P03 Condicionales en Python

1.- Sintaxis

Todos los programas que hemos visto hasta ahora se ejecutan línea a línea, no hemos visto ninguna forma de hacer que algunas líneas no se ejecuten.

Consideremos el siguiente problema. Dado un entero X devolver su valor absoluto.
Si X>0 el programa debe devolver X, en otro caso debe devolver –X. Este comportamiento no se puede conseguir empleando un programa secuencial. El programa debe decidir de forma condicional, cual es el siguiente paso a seguir. Aquí es donde entran los condicionales.

```
 x = int(input())
if x > 0:
    print(x)
else:
    print(-x)
```

Este programa usa el condicional `if`. Después del `if` aparece una condición, `(x > 0)` seguida de dos puntos. Después de esto aparece el bloque de instrucciones que se deben ejecutar solo si la condición es cierta (si obtiene un valor de `True`)

Opcionalmente este bloque puede ir seguido de la instrucción `else`, dos puntos y un bloque de instrucciones que se ejecutaran solo si la condición es falsa (si obtiene un valor de `False`). Cada bloque se debe indentar usando espacios.

En resumen, el condicional en Python tiene la siguiente estructura:

```
if condición:
     true-block     conjunto de instrucciones que se deben ejecutar
     si la condición se evalua como True
else:
     false-block     conjunto de instrucciones que se deben ejecutar
     si la condición se evalua como  False
```

La instrucción `else` junto con el bloque de instrucciones se puede omitir si no hay que hacer nada cuando la condición se evalúe como falsa. Por ejemplo, podríamos repetir el programa que calcula el valor absoluto haciendo:

```
x = int(input())
if x < 0:
    x = -x
print(x)
```

En este ejemplo la variable `x` se asigna a `-x` solo si `x < 0` . La diferencia es que la instrucción se ejecuta siempre, por que no esta indentada, y por tanto no pertenece al bloque de cierto.

La indentación es la forma en la que Python separa bloques de código. Todas las instrucciones de un mismo bloque deben estar indentadas de la misma forma. Es decir, deben tener el mismo numero de espacios al principio de la línea. Se recomienda emplear 4 espacios en cada indentación.

Este uso de la indentación hace a Python diferente de la mayor parte de lenguajes, en los que se suelen emplear llaves { } para formar los bloques.

También se puede señalar que Python tiene una función para calcular el valor absoluto

```
 x = int(input())
print(abs(x))
```

2.- Condicionales anidados

En un bloque condicional se puede poner cualquier instrucción de Python, incluyendo otros condicionales. Esta es la forma en que obtenemos condicionales anidados. Los bloques de los condicionales internos se indentan usando el doble de espacios (se recomienda que sean 8). Veamos un ejemplo, dadas las coordenadas de un punto del plano devolver su cuadrante.

```
x = int(input())
y = int(input())
if x > 0:
    if y > 0:
        # x > 0, y > 0
        print("Cuadrante I")
    else:
        # x > 0, y <= 0
        print("Cuadrante IV")
else:
    if y > 0:
        # x <= 0, y > 0
        print("Cuadrante II")
    else:
        # x <= 0, y <= 0
        print("Cuadrante III")
```

3.- Operadores de comparación

Por lo general la condición después de `if` emplea uno o mas de los siguientes operadores:

`<` Menor — La comparación es true si el lado izquierdo es menor que el derecho.

`>` Mayor — La comparación es true si el lado izquierdo es mayor que el derecho.

`<=` Menor o igual.

`>=` Mayor o igual.

`==` Igual.

`!=` Distinto.

Por ejemplo, la condición `x * x < 1000` significa "El valor de la expresión `x * x` es menor que 1000", y la condición `2 * x != y` significa "el doble del valor de la variable `x` no es igual al valor de la variable `y`"

Los operadores de comparación en Python se pueden emplear encadenados como : `a == b == c` o `x <= y >= 10` Esto no es lo habitual en otros lenguajes.

4.- Objetos booleanos

Cuando sumamos dos enteros empleando el operador `+` , como en `2 + 5` , obtenemos un nuevo objeto `7`. Del mismo modo, cuando comparamos dos enteros usando el operador `<`, como en `2 < 5`, tenemos un nuevo objeto `True`:

Se puede comprobar empleando este programa

```
print(2 < 5)
print(2 > 5)
```

Los objetos y son de un tipo especial, llamado. Como cada nombre de tipo se puede emplear para convertir otros objetos en ese tipo. Veamos que se obtiene

```
print(bool(-10))    # True
print(bool(0))      # False – cero es el unico numero falso
print(bool(10))     # True
print(bool(''))     # False – La cadena vacia es la unica falsa
print(bool('abc'))  # True
```

4.1.- Operadores lógicos

En algunos casos necesitaremos comprobar varias condiciones de golpe. Por ejemplo puedes comprobar si un numero es ar usando la condición `n % 2 == 0` (`n` tiene resto `0` al dividirse por `2`). Si necesitamos comprobar que dos números `n` y `m` son los dos pares deberíamos comprobar `n % 2 == 0` y `m % 2 == 0`. para hacer esto los unimos empleando el operador `and` (AND lógico) `n % 2 == 0 and m % 2 == 0`.

