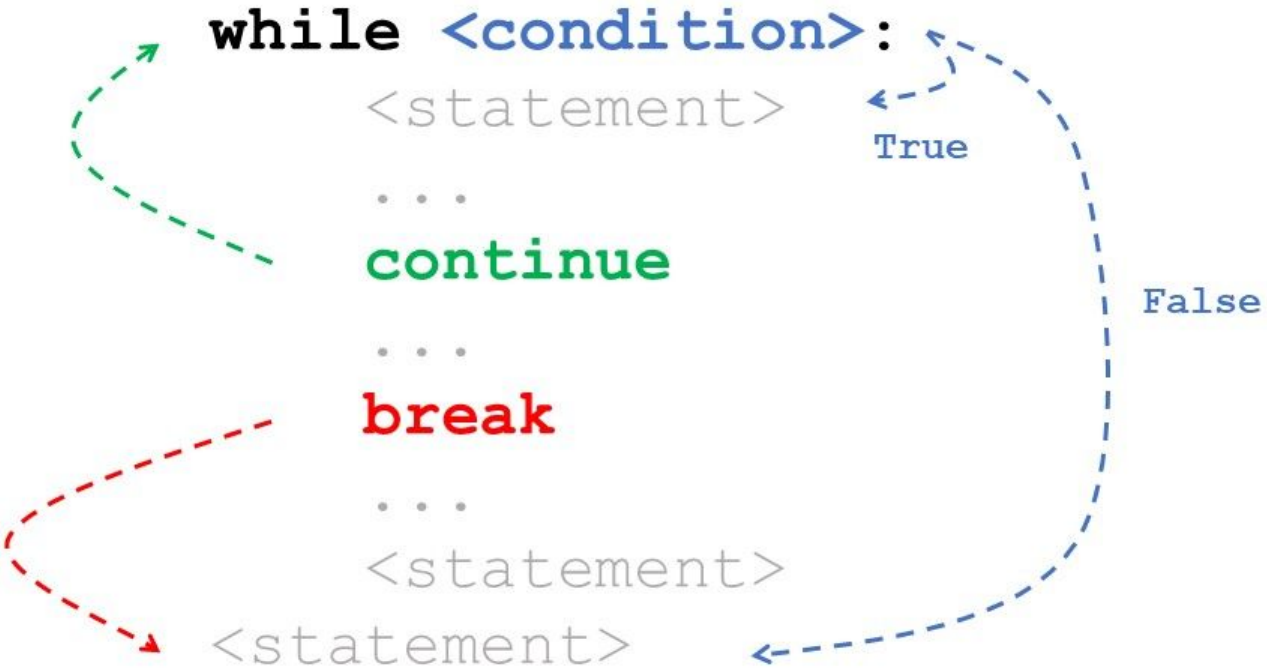
Break y Continue

- La sentencia **break** termina el ciclo que lo contiene. El control del programa fluye a la declaración inmediatamente después del cuerpo del bucle.

```
1 i = 0
2 while i <= 5:
3     i +=1
4     print(i)
5     if i == 3:
6         break
7     print(i)
8 print("Sali del while")
9
```

```
1
1
2
2
3
Sali del while
```

