**Operators in Python: -**

1. **Arithmetic operators**
2. **Assignment operators**
3. **Comparison operators**
4. **Logical operators**
5. **Identity operators**
6. **Membership operators**
7. **Bitwise operators**
8. **Arithmetic operators:** These operators are used with numeric values to perform mathematical operations.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operator | Name | Example | Note | Variables: - a=10 and b=20 | Output |
| + | Addition | a + b |  | print(a+b) | 30 |
| - | Subtraction | a - b |  | print(a-b) | -10 |
| \* | Multiplication | a \* b |  | print(a\*b) | 200 |
| / | Division | a / b | It returns the division operation’s quotient as a floating-point number and hence known as Float division | print(a/b) | 0.5 |
| % | Modulus | a % b | It's not percent, It's modulo operation (%) which returns the remainder of dividing the left-hand operand by right-hand operand | print(a%b) | 10 |
| \*\* | Exponentiation | a \*\* b | a^b or POWER (a, b) | print(a\*\*b) | 100000000000000000000 |
| // | Floor division | a // b | It can be used to divide two numbers and get the result in which the quotient is rounded down to the nearest integer value, it's also known as Integer division. | print(a//b) | 0 |

1. **Assignment operators:** