Introducción

Se denominan ciclos repetitivos o iterativos a aquellos que agrupan un bloque de instrucciones para ser ejecutado un determinado número de veces. Este número puede ser fijo (definido por el programador) o puede ser variable (dependiendo de algún valor o condición en el programa).

Se puede ver un ejemplo de un ciclo iterativo en el siguiente gráfico:

Mientras la condición sea Verdadera, se realizará el bloque de instrucciones, hasta que la condición sea falsa.

Para Marzal(2014), los ciclos iterativos en Python, usan dos instrucciones: el ciclo ***FOR*** y el ciclo ***WHILE***. Dependiendo de la situación que se deba resolver, se selecciona la instrucción mas adecuada a utilizar.

1. Ciclo FOR

La estructura ***for*** se usa en aquellos casos en los cuales se tiene certeza de la cantidad de veces que se desea ejecutar un bloque de instrucciones.

Ejemplo: leer 6 edades de aprendices, ingresar 100 salarios de empleados.

El bloque de instrucciones que se repite se denomina cuerpo del bucle y cada repetición se denomina una iteración.

Esta estructura de control iterativa se utiliza en situaciones en las que se desea que una variable de control tenga un valor inicial y un valor final predefinidos. No es necesario definir la variable de control antes del bucle, aunque se puede utilizar una variable ya definida previamente en el programa.

El bucle ***for*** se puede usar para recorrer los items de cualquier secuencia (cadena, lista, tupla, conjunto, diccionario) y ejecutar un bloque de código sobre ese item. En cada iteración se tiene acceso a un único elemento de la secuencia.

```
Ejemplo:

animales = ['leon', 'tigre', 'cocodrilo']

for animal in animales:

print ("El animal es: {0}, tamaño de palabra es: {1}".format(animal, len(animal)) )
```

```
El animall es: leon , tamaño de palabra es: 4

El animall es: tigre , tamaño de palabra es: 5

El animall es: cocodrilo , tamaño de palabra es: 9
```

La estructura del ciclo for es la siguiente:

1.1 Uso de RANGE

Según Pérez(2016) en el ciclo iterativo ***for*** se puede usar la función range, para indicar un rango de valores.

```
range(valor_inicial, valor_final)
```

Es posible indicar solo un valor final, en caso tal, el ciclo inicia en 0.

```
Ejemplo

for i in range(5):

print(i)

print('fin del ciclo')
```

```
0

1

2

3

4

fin del ciclo
```

Si se especifica el valor inicial y el valor final en la función range, el resultado imprimirá desde el valor inicial hasta el valor final - 1, así:

```
Ejemplo

for i in range(2, 5):

print(i, i ** 3)
```

```
print('fin del ciclo')
```

```
2 8

3 27

4 64

fin del ciclo
```

Para imprimir hasta el valor final, éste debe incrementarse en 1.

1.2 Conceptos: valor inicial, valor final, incremento.

Cuando se deseen especificar los valores inicial, final y el incremento o decremento de la variable de control en el ciclo iterativo for, se usa la función range con tres argumentos así:

range(valor_inicial, valor_final, incremento/decremento)

Cuando se omite, el incremento/decremento es implícitamente igual a 1.

El ciclo siempre incluye el **valor_inicial** y excluye el **valor_final** durante la iteración:

```
Ejemplo

for i in range(10, 0, -2):

print(i)

print('fin del ciclo')
```

```
10

8

6

4

2

fin del ciclo
```

1.3 Usos y estructura

La sentencia **for** se puede usar para imprimir secuencias, tales como cadenas, listas, tuplas, conjuntos o diccionarios.

Uso del ciclo iterativo for en cadenas

```
Ejemplo

for character in "hola mundo":

print(character)

print("fin del ciclo")
```

```
h

o

l

a

m

u

n

d

o

fin del ciclo
```

Uso de la estructura de control iterativa for en listas

```
Ejemplo:

nombre = "Alicia"

edad = 25

print(f"Me llamo {nombre} y tengo {edad} años.")

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

```
for numero in numeros:

print (numero, end=" " )

print("fin del ciclo")
```

```
Me llamo Alicia y tengo 25 años

0 1 2 3 4 5 fin del ciclo
```

Uso del bucle *for* en tuplas

```
Ejemplo

conexion_bd = ("115.16.10.5","root","7890","clientes")

print("la tupla es: ")

for parametro in conexion_bd:

print (parametro)

print ("fin del ciclo")
```

```
la tupla es:

