Guía Práctica N.º 9: Estructuras Cíclicas (WHILE) en Python

Resultado De Aprendizaje:

- Conocer las estructuras cíclicas en Python modo consola
- Desarrollar ejercicios propuestos con la estructura cíclica WHILE para resolver situaciones cotidianas.

Existe dos tipos de bucles, los que tienen un **número de iteraciones no definidas**, como lo es la estructura **while**, y los que tienen un **número de iteraciones definidas**, como lo es el **for**.

El ciclo while

En Python esta estructura permite ejecutar un bloque de código *repetidamente* **mientras** se cumpla una condición, la cual se evalúa en cada iteración del ciclo.
El código se ejecutará **mientras** una condición determinada se cumpla, de ahí su nombre. Cuando se deje de cumplir, se saldrá del bucle y se continuará la ejecución normal.

Estructura básica

```
1.  while condición:
2.      bloque de código
```

El while tiene dos partes:

- La **condición** que se tiene que cumplir para que se ejecute el código.
- El **bloque de código** que se ejecutará mientras la condición se cumpla.

Ejemplo:

Código en el editor	Salida en Consola

`x = 5`
`while x > 0:`
`    x -=1`
`    print(x)`
	`4`
	`3`
	`2`
	`1`
	`0`

En el ejemplo anterior se tiene un caso sencillo de **while**. Se tiene una condición **x>0** y un bloque de código a ejecutar mientras dure esa condición **x-=1** y **print(x)**. Por lo tanto, mientras que **x** sea mayor que **0**, se ejecutará el código. Una vez se llega al final, se vuelve a empezar y si la condición se cumple, se ejecuta otra vez. En este caso se entra al bloque de código 5 veces, hasta que, en la sexta, **x** vale cero y por lo tanto la condición ya no se cumple.

Se debe tener cuidado ya que un mal uso del **while** puede dar lugar a bucles infinitos y problemas.
Ejemplo:

```
# No ejecutes esto
while True:
    print("Bucle infinito")
```

Es posible tener un **while en una sola línea**, algo muy útil si el bloque que se quiere ejecutar es corto. En el caso de tener más de una sentencia, se debe separar con **;**

```
x = 5
while x > 0: x-=1; print(x)
```

También se puede usar otro tipo de operación dentro del **while**, como la que se muestra a continuación. En este caso se tiene una lista que **mientras no este vacía**, se va eliminando su primer elemento.

```
x = ["Uno", "Dos", "Tres"]
while x:
    x.pop(0)
    print(x)
```

El resultado sería:

```
['Dos', 'Tres']
['Tres']
[]
```

Else y while

Algo no muy corriente en otros lenguajes de programación pero si en Python, es el uso de la cláusula **else** al final del **while**. La sección de código que se encuentra dentro del else, se ejecutará cuando el bucle termine.

```
x = 5
while x > 0:
    x -=1
    print(x)
else:
    print("El bucle ha finalizado")
```

Interrupciones en el ciclo while

Son mecanismos que permiten salir del ciclo antes de que la condición lógica se vuelva falsa. Hay dos tipos principales de interrupciones en el ciclo while en Python: la instrucción **break** y la instrucción **continue**.

La instrucción **break** permite salir del ciclo while inmediatamente, cuando se cumple una determinada condición. Por ejemplo, si se tiene un ciclo **while** que se ejecuta hasta que se ingresa la palabra salir, se puede utilizar la instrucción **break** para salir del ciclo en el momento en que se ingresa dicha palabra.

Por otro lado, la instrucción **continue** permite saltar la iteración actual del ciclo while y **continuar** con la siguiente iteración. Esto significa que cualquier código que esté después de la instrucción continue dentro del bloque de código del ciclo while no se ejecutará en esa iteración. Por ejemplo, si se tiene un ciclo while que itera sobre una lista de elementos y se desea omitir un elemento en particular, se puede utilizar la instrucción continue para saltar la iteración actual cuando se encuentra dicho elemento.

Ejemplo:

```
num = 0
while num <= 10:
    num += 1
    if num == 9:
        break
    elif num == 6:
        continue
    print(num)
```

El resultado sería:

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

Observe que el ciclo while esta programado para que termine cuando la variable num sea igual 10 pero nunca llega a ese valor porque la condicion de **num==9** hace que se ejecute la instrucción **break** y finaliza el ciclo completo, pero en cambio la instrucción **continue** solo hace que termine la **iteracion en curso** por lo tanto cuando **num==6** continua a la siguiente iteracion sin pasar por la instrucción de **print(num)** haciendo que se omita el numero 6 en los resultados.

GUIA DE EJERCICIOS.

1. **Imprimir los números del 1 al 10**: Escribe un programa que imprima los números del 1 al 10 utilizando un bucle while.
2. **Imprimir los números impares del 1 al 20**: Escribe un programa que imprima los números pares del 1 al 20 utilizando un bucle while.
3. **Imprimir la tabla de multiplicar de un numero cualquiera**: Escribe un programa que imprima la tabla de multiplicar de un numero introducido por el usuario utilizando un bucle while.
4. **Sumar los múltiplos de 3 antes del 20**: Escribe un programa que sume los múltiplos de 3 que están antes del número 20 y luego imprimir el resultado.
5. **Calcular el factorial de un número**: Escribe un programa que calcule el factorial de un número dado.
6. **Imprimir los números de la serie de Fibonacci antes del 25**: Escribe un programa que imprima los números de la serie de Fibonacci que aparecen antes del número 125.
7. **Imprimir una pirámide de asteriscos**: Escribe un programa que imprima una pirámide de asteriscos.