José Antonio Vacas Martínez

Aprende a programar Python

Logo Python

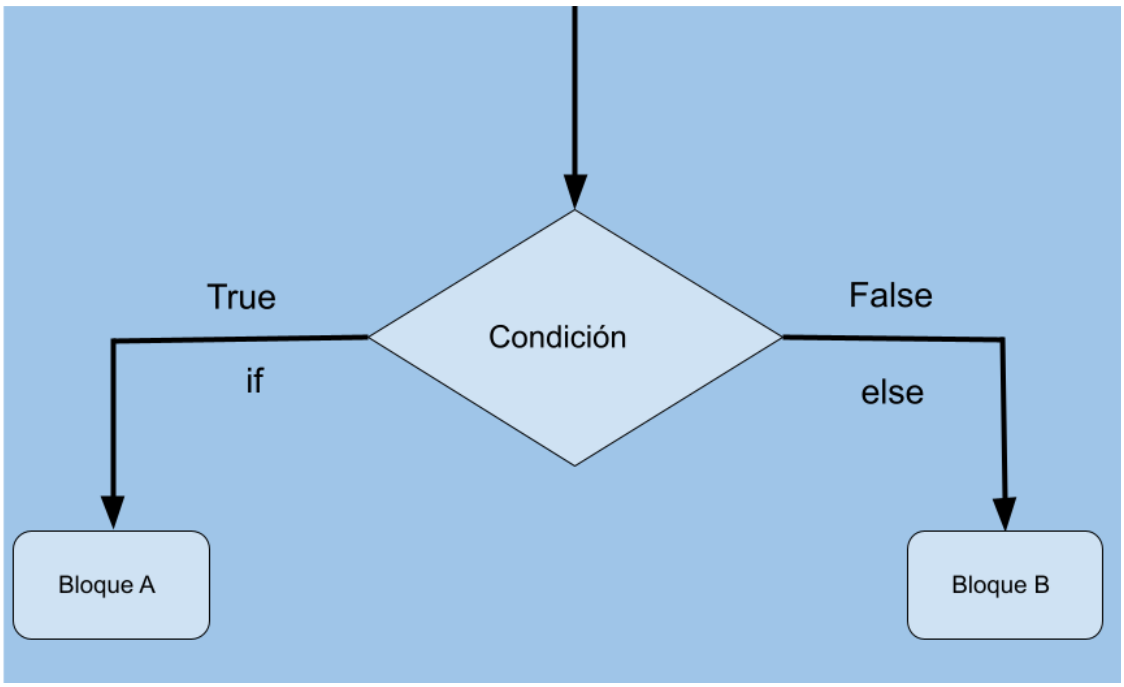
Versión 0.99.2

José Antonio Vacas Martínez

https://elCacharreo.com

Octubre 2022

Sentencias condicionales



Sentencias condicionales

En ocasiones nos interesa que determinadas sentencias de nuestro programa se ejecuten o no dependiendo del valor de alguna variable o de relación entre variables.

En estos casos usamos sentencias condicionales. La más simple es la sentencia **if** que irá seguida de una comparación que debe de dar como resultado un valor booleano y después pondremos "**:**". En caso de que este valor sea **True** se ejecutará el bloque de líneas que sigue a la sentencia. Si es **False** no se ejecutará.

Recordemos que para definir un bloque haremos que las líneas de código empiecen con 4 espacios (no es obligatorio que sean 4, pero es lo recomendado) más a la derecha. **Es fundamental que respetemos el formato de las líneas.**

Más adelante vamos a hablar de las posibles comparaciones, pero ahora mismo vamos a usar una comparación entre una variable y un número, y veremos cuál es mayor:

```
strEdad = input('¿Qué edad tiene? ')
edad = int(strEdad) # Convertimos a entero para comparar
if edad > 65: # ¿es el valor de edad mayor que 65?
    print('Usted puede jubilarse')
```

Llamamos **operadores relacionales** a los que usamos para comparar valores entre sí. Todos ellos dan un resultado booleano: **True** o **False**

Operador	Descripción
>	Mayor que
<	Menor que
>=	Mayor o igual
<=	Menor o igual
==	Igual
!=	Distinto

¡¡Cuidado con el operador "igual" que se representa con "dos signos igual" "=="!! No confundirlo con la asignación que usa sólo uno "=".