115.16.10.5

root

7890

clientes

fin del ciclo
```

Uso del ciclo iterativo *for* en conjuntos

```
Ejemplo

A={1, 2, 3, 5, 3, 6, 4, 2}
```

```
print ("Los elementos del conjunto A son:")

for elemento in A:

print(elemento)

print ("fin del ciclo for")
```

```
Los elementos del conjunto A son:

1

2

3

4

5

6

fin del ciclo for
```

Uso del bucle *for* en diccionarios

```
Ejemplo

ensalada = {'Manzana':'verde', 'Patilla':'rosado', 'Banano':'amarillo', 'Papaya':'naranja'}

for nombre, color in ensalada.items():

print (nombre, "es de color", color)

print ("fin del ciclo")
```

```
Manzana es de color verde

Patilla es de color rosado

Banano es de color amarillo

Papaya es de color naranja
```

```
fin del ciclo
```

Otra forma de hacerlo en un diccionario es:

```
Ejemplo

ensalada = {'Manzana':'verde', 'Patilla':'rosado', 'Banano':'amarillo', 'Papaya':'naranja'}

for elemento in ensalada:

print (elemento, 'es de color', ensalada[elemento] )

print ("fin del ciclo")
```

```
Manzana es de color verde

Patilla es de color rosado

Banano es de color amarillo

Papaya es de color naranja

fin del ciclo
```

1.4 Sentencias BREAK, CONTINUE y PASS

Nolasco(2018) afirma que con frecuencia se requiere interrumpir un ciclo repetitivo, bien sea porque se cumple una condición y se desea salir inmediatamente de él o simplemente se requiere volver a realizar otra iteración sin ejecutar las demás instrucciones del bloque de código. Algunas veces el programador suspende la codificación de un ciclo iterativo y desea colocar un mensaje temporal. Para esto existen las sentencias *Break, Continue y Pass.*

BREAK

A veces los ciclos se vuelven infinitos, porque la condición siempre es verdadera. Existe una instrucción de Python, break, que permite salir de un ciclo en medio de su ejecución.

```
Ejemplo

Print (“Uso de la sentencia break”)

for caracter in "PYTHON SENA":

if caracter == "N":
```

```
break

print ("El caracter actual es : " +caracter)

print (“fin del ciclo”)
```

```
Uso de la sentencia break

el caracter actual es: P

el caracter acutal es: Y

el caracter actual es: T

el caracter actual es: H

el caracter actual es: O

fin del ciclo
```

CONTINUE

A veces se requiere en los ciclos iterativos que se ignoren las siguientes instrucciones y regresar al inicio del bucle, para continuar con una nueva iteración. Para solucionar esto existe la instrucción *continue*.

En el siguiente ejemplo, si la variable toma el valor de 9, salta al inicio e ejecuta otra iteración.

```
Ejemplo

print("Uso de la sentencia continue")

variable = 15

while variable > 1:

variable = variable -2

if variable == 9:

continue

print ("Valor actual de la variable : " + str(variable))
```

```
print ("fin del ciclo")
```

```
Uso de la sentencia continue

Valor actual de la variable: 13

Valor actual de la variable: 11

Valor actual de la variable: 7

Valor actual de la variable: 5

Valor actual de la variable: 3

Valor actual de la variable: 1

fin del ciclo
```

PASS

La sentencia pass, tal como lo dice su nombre (pasar), es una sentencia nula, o sea que no pasa nada cuando se ejecuta. Se utiliza pass cuando se requiere por sintaxis una instrucción pero no se quiere ejecutar ningún código. También se utiliza en programación donde donde el código irá a futuro, pero no ha sido escrito todavía, utilizándolo como un relleno temporal, es decir, se encuentra “en construcción”.

