Bitwise NOT

The **NOT** bitwise operation takes one set of bits, and for each bit returns 0 if the bit is 1, and 1 if the bit is 0.

```java
~ 0 // -1
~ 1 // -2
```

When performing NOT on an integer, each bit of the integer is inverted.

```java
~ 5 // Gives -6

// At the bit level:
//   ~ 0000 0101  (5)
//   = 1111 1010  (-6)
```

If you're unsure why the resulting number is negative in this example, it's because numbers are represented using two's complement. Read up on binary numbers here (/concept/binary-numbers).

See also:

- Binary Numbers (/concept/binary-numbers)
- Bitwise AND (/concept/and)
- Bitwise OR (/concept/or)
- Bitwise XOR (eXclusive OR) (/concept/xor)
- Bit Shifting (/concept/bit-shift)

What's next?

If you're ready to start applying these concepts to some problems, check out our mock coding interview questions (/next).

They mimic a real interview by offering hints when you're stuck or you're missing an optimization.

Try some questions now ➡