Del mismo modo podemos usar comparaciones entre cadenas, comparando el valor de la variable con una cadena o dos variables entre sí.

Vamos a hacer un ejemplo sencillo de lo que sería un interface de comandos para un ordenador, donde el usuario introduce un comando y simulamos lo que el ordenador tendría que hacer con sentencias *print*:

```
# Simulación de consola de comandos v1
comando = input('Introduzca el comando: ')
if comando == 'Apagar':
    print('Apagando el equipo...')
if comando == 'Encender':
    print('Encendemos el equipo')
```

También podemos comparar para "ordenar" cadenas, puesto que existe un orden alfabético en ellas:

```
miPalabra = input('Introduzca una palabra: ')
if miPalabra > 'hola':
    print('la palabra "' + miPalabra + '" se ordena después de "hola"')
if miPalabra < 'adios':
    print('la palabra "' + miPalabra + '" se ordena antes que "adios"')
```

Me gustaría resaltar el uso de los distintos tipos de comillas para poder mostrarlas en el resultado de la sentencia print.

Tenemos que tener presente que si al ordenar cadenas que contienen números la ordenación no se hará como esperamos, numéricamente, puesto que se va realiza carácter a carácter:

```
if '10' < '9':
    print('esto me vuelve loco')
```

Es decir, la comparación se hace por caracteres, no como un todo: se compara el carácter '1' con el carácter '9', y por eso la cadena "10" es menor que otra.

Si convertimos las cadenas a enteros todo es como esperamos:

```
if int('10') < int('9'):
    print('esto me vuelve loco')
```

Podemos almacenar el resultado de una comparación en una variable booleana, es decir, la variable booleana guardará el resultado de la comparación que podremos usar después:

```
contento = input('¿Está contento?') == "si"
if contento == True:
    print('Está contento')
```

La variable 'contento' tendrá un valor booleano True o False, según la respuesta.

No me canso de repetir la diferencia entre el operador *comparación de igualdad* "**==**" y el operador *asignación* "**=**".

Veamos ahora nuestro interface de comandos usando estas variables booleanas:

```
# consola de comandos v1.5
comando = input('Introduzca el comando: ')
bApagado  = comando == 'Apagar'
bEncender = comando == 'Encender'

if bApagado == True:
    print('Apagando el equipo')
if bEncender == True:
    print('Encendiendo el equipo')
```

Sentencias "else" y "elif"

En muchas ocasiones tenemos un bloque de instrucciones para el caso que se cumpla y otro para cuando no se cumpla. En ese caso incluiremos la sentencia **else :** para delimitar el bloque que se ha de ejecutar en caso de que no se cumpla la condición:

```
strEdad = input('¿Qué edad tiene? ')
edad = int(strEdad)
if edad > 65: # ¿es el valor de edad mayor que 65?
    print('Usted puede jubilarse')
else:  # no se cumple la condición
    print('Usted NO puede jubilarse')
```

También se da a veces que existen varios casos excluyentes entre sí y que queremos hacer comparaciones para todos ellos. En ese caso usamos **elif** que equivale a un "else if". Veamos un caso en el ahora comparamos como el operador "menor que":

```
strEdad = input('¿Qué edad tiene? ')
edad = int(strEdad)
if edad < 12:
    print('niño')
elif edad < 17 : # No es menor de 12 y sí menor de 17
    print('adolescente')
elif edad < 25 : # No es menor de 17 y sí menor de 25
    print('joven')
elif edad < 50 : # No es menor de 25 y sí menor de 50
    print('una buena edad')
elif edad < 65 : # No es menor de 50 y sí menor de 65
    print('casi abuelete')
else:  # es mayor o igual de 65
    print('Usted puede jubilarse')
```

Conviene fijarse en este ejemplo que no funcionaría igual si pusiéramos sólo sentencias if puesto que cuando un valor es menor que 25 también lo es que 50 y los sucesivos...

