

Estructuras Condicionales. Estructuras Ciclicas.

Elaborado por: Inst. Leticia Mendieta

Estructuras Condicionales

Las estructuras de control condicionales, son aquellas que nos permiten evaluar si una o más condiciones se cumplen, para decir qué acción vamos a ejecutar. La evaluación de condiciones, solo puede arrojar 1 de 2 resultados: verdadero o falso (True o False).

Para describir la evaluación a realizar sobre una condición, se utilizan **operadores relacionales** (o de comparación):

| Símbolo | Significado | Ejemplo | Resultado |
|---------|-------------------|---------------|-----------|
| == | Igual que | 5 == 7 | False |
| != | Distinto que | rojo != verde | True |
| < | Menor que | 8 < 12 | True |
| > | Mayor que | 12 > 7 | True |
| <= | Menor o igual que | 12 <= 12 | True |
| >= | Mayor o igual que | 4 >= 5 | False |

Y para evaluar más de una condición simultáneamente, se utilizan operadores lógicos:

| Operador | Ejemplo | Explicación | Resultado |
|----------|--------------------|-----------------|-----------|
| and | 5 == 7 and 7 < 12 | False and False | False |
| and | 9 < 12 and 12 > 7 | True and True | True |
| and | 9 < 12 and 12 > 15 | True and False | False |
| or | 12 == 12 or 15 < 7 | True or False | True |
| or | 7 > 5 or 9 < 12 | True or True | True |
| xor | 4 == 4 xor 9 > 3 | True o True | False |
| xor | 4 == 4 xor 9 < 3 | True o False | True |

Las estructuras de control de flujo condicionales, se definen mediante el uso de tres palabras claves reservadas, del lenguaje: if (si), elif (sino, si) y else (sino).

Ejemplo: Si gasto hasta 10000, pago con dinero en efectivo. Si no, si gasto más de 10000 pero menos de 30000, pago con tarjeta de débito. Si no, pago con tarjeta de crédito.

```
if compra <= 10000:
```

```
    print ("Pago en efectivo")
```

```
elif compra > 10000 and compra < 30000:
```

```
    print ("Pago con tarjeta de débito")
```

```
else:
```

```
    print ("Pago con tarjeta de crédito")
```


Estructuras Cíclicas

A diferencia de las estructuras de control condicionales, las iterativas (también llamadas cíclicas o bucles), nos permiten ejecutar un mismo código, de manera repetida, mientras se cumpla una condición.

En Python se dispone de dos estructuras cíclicas:

- El bucle while
- El bucle for

Bucle While

- Este bucle, se encarga de ejecutar una misma acción "mientras que" una determinada condición se cumpla. Ejemplo: Mientras que año sea menor o igual a 2012, imprimir la frase "Informes del Año año".

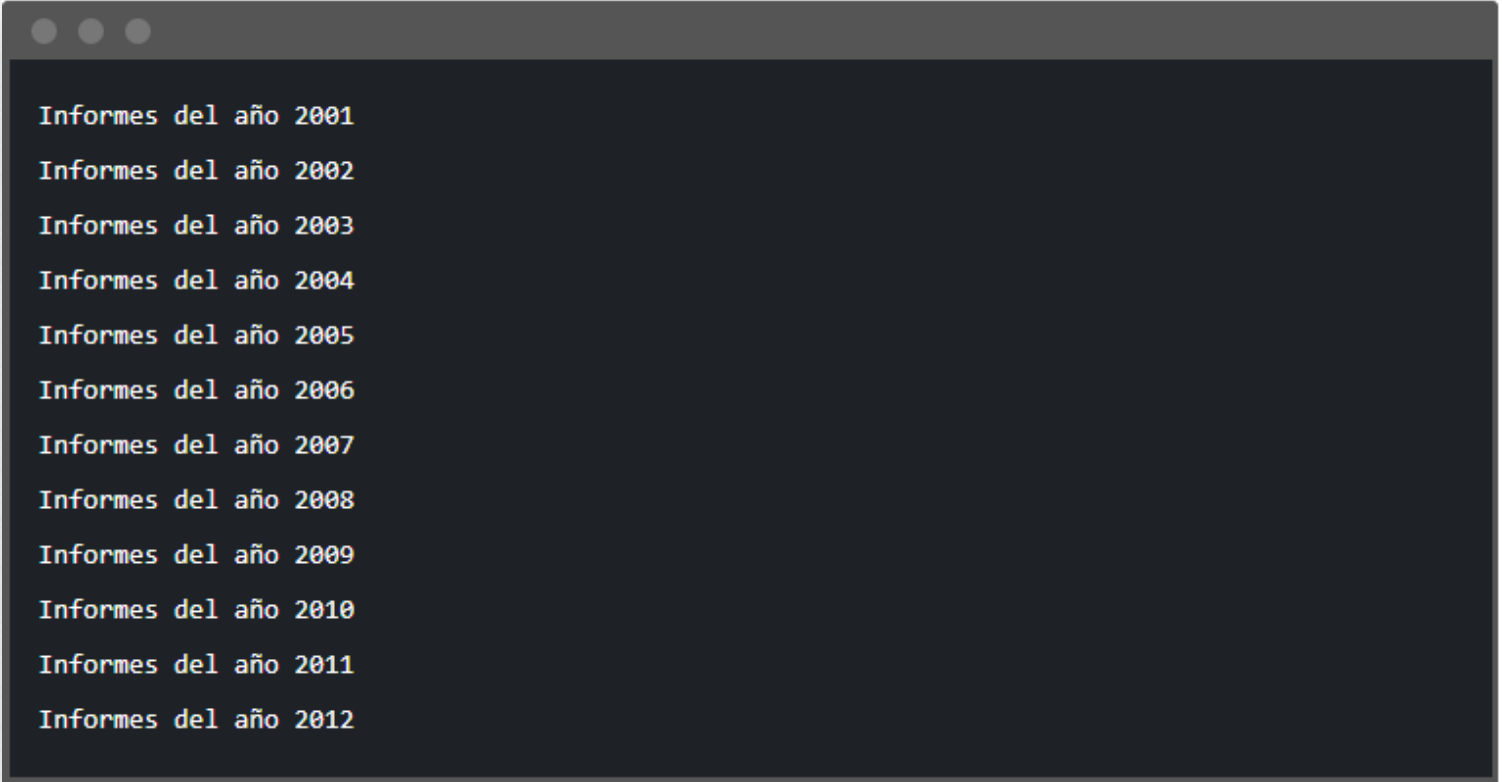
```
anio = 2001
```

```
while anio <= 2012:
```

```
    print ("Informes del Año", str(anio))
```

```
    anio += 1
```

