Puntos Clave

1. Python es compatible con los siguientes operadores lógicos:

- `and` → si ambos operandos son verdaderos, la condición es verdadera, por ejemplo, `(True and True)` es `True`.
- `or` → si alguno de los operandos es verdadero, la condición es verdadera, por ejemplo, `(True or False)` es `True`.
- `not` → devuelve False si el resultado es verdadero y devuelve True si es falso, por ejemplo, `not True` es `False`.

2. 2. Puedes utilizar operadores bit a bit para manipular bits de datos individuales. Los siguientes datos de muestra:

- `x = 15`, which is `0000 1111` en binario.
- `y = 16`, which is `0001 0000` en binario.

Se utilizarán para ilustrar el significado de operadores bit a bit en Python. Analiza los ejemplos a continuación::

- `&` hace un *bit a bit and (y)*, por ejemplo, `x & y = 0`, el cual es `0000 0000` en binario.
- `|` hace un *bit a bit or (o)*, por ejemplo, `x | y = 31`, el cual es `0001 1111` en binario.
- `~` hace un *bit a bit not (no)*, por ejemplo, `~ x = 240`, el cual es `1111 0000` en binario.
- `^` hace un *bit a bit xor*, por ejemplo, `x ^ y = 31`, el cual es `0001 1111` en binario.
- `>>` hace un *desplazamiento bit a bit a la derecha*, por ejemplo, `y >> 1 = 8`, el cual es `0000 1000` en binario.
- `<<` hace un *desplazamiento bit a bit a la izquierda*, por ejemplo, `y << 3 =` , el cual es `1000 0000` en binario.

* `-16` (decimal del complemento a 2 con signo) -- lee más acerca de la operación Complemento a dos.