Ans. 2.

(i) / and // operators:

The / operator performs floating-point division. It returns the quotient as a floating-point number.

The // operator performs floor division. It returns the quotient as an integer, discarding the decimal part (i.e., performing integer division).

x = 10

y = 3

print(x / y) # Output: 3.3333333333333335 (floating-point division)

print(x // y) # Output: 3 (floor division)

(ii) \*\* and ^ operators:

The \*\* operator is used for exponentiation. It raises a number to the power of another number.

The ^ operator, also known as the bitwise XOR operator, is used for bitwise exclusive OR operation between two integers.

x = 2

y = 3

print(x \*\* y) # Output: 8 (2 raised to the power of 3)

print(x ^ y) # Output: 1 (bitwise XOR operation: 10 XOR 11)

Ans. 3.

The logical operators in Python are:

and: The and operator returns True if both operands are True, otherwise, it returns False. It performs a logical conjunction.

or: The or operator returns True if at least one of the operands is True, otherwise, it returns False. It performs a logical disjunction.

not: The not operator negates the logical value of its operand. It returns True if the operand is False, and False if the operand is True.

Ans. 4.

The right shift operator (>> ) shifts the bits of a number to the right by a specified number of positions. The rightmost bits are discarded, and the leftmost bits are filled with the same value as the sign bit (for signed numbers) or with zeros (for unsigned numbers). Here's an example:

num = 10 # Binary: 1010

result = num >> 1

print(result) # Output: 5

The left shift operator (<< ) shifts the bits of a number to the left by a specified number of positions. The rightmost bits are filled with zeros. Here's an example:

num = 5 # Binary: 0101

result = num << 2

print(result) # Output: 20