```
Ejemplo

print("Uso de la sentencia pass")

for letra in "PYTHON SENA":

if letra == "N":

pass

print ("El caracter actual es :" + letra)

print ("fin del ciclo")
```

```
Uso de la sentencia pass

El caracter actual es: P
```

```
El caracter actual es: Y

El caracter actual es: T

El caracter actual es: H

El caracter actual es: O

El caracter actual es: N

El caracter actual es:

El caracter actual es: S

El caracter actual es: E

El caracter actual es: N

El caracter actual es: A

fin del ciclo
```

1.5 Estructura *FOR - ELSE*

De forma similar a la sentencia ***if***, la estructura ***for*** también se puede combinar con una sentencia ***else***.

El bloque ***else*** se ejecutará cuando la expresión condicional del bucle for sea ***False***.

El siguiente ejemplo presenta una conexión a una base de datos de empleados:

```
Ejemplo

db_connection = "115.0.0.5","7890","root","empleados"

for parametro in db_connection:

print parametro

else:

print """El comando PostgreSQL es:

$ psql -h {server} -p {port} -U {user} -d {db_name}""".format(

server=db_connection[0], port=db_connection[1],
```

```
user=db_c
```

Ciclo WHILE

Según Fernández(2016), un ciclo de control iterativo ***WHILE*** permite repetir la ejecución de un bloque de instrucciones mientras se cumpla una condición, es decir, mientras la condición tenga el valor ***True.***

2.1 Estructura WHILE

La estructura de la sentencia de control while es la siguiente:

```
while condicion:

cuerpo del bucle
```

Las variables que se encuentran en la condición se denominan variables de control. Las variables de control se deben definir antes del ***while*** y modificarse dentro del ***while***.

Si el resultado es True se ejecuta el bloque de instrucciones del bucle. Una vez ejecutado el bucle, se repite el proceso una y otra vez mientras la condición sea verdadera.

Si el resultado es ***False***, no se ejecuta el bloque de instrucciones del bucle y continúa la ejecución en la siguiente instrucción después del bucle en el programa.

Una forma de incrementar la variable de control es usando el símbolo +=

```
Ejemplo

i = 1

while i <= 11:

print(i)

i += 2

print("Programa Finalizado")
```

Donde i += 2 significa que el valor de i se incrementa en 2 cada vez que se ejecuta el ciclo repetitivo.

```
1
```

```
3

5

7

9

11

Programa finalizado
```

Una ventaja adicional del bucle while es que el número de iteraciones lo puede definir el usuario durante el bucle. Por ejemplo, el programa solicita un número al usuario una y otra vez hasta que el usuario acierte.

```
Ejemplo

i = 1

numero1=100

numero2 = int(input("Digite un número de 1 a 100: "))

while numero2 != numero1:

i += 1

numero2 = int(input("Digite un número de 1 a 100: "))

print ("Acertaste en el intento No.", i)
```

```
Digite un número de 1 a 100: 5

Digite un número de 1 a 100: 8

Digite un número de 1 a 100: 10

Digite un número de 1 a 100: 100

Acertaste en el intento No: 4
```

2.2 Tipos de bucle WHILE

La sentencia ***while*** permite ejecutar ciclos, es decir, ejecutar un bloque de código múltiples veces.

El ciclo ***while*** realiza múltiples iteraciones con base en el resultado de una expresión lógica: *True o False*.

Para Zed(2014), existen 3 tipos de ciclos repetitivos con la sentencia while:

While controlado por contador.

While controlado por evento.

While con Else.

A continuación se detalla el uso de cada uno de ellos, dependiendo de la situación que deba resolver el programador.

Ciclo *while* controlado por contador

Es un ciclo donde existe una variable o contador con un valor inicial, el cual se incrementa / decrementa en cada iteración hasta que llega a un valor final definido en la condición incluida en la sentencia ***While***.

```
print ("Uso del ciclo while con contador")

total, valor = 0, 1

while valor <= 12:

print (valor)

total = total + valor

print ("el total parcial es ",total)

valor = valor + 1

print ("El total final es ",total)

print( "fin del ciclo while")
```

El resultado de este ciclo while controlado por contador es:

```
Uso del ciclo while con contador

1

el total parcial es 1

2

el total parcial es 3

3

el total parcial es 6

4

el total parcial es 10

5

el total parcial es 15

6

el total parcial es 21

7

el total parcial es 28

8

el total parcial es 36

9

el total parcial es 45

10

el total parcial es 55

11
```

```
el total parcial es 66

12

el total parcial es 78

El total final es 78

Fin del ciclo while
```

Ciclo *while* controlado por evento

Es un ciclo repetitivo ***while*** donde se activa un evento dentro del ciclo, el cual causa que el bucle se interrumpa.

