Estructuras de control

Las estructuras de control nos permiten crear programas con múltiples caminos posibles a seguir, dependiendo de ciertas condiciones a comprobar. Estas estructuras son también habituales en otros lenguajes de programación... hablamos de *if*, *while* o *for*.

1. Estructuras selectivas: *if*, *if..else*, *if..elif..else*

Si queremos ejecutar un conjunto de instrucciones si se cumple una determinada condición, debemos añadir esta condición en una cláusula `if`, y las instrucciones vinculadas a esa condición deben quedar **tabuladas** respecto a ese *if*. Por ejemplo:

```
numero = int(input())
if numero > 0:
    numero = numero + 1
    print ("El siguiente número es", numero)
print ("Fin del programa")
```

Podemos también distinguir entre dos posibles caminos con la estructura `if..else`. En este caso, debemos tabular el bloque de instrucciones que vaya asociado a cada parte (*if* o *else*):

```
numero = int(input())
if numero > 0:
    numero = numero + 1
    print ("El siguiente número es", numero)
else:
    print ("El número no es positivo")
print ("Fin del programa")
```

Si necesitamos tener más de dos caminos diferentes, podemos anidar estructuras `if..else` unas dentro de otras...

```
numero = int(input())
if numero > 0:
    numero = numero + 1
    print ("El siguiente número es", numero)
else:
    if numero < -10:
        print ("El número es demasiado bajo")
    else:
        print ("El número no es positivo")
print ("Fin del programa")
```

... pero, en lugar de anidar estas estructuras, también podemos utilizar la cláusula `if..elif` para especificar más de un bloque de condiciones. Podemos enlazar tantas clásulas `elif` como necesitemos, y también concluir con una cláusula `else` si queremos, para el último camino a distinguir:

```
numero = int(input())
if numero > 0:
    numero = numero + 1
    print ("El siguiente número es", numero)
elif numero < -10:
    print ("El número es demasiado bajo")
else:
    print ("El número no es positivo")
print ("Fin del programa")
```

No existe cláusula *switch/case* en Python, ya que normalmente lo que se puede hacer con este tipo de estructuras se puede hacer también con *if*.

2. Estructuras repetitivas o bucles

2.1. El bucle *while*

La estructura `while` tiene una sintaxis similar a *if*. El bloque de instrucciones asociado se va a ejecutar repetidamente mientras se cumpla la condición indicada. Por ejemplo, este código le pide números al usuario repetidamente hasta que escribe un número negativo o cero:

```
numero = int(input())
while numero > 0:
    print ("Has escrito", numero)
    numero = int(input())
print ("Fin del programa")
```

No hay una estructura *do..while* en Python, que sí existe en otros lenguajes, ya que cualquier bucle *do..while* se puede representar como *while* con algunos pequeños cambios.

2.2. El bucle *for*

Sin embargo, como en otros muchos lenguajes, sí hay una cláusula `for` en Python, y se puede utilizar de varias formas:

- Podemos, por ejemplo, explorar una secuencia determinada de valores. En este ejemplo, la variable `valor` tomará los distintos valores indicados en la lista (2, 4 y 5):

```
for valor in [2, 4, 5]:
    print (valor)
```

- También podemos hacer un uso más "tradicional", e iterar desde un valor inicial hasta uno final. En este ejemplo, mostramos por pantalla los números del 1 al 4 (inclusive). Si sólo especificamos un número en el rango, entonces Python cuenta del 0 hasta ese número (sin incluir dicho número).

```
for valor in range(1, 5):
    print (valor)
```

- En el caso de que sólo queramos indicar un número de repeticiones sin intención de usar el contador, podemos especificar un único valor en `range`:

```
for i in range(4):
    ...
```

- Finalmente, podemos establecer un incremento distinto de 1, especificando un tercer parámetro dentro de la opción *range*. Por ejemplo, aquí contamos del 0 al 100 con un incremento de 10 en 10:

```
for valor in range(0, 101, 10):
    print (valor)
```

> **Ejercicio 1**:
>
> Crea un programa llamado `Notas.py` que le pida al usuario 3 notas, y calcule la nota final según estas reglas:
>
> - Si ninguna nota es mayor que 4, la nota final es 0
> - Si algunas notas son mayores que 4 (pero no todas), la nota final es 2

- Si todas las notas son mayores que 4, la nota final será el 30% de la primera más el 20% de la segunda más el 50% de la tercera

Ejercicio 2:

Crea un programa llamado `Factura.py` que le pida al usuario precios para una factura, hasta que escriba 0. Entonces, el programa debe mostrar el total de la factura con 2 dígitos decimales.

Ejercicio 3:

Crea un programa llamado `MayorMenor.py` que le pida al usuario que introduzca una secuencia de N números positivos (primero el usuario deberá indicar cuántos números va a introducir). Al final del proceso, el programa deberá mostrar por pantalla el valor del número mayor y el menor introducidos por el usuario. Por ejemplo:

```
Dime cuántos números vas a introducir:
3
Escribe 3 números:
3
7
2
El mayor es 7
El menor es 2
```