BUCLES PYTHON

Bucles o ciclos

Mientras que los condicionales nos permiten ejecutar diferentes fragmentos de código dependiendo de ciertas condiciones, los bucles nos permiten ejecutar un mismo fragmento de código un cierto número de veces, mientras se cumpla una determinada condición.

Un bucle es una parte de nuestro programa que se ejecuta continuamente mientras se cumpla determinada condición. Existen también los bucles infinitos, en python son aquellos en los cuales la condición nunca se cumple.

Estructuras repetitivas

Las estructuras repetitivas están compuestas por tres partes:

1. La condición lógica,
2. Las acciones o instrucciones que se repiten, y
3. La salida de la repetición

Existen tres tipos de estructuras repetitivas:

1. Mientras (**while**)
2. Repetir hasta (**do while**), Python no tiene el bucle do-while.
3. Para (**for**)

Variables acumuladores

Es una variable que, como su nombre lo indica, va a ser usada para sumar sobre sí misma un conjunto de valores.
Es necesario haber inicializado su valor antes del comienzo de un ciclo de repetición.
La inicialización consiste en asignarle al sumador un valor inicial, es decir el valor desde el cual necesitamos se inicie la sumatoria (por lo general comienzan en cero).

Ejemplo:

$\mathbf{S = 0.}$

$\mathbf{S = S + N}$

Variable contadora

Es una variable que se encuentra en ambos miembros de una asignación a la que se le suma un valor constante.
Un contador es una variable cuyo valor se incrementa o decrementa en una cantidad constante cada vez que se produce un determinado suceso, acción o iteración. La inicialización implica darle un valor inicial, en este caso, el número desde el cual necesitamos se inicie el conteo (por lo general comienzan en cero).

Ejemplo:

C=0 (inicializa la variable C en 0).
C=C+1 (incrementar).
H=H-1 (decrementar).

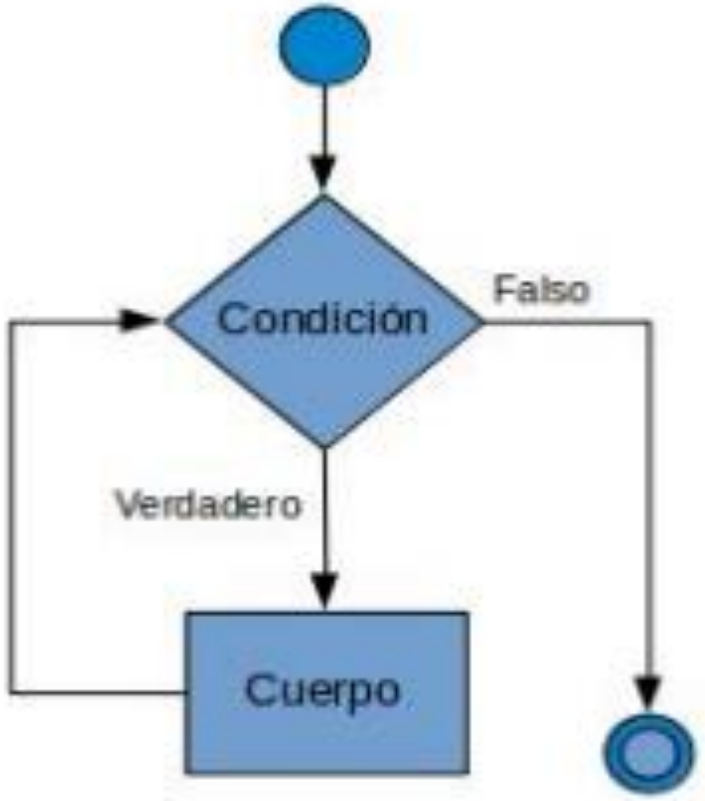
Bucle While en Python

La sentencia o bucle while en Python es una sentencia de control de flujo que se utiliza para ejecutar un bloque de instrucciones de forma continuada mientras se cumpla una condición determinada.

```
while condicion:
    bloque de código
```

Figure: while en Python

Es decir, mientras la condición se evalúe como **True**, se ejecutarán las instrucciones y sentencias de bloque de código.



Ciclo `while`

Figure: Diagrama de flujo

Bucle For

Iterar: significa realizar cierta acción repetidas veces y en el caso de for hace referencia a recorrer elementos iterables.

En Python **for** se utiliza como una forma genérica de iterar sobre una secuencia

Sintaxis:

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

Figure: for en Python

Bucle for en Python

Tipos de iterables (secuencias en Python)

La secuencia indica el número de veces que se va a repetir el código.
Los tipos de iterables (secuencias) que podemos encontrar son:

1. Listas: Estructura de datos que nos permiten almacenar cualquier tipo de valor como enteros, cadenas y hasta otras funciones
2. Strings: Los cadenas (o strings) son un tipo de datos compuestos por secuencias de caracteres que representan texto. Estas cadenas de texto son de tipo str.
3. Rangos: se usa la función **range** en Python que es un tipo que se utiliza para representar una secuencia inmutable de números. Uno de sus principales usos es junto a la sentencia for, para definir un bucle sobre el que se itera un número determinado de veces.

Diferencias entre For y While en python

El bucle for podría traducirse como **"para"** y el While como **"mientras"**.

En el caso de **for** no nos permite realizar un ciclo infinito.
A diferencia de **while** que si nos brinda esa posibilidad, a la vez que desde el comienzo no están declaradas la cantidad de iteraciones a realizar.

Generar números aleatorios en Python

Para generar números aleatorios en Python se hace uso del módulo random de la biblioteca estándar. **import random**

Generar números aleatorios entre dos valores en Python

1. **randint(a, b):** Para generar números aleatorios en Python de valor entero, se suele utilizar la función randint(). La función randint(a, b) devuelve un número entero comprendido entre a y b (ambos inclusive) de forma aleatoria.
2. **randrange(a, b, salto):** La función randrange(a, b, salto) genera números enteros aleatorios comprendidos entre a y b separados entre sí con un salto.
3. **random():** La función random() devuelve un float comprendido entre [0.0 y 1.0)
4. **uniform(a, b)** La función uniform(a, b) devuelve un float aleatorio comprendido entre a y b (ambos inclusive).

Agregar Elementos a una Lista Vacía

Si se requiere agregar elementos a una lista vacía, puede hacer el llamado a los métodos **append()** e **insert()**:

1. **append()**: agrega un elemento al final de la lista.
2. **insert()**: agrega un elemento en una posición específica de la lista. Esta posición representa el índice de la lista.

Control de bucles, break, continue y pass en python

1. **Break:**
   Esta instrucción se utiliza para finalizar un bucle, es decir, salir de el y continuar con la ejecución del resto de instrucciones del programa.
2. **Continue:**
   La instrucción continue dentro de un bucle obliga al interprete a volver al inicio del bucle obviando todas las instrucciones o iteraciones debajo de el.
3. **Pass** La instrucción Pass es como lo indica su nombre una expresión nula, no hace nada.. Es casi como si no existiera, pero nos permite crear un bucle sin colocar código en su cuerpo para añadirlo mas tarde utilizándolo como un relleno temporal.