Python tiene los operadores lógicos AND, OR y la negación.

El operador `and` es un operador binario que devuelve `True` si y solo si los dos argumentos son `True`

El operador `or` es un operador operador binario que devuelve `True` si al menos uno de los dos argumentos es `True`

El operador `not` es una negación unitaria. Se escribe seguido de algún valor, se obtiene `True` si el valor es `False` y viceversa.

Ejemplo: comprobamos si al menos uno de los dos números termina en 0:

```
a = int(input())
b = int(input())
if a % 10 == 0 or b % 10 == 0:
    print('YES')
else:
    print('NO')
```

Condición de que a sea positivo y b no negativo.

```
if a > 0 and not (b < 0):
```

5.- if-elif-else

Si tienes mas de dos opciones para elegir, el operador condicional se puede emplear con la siguiente estructura `if... elif... else`

Veamos como funciona reescribiendo el ejemplo anterior en el que elegíamos el cuadrante de un punto del plano

```
x = int(input())
y = int(input())
if x > 0 and y > 0:
    print("Quadrant I")
elif x > 0 and y < 0:
    print("Quadrant IV")
elif y > 0:
    print("Quadrant II")
else:
    print("Quadrant III")
```

En este caso las condiciones se van comprobando una por una hasta que se encuentra **la primera** que es cierta y se ejecuta su bloque. Si todas son falsas se ejecuta el bloque else si esta presente.

Ejercicios:

P1 Mínimo de dos números

Dados dos enteros, imprimir el menor de ellos.

P2 Función signo

Dado el entero X imprimir 1 si es positivo, -1 si es negativo y 0 si es 0

Emplear una estructura if elif else

P3 Mínimo de tres números

Dados tres enteros, imprimir el menor de ellos.

P4 Año bisiesto

Dado un año responder si es bisiesto. Si lo es devolver bisiesto, y si no devolver no bisiesto

Las reglas del calendario gregoriano son

Un año es bisiesto si es divisible por 4 y no es divisible por 100

Un año siempre es bisiesto si es divisible por 400

P5 Números iguales

Dados tres enteros determinar cuantos son iguales entre si. El programa debe devolver 3 (si son todos iguales), 2 (si dos de ellos son iguales entre si y el tercero es diferente) o 0 (si todos los números son diferentes)

Problemas de ajedrez:

Consideraremos siempre cada casilla del tablero dada por un número que determina su fila y otro su columna (esto se debe decir cuando se pida la posición)

P6 Movimiento de torre

Las torres mueven horizontal o verticalmente, dadas dos casillas del tablero determinar si una torre puede ir de la primera a la segunda en un solo movimiento.

El programa recibe 4 números, del 1 al 8 especificando las dos casillas y debe responder “Si” o “No” si puede realizarse el movimiento

Como mejora para el programa se puede comprobar si los números se corresponden a una casilla real y en caso contrario terminar el programa y decir que no son validos.

P7 Casillas blancas o negras

D dos casillas del tablero determinar si son del mismo color.

El programa recibe 4 números, del 1 al 8 especificando las dos casillas y debe responder “Si” o “No” si son del mismo color

Como mejora para el programa se puede comprobar si los números se corresponden a una casilla real y en caso contrario terminar el programa y decir que no son validos.

P8 Movimiento de rey

El rey mueve a cualquier casilla adyacente, dadas dos casillas del tablero determinar si el rey puede ir de la primera a la segunda en un solo movimiento.

El programa recibe 4 números, del 1 al 8 especificando las dos casillas y debe responder “Si” o “No” si puede realizarse el movimiento

Como mejora para el programa se puede comprobar si los números se corresponden a una casilla real y en caso contrario terminar el programa y decir que no son validos.

P9 Movimiento de alfil

Los alfiles mueven en diagonal, dadas dos casillas del tablero determinar si un alfil puede ir de la primera a la segunda en un solo movimiento.

El programa recibe 4 números, del 1 al 8 especificando las dos casillas y debe responder “Si” o “No” si puede realizarse el movimiento

Como mejora para el programa se puede comprobar si los números se corresponden a una casilla real y en caso contrario terminar el programa y decir que no son validos.

P10 Movimiento de dama

La dama mueven horizontal, verticalmente, o en diagonal dadas dos casillas del tablero determinar si una dama puede ir de la primera a la segunda en un solo movimiento.

El programa recibe 4 números, del 1 al 8 especificando las dos casillas y debe responder “Si” o “No” si puede realizarse el movimiento

Como mejora para el programa se puede comprobar si los números se corresponden a una casilla real y en caso contrario terminar el programa y decir que no son validos.

P11 Movimiento de caballo. (Opcional, es bastante mas dificil)

Los caballos mueven en L, dadas dos casillas del tablero determinar si un caballo puede ir de la primera a la segunda en un solo movimiento.

El programa recibe 4 números, del 1 al 8 especificando las dos casillas y debe responder “Si” o “No” si puede realizarse el movimiento

Como mejora para el programa se puede comprobar si los números se corresponden a una casilla real y en caso contrario terminar el programa y decir que no son validos.