```
print ("Uso del ciclo while controlado por evento")

promedio, total, contador = 0, 0, 0

print ("=== Software para parqueadero ===")

placa = input("Introduzca la placa del vehiculo (99 para salir): " )

while placa != "99":

valor= float(input("Digite valor del parqueadero: "))

total = total + valor

contador= contador+1

placa = input("Introduzca la placa del vehiculo (99 para salir): " )

promedio = round(total / contador)

print ("Promedio de vr. parqueadero por vehiculo: " + str(promedio))

print ("Total dinero recaudado: ", round(total))

print( "fin del ciclo while")
```

El resultado de este ciclo while controlado por evento es:

```
Uso del ciclo while controlado por evento
```

```
=== software para parqueadero ===

Introduzca la placa del vehiculo (99 para salir): XER45T

Digite valor del parqueadero: 2500

p Introduzca la placa del vehiculo (99 para salir): JEF56R

Digite valor del parqueadero: 1500

Introduzca la placa del vehiculo (99 para salir): 99

Promedio de vr. parqueadero por vehiculo: 2000

Total dinero recuadado: 4000

fin del ciclo while
```

Ciclo *while* controlado por evento

De manera similar a la sentencia ***if***, el ciclo de control iterativo ***while*** se puede combinar con la sentencia ***else***.

Pérez(2016), explica que la sentencia else ocurre cuando la expresión condicional del ***while*** es ***False***.

```
print ("Uso del ciclo while + else")

promedio, total, contador = 0, 0, 0

print ("=== Software para parqueadero ===")

placa = input("Introduzca la placa del vehiculo (99 para salir): " )

while placa != "99":

valor= float(input("Digite valor del parqueadero: "))

total = total + valor

contador= contador+1

placa = input("Introduzca la placa del vehiculo (99 para salir): " )

else:
```

```
promedio = round(total / contador)

print ("Promedio de vr. parqueadero por vehiculo: " + str(promedio))

print ("Total dinero recaudado: ", round(total))

print( "fin del ciclo while")

print("final del ciclo while + else")
```

El resultado de este ciclo ***while*** con ***else*** es:

```
Uso del ciclo while + else

=== software para parqueadero ===

Introduzca la placa del vehiculo (99 para salir): XER45T

Digite valor del parqueadero: 2500

p Introduzca la placa del vehiculo (99 para salir): JEF56R

Digite valor del parqueadero: 1500

Introduzca la placa del vehiculo (99 para salir): 99

Promedio de vr. parqueadero por vehiculo: 2000

Total dinero recuadado: 4000

fin del ciclo while

final del ciclo while + else
```

2.3 Sentencias BREAK, CONTINUE y PASS

Similar al ciclo iterativo ***for***, en el bucle con ***while*** existen existen las sentencias ***Break***, ***Continue*** y ***Pass***.

BREAK

Break termina el bucle actual y continúa con la ejecución de la siguiente instrucción después del ciclo.

```
print("Uso de la sentencia BREAK en while")
```

```
variable = 35

while variable > 1:

variable = variable -5

if variable == 10:

break

print ("Valor actual del caracter : " + str(variable))

print ("fin del ciclo")
```

```
Uso de la sentencia BREAK en while

Valor actual del caracter : 30

Valor actual del caracter : 25

Valor actual del caracter : 20

Valor actual del caracter : 15

fin del ciclo
```

CONTINUE

La sentencia ***Continue*** en Python, regresa el control de la ejecución del programa al inicio del bucle, desechando todas las instrucciones que quedan en la iteración actual del bucle e inicia una nueva iteración.

```
print("Uso de la sentencia CONTINUE en while")

variable = 35

while variable > 1:

variable = variable -5

if variable == 15:

continue

print ("Valor actual de la variable : " + str(variable))
```

```
print ("fin del ciclo")
```

```
Uso de la sentencia CONTINUE en while

Valor actual del caracter : 30

Valor actual del caracter : 25

Valor actual del caracter : 20

Valor actual del caracter : 10

Valor actual del caracter : 5

Valor actual del caracter : 0

fin del ciclo
```

PASS

La sentencia ***Pass*** se utiliza cuando se requiere por sintaxis una instrucción pero no se requiere ejecutar ningún comando o código.

```
print("Uso de la sentencia PASS en while")

variable = 35

while variable > 1:

variable = variable -5

if variable == 25:

pass

print ("Valor actual de la variable : " + str(variable))

print ("fin del ciclo")
```

```
Uso de la sentencia PASS en while

Valor actual del caracter : 30

Valor actual del caracter : 25
```

Valor actual del caracter : 20

Valor actual del caracter : 15

Valor actual del caracter : 10

Valor actual del caracter : 5

Valor actual del caracter : 0

fin del ciclo