Bucles en Python:

Los bucles permiten que varios elementos combinados en una misma "colección" sean procesados individualmente según un mismo esquema. Nos permiten ejecutar un bloque de código varias veces seguidas de forma automática. Esto se traduce en un código más eficiente, compacto y flexible.

En Python existen dos tipos principales de bucles: *for* y *while*.

Bucle *for*

El bucle *for* se utiliza para recorrer los elementos de un objeto iterable (lista, tupla, conjunto, diccionario, ...) y ejecutar un bloque de código, lo que se denomina "iteración". En cada paso de la iteración se tiene en cuenta a un único elemento del objeto iterable, sobre el cuál se pueden aplicar una serie de operaciones. Un *iterable* es un objeto que permite recorrer sus elementos uno a uno. Un *iterador* es un objeto que define un mecanismo para recorrer los elementos del iterable asociado.

La sintaxis básica del bucle *for* es la siguiente:

```
for <elemento> in <objeto iterable>:
    <código>
```

El bucle finaliza su ejecución cuando se recorren todos los elementos del objeto iterable. Por ejemplo:

Ejemplo 1

```
numeros = [1, 2, 3, 4, 5]

for num in numeros: # para (for) cada elemento (num) en el objeto iterable (la lista numeros)
    print(num) #acción a realizar. Se imprimirá cada número en la lista.
```

También podemos incluir una lógica más compleja en el cuerpo de un bucle *for*

Ejemplo 2

```
numeros = [1, 2, 3, 4, 5]

for num in numeros: # para (for) cada elemento (num) en el objeto iterable (la lista numeros)
    calc = num**2 - (num-1)*(num+1) # cálculo
    print(calc) #acción
```

Bucle *for* en diccionarios

Un diccionario está compuesto por pares clave/valor, por lo que hay distintas formas de iterar sobre ellas.

1 – Recorrer las claves del diccionario.

Ejemplo 3

```
capitales_de_paises = {"España": "Madrid", "Francia": "Paris", "Italia": "Roma"}

for pais in capitales_de_paises:
    print(pais)
```

2 - Iterar sobre los valores del diccionario.

Ejemplo 4

```
capitales_de_paises = {"España": "Madrid", "Francia": "Paris", "Italia": "Roma"}

for capital in capitales_de_paises.values():
        print(capital)
```

3 – Iterar a la vez sobre la clave y el valor de cada uno de los elementos del diccionario.

Ejemplo 5

```
capitales_de_paises = {"España": "Madrid", "Francia": "Paris", "Italia": "Roma"}

for pais, capital in capitales_de_paises.items():
        print(pais+":", capital)
```

Python *for* y la clase range

Para los casos en los que se quiera realizar un bucle *for* en una secuencia numérica, se puede utilizar la clase *range*.

El tipo de datos *range* se puede invocar con uno, dos e incluso tres parámetros:

- range (max): Un iterable de números enteros consecutivos que empieza en *0* y acaba en el valor máximo menos uno (*max-1*)

Ejemplo 6

```
for num in range(10):
        print (num) # imprimirá los números desde el 0 al 9.
```

- range(min, max): Un iterable de números enteros consecutivos que empieza en el valor mínimo (*min*) y acaba en el valor máximo menos uno (*max-1*)

Ejemplo 7

```
for num in range(4,10):
        print (num) # imprimirá los números desde el 4 al 9.
```

- range(min, max, step): Un iterable de números enteros consecutivos que empieza el valor mínimo (*min*), acaba en el valor máximo menos uno (*max-1*) y los valores se van incrementando paso a paso (*step by step)*. Este último caso simula el bucle *for* con variable de control.

Ejemplo 8

```
for num in range(1,10, 2):
        print (num) # imprimirá los números del 1 al 9, cada dos números (es decir, sólo los números impares)
```

Bucle *for* con *break* y *continue*

La iteración del bucle *for* se puede modificar mediante el uso de las sentencias *break* y *continue*.

- *break* se utiliza para finalizar y salir del bucle si se cumple la condición.

Ejemplo 9

```
nombre = "Enrique"
lista_nombre = ["Jessica", "Paul", "George", "Henry", "Adán"]

for item in lista_nombre:
    if item == nombre: # condición
        print(f"{nombre} está en la lista.") # acción si condición es True
        break # si condición es True el bucle se termina y se sale de él.
    else:
        print(f"Buenos días {item}.") # acción si condición es False
```

- *continue* se utiliza para saltar al siguiente paso de la iteración, ignorando todas las sentencias que le siguen y que forman parte del bucle.

Ejemplo 10

```
nombre = "Enrique"
lista_nombre = ["Jessica", "Paul", "George", "Henry", "Adán"]

for item in lista_nombre:
    if item == nombre: # condición
        print(f"{nombre} está en la lista.") # acción si condición es True
        continue # el bucle pasa al siguiente elemento.
    else:
        print(f"Buenos días {item}.") # acción si condición es False
```

Bucle *while*

El bucle *while* se utiliza para ejecutar un bloque de código tantas veces como la condición específica sea verdadera. Utilizamos un bucle *while* cuando el tamaño de la colección no puede determinarse en el momento de ejecutar el programa.

La estructura de esta sentencia *while* es la siguiente:

```
while <condición>:
    <código>
```

Ejemplo 11

```
numeros = [1, 2, 3, 4, 5]
contador = 0

while contador < 5: # la condición que se evalúa en cada iteración del bucle. Mientras la condición sea True, la acción o acciones se realizarán.
    print(contador) # acción
    contador += 1  # acción
```

Este bucle *while* imprimirá los números del 0 al 4, ya que incrementamos el contador en cada iteración. El bucle continuará ejecutándose mientras la condición (contador < 5) sea verdadera (*True*). Una vez que la condición se vuelva falsa (False) el bucle se detendrá.

Bucle while con *break* / *continue* y *else*

El bucle *while* también puede ser alterado con las sentencias *break* y *continue*.

Ejemplo 12

```
lista_nombre = ["Jessica", "Paul", "George", "Henry", "Adam"]

contador = 0

while contador < len(lista_nombre): # primera condición
        nombre = lista_nombre[contador]
        if nombre == "Enrique":  # segunda condición
                print(f'{nombre} ha sido encontrado en la posición número {contador+1}')
                break # el bucle se termina si esta segunda condición es True
         else: # en caso de que esta segunda condición sea False
                contador += 1
else: # en caso de que la primera condición sea False
        print(f"Enrique no se encuentra en la lista.")
```

Por otro lado, como el ejemplo muestra, al bucle *while* le podemos añadir la sentencia opcional *else*. El bloque de código bajo esta sentencia se ejecutará siempre y cuando la condición de la sentencia *while* sea *False* y no se haya ejecutado una sentencia *break*.