Ahora podemos hacer más eficiente nuestro interface de comandos, puesto que sólo hacemos las comparación hasta que se encuentra el comando solicitado y ya se dejan de hacer más comparaciones:

```
# Simulación de consola de comandos v2
comando = input('Introduzca el comando: ')
if comando == 'Apagar':
    print('Apagando el equipo...')
elif comando == 'Encender':
    print('Encendemos el equipo')
else :
    print('Comando ' + comando + ' no reconocido')
```

Operadores lógicos

A veces tenemos que comprobar varias expresiones combinando los resultados de todas ellas en un único resultado.

Para ello usaremos los operadores lógicos

Operador	Resultado
and	todos los valores han de ser *True*
or	al menos uno de los valores ha de ser *True*

not	se invierte el resultado

Podemos usar los operadores lógicos para encadenar varias comparaciones como vemos en este ejemplo que determina si un año es bisiesto para lo que hay que ver si divisible por 4, pero no múltiplo de 100 salvo que lo sea de 400 (sí, es un poco lioso, pero así es como se hace...)

Código

```
# Programa que determina si un año es o no bisiesto
# Divisible por 4
# No divisible por 100 salvo que
# Sea divisible por 400
year = int(input('Introduzca el año: '))

if (year%4)==0 and ( (year%400)==0 or  not ((year%100)== 0)):
  print('Es bisiesto!!')
else:
  print ('No es bisiesto!!')
```

Otra forma de hacer varias comparaciones es usando notación matemática, por ejemplo, para comprobar si una variable está dentro de un rango.

Normalmente haríamos:

```
if minimo < variable and variable < maximo :
    ...
```

Usando notación matemática queda más claro y compacto:

```
if minimo < variable < maximo :
    ...
```

Sentencias anidadas

A veces la decisión es compleja y necesitamos poner unas condiciones dentro de otras. Decimos entonces que las condiciones están anidadas. En este caso tenemos que extremar el cuidado con la indentación de las líneas.

Vamos a hacer un ejemplo que nos da los días que tiene un mes dado. Convertimos directamente el valor que nos da el usuario a entero, para poder hacer las comparaciones entre números.

En el if/else más externo comprobamos si el valor del mes es correcto, en los interiores según los valores de los meses:

Código

```
# Nos da los días que tiene el mes seleccionado
mes = int(input('Introduce el mes:'))
year = int(input('Introduce el año:'))
# Comprobamos si está entre 1 y 12
if 1 <= mes <= 12:
    if mes == 2:
        if (year%4)==0 and ( (year%400)==0 or  not ((year%100)== 0)):
            dias = 29
        else:
            dias =28
    elif (mes == 4) or (mes == 6) or (mes == 9) or (mes == 11):
        dias = 30
    else:
        dias = 31
    print (f'El mes {mes} del año {year} tiene {dias} días')
else:
    print ('El mes debe ser entre 1 y 12')
```

Vamos a aprovechar lo visto para dar un nivel de seguridad a nuestro interface de comandos. Para ello preguntaremos el nombre al usuario y sólo si el usuario es adecuado daremos acceso a los comandos. Además si quiere apagar el ordenador le pediremos una confirmación:

Codigo

```
# Simulación de consola de comandos v3
usuario = input('Introduzca el nombre de usuario: ')
if usuario == 'pepe' or usuario == 'admin':
    print('Bienvenido ' + usuario)
    comando = input('Introduzca el comando: ')
    if comando == 'Apagar':
         confirmacion = input('Confirme que desea apagar respondiendo "Si": '
)
        if confirmacion == 'Si':
            print('Apagando el equipo...')
        else:
            print('No se ha confirmado el apagado')
    elif comando == 'Encender':
        print('Encendemos el equipo')
    else :
        print('Comando ' + comando + ' no reconocido')
else:
    print('Acceso denegado')
```

Vamos a fijarnos en los diferentes niveles de anidamiento de las sentencias y cómo hemos utilizado el operador lógico “*or*” para comprobar entre los nombres de usuario.