La iteración anterior, generará la siguiente salida:



```
Informes del año 2001
Informes del año 2002
Informes del año 2003
Informes del año 2004
Informes del año 2005
Informes del año 2006
Informes del año 2007
Informes del año 2008
Informes del año 2009
Informes del año 2010
Informes del año 2011
Informes del año 2012
```

A terminal window with a dark background and light-colored text. It displays a list of 12 lines, each starting with "Informes del año" followed by a year from 2001 to 2012. The window has a title bar with three small circles (red, yellow, green) on the left.

Si miras la última línea:

```
anio += 1
```

Podrás notar que en cada iteración, incrementamos el valor de la variable que condiciona el bucle (anio). Si no lo hiciéramos, esta variable siempre sería igual a 2001 y el bucle se ejecutaría de forma infinita, ya que la condición (anio <= 2012) siempre se estaría cumpliendo.

Pero ¿Qué sucede si el valor que condiciona la iteración no es numérico y no puede incrementarse? En ese caso, podremos utilizar una estructura de control condicional, anidada dentro del bucle, y frenar la ejecución cuando el condicional deje de cumplirse, con la palabra clave reservada break:

```
while True:  
    nombre = input("Indique su nombre: ")  
    if nombre:  
        break
```

El bucle anterior, incluye un condicional anidado que verifica si la variable nombre es verdadera (solo será verdadera si el usuario tepea un texto en pantalla cuando el nombre le es solicitado). Si es verdadera, el bucle para (break). Sino, seguirá ejecutándose hasta que el usuario, ingrese un texto en pantalla.

Bucle For

El bucle for, en Python, es aquel que nos permitirá iterar sobre una variable compleja, del tipo lista o tupla:

1) Por cada nombre en mi_lista, imprimir nombre

```
mi_lista = ['Juan', 'Antonio', 'Pedro', 'Herminio']
```

```
for nombre in mi_lista:
```

```
    print(nombre)
```

2) Por cada color en mi_tupla, imprimir color:

```
mi_tupla = ('rosa', 'verde', 'celeste', 'amarillo')
```

```
for color in mi_tupla:
```

```
    print (color)
```

Bucle For

En los ejemplos anteriores, nombre y color, son dos variables declaradas en tiempo de ejecución (es decir, se declaran dinámicamente durante el bucle), asumiendo como valor, el de cada elemento de la lista (o tupla) en cada iteración.

Otra forma de iterar con el bucle for, puede emular a while:

3) Por cada año en el rango 2001 a 2019, imprimir la frase "Informes del Año año":

```
# -*- coding: utf-8 -*-
```

```
for anio in range(2001, 2019):
```

```
    print ("Informes del Año", str(anio))
```


Un poco acerca de Estructuras de Datos en Python

Listas

Una lista es una estructura de datos que contiene una colección o secuencia de datos. Los datos o elementos de una lista deben ir separados con una coma y todo el conjunto entre corchetes. Se dice que una lista es una estructura mutable porque además de permitir el acceso a los elementos, pueden suprimirse o agregarse nuevos.

```
ListaEstaciones = ["Invierno", "Primavera", "Verano", "Otoño"] # Declara  
lista
```

Un poco acerca de Estructuras de Datos en Python

Tuplas

Una tupla permite tener agrupados un conjunto inmutable de elementos, es decir, en una tupla no es posible agregar ni eliminar elementos. Las tuplas se declaran separando los elementos por comas y éstos, opcionalmente, pueden ir entre paréntesis. Se recomienda el uso de paréntesis para evitar ambigüedades del tipo: `print(9, 8, 7)` y `print((9, 8, 7))`.

```
TuplaDiasSemana = ("LU", "MA", "MI", "JU", "VI", "SA", "DO") # Declara tupla
```

Estructuras de Datos se profundizara en la ultima unidad del curso...

Fuentes Consultadas

<https://uniwebsidad.com/libros/python/capitulo-1>

https://www.w3schools.com/python/python_for_loops.asp

<https://www.tutorialesprogramacionya.com/pythonya/detalleconcepto.php?punto=32&codigo=32&inicio=30>