Break y Continue



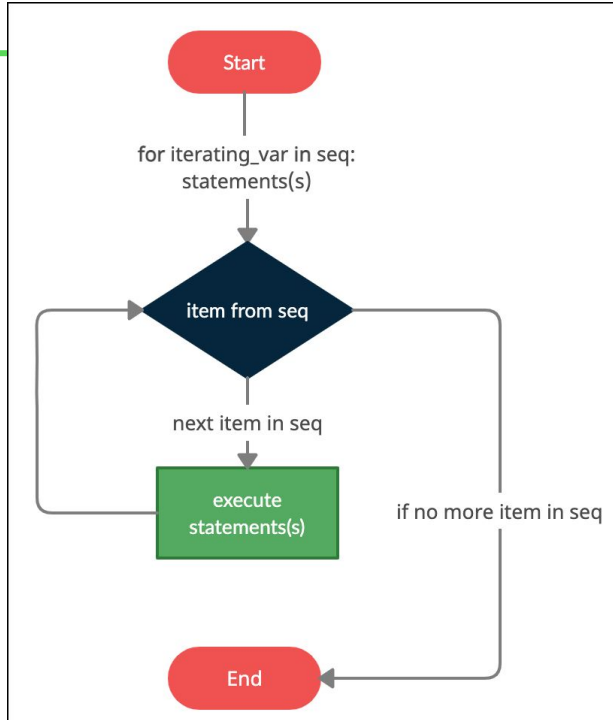
FOR

Contenido

- Concepto
- Ejemplos
- Función range
- For anidado

Concepto

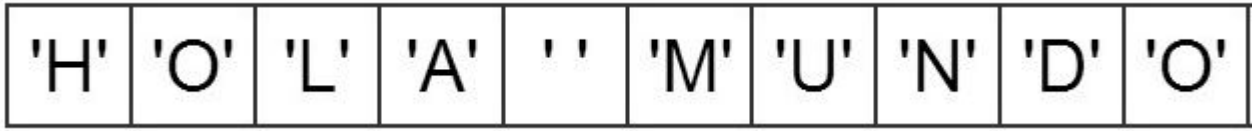
Un bucle **for** establece la variable iteradora en cada valor de una lista, arreglo o cadena proporcionada y repite el código en el cuerpo del bucle for para cada valor de la variable iteradora.



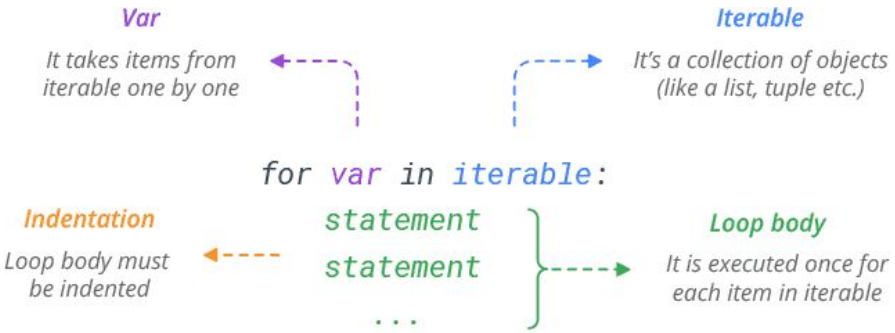
Concepto

Los **iteradores** son objetos que hacen referencia a un elemento, y que tienen un método next que permite hacer referencia al siguiente.

Los **iterables** son aquellos objetos que como su nombre indica pueden ser iterados. Por ejemplo: listas, tuplas, cadenas o diccionarios



Concepto




```
1 frutas = ["manzana", "banana", "naranja", "uva"]
2 for var in frutas:
3   print(var)
```

```
manzana
banana
naranja
uva
```

Ejemplo

Ejemplo: Recorrer una lista de números e imprimir el doble de cada elemento.

```
1 valores = [1, 2, 3, 4, 5, 6]
2 for val in valores:
3   print(val*2, end='\t')
```

```
2	4	6	8	10	12
```

Ejemplo

Ejemplo: Invertir la palabra Computacion.

```
1 texto = "Computacion"
2 for letra in texto[::-1]:
3     print(letra, end="")
```

```
noicatupmoC
```

range()

La función range() proporciona una secuencia de enteros basada en los argumentos de la función.

range (fin)
range(inicio, fin)
range(inicio, fin , paso)

range() - Ejemplo

1

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

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
```

2

```
for i in range(5, 10):
    print(i, end=", ")
```

```
5, 6, 7, 8, 9,
```

3

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

```
1, 3, 5, 7, 9,
```

4

```
for i in range(10,0,-2):
    print(i, end=", ")
```

```
10, 8, 6, 4, 2,
```

for anidado

- Es posible **anidar los for**, es decir, un for dentro de otro.
- Es muy útil si queremos iterar algún objeto que en cada elemento, tiene a su vez otra clase iterable.
- Podemos tener por ejemplo, una lista de listas, una especie de matriz.

for anidado - ejemplo

Ejemplo: Imprimir los elementos de una matriz.

```
1 matriz = [[10, 20, 40],
2           [40, 50, 60],
3           [70, 80, 90]]
4 for i in matriz:
5   print(i)
```

```
[10, 20, 40]
[40, 50, 60]
[70, 80, 90]
```

for anidado - ejemplo

Ejemplo: Imprimir los elementos de una matriz.

```
1 matriz = [[10, 20, 40],
2           [40, 50, 60],
3           [70, 80, 90]]
4 for i in matriz:
5   print(i)
```

```
[10, 20, 40]
[40, 50, 60]
[70, 80, 90]
```

```
1 for i in matriz:
2   for j in i:
3    print(j)
```

```
10
20
40
40
50
60
70
80
90
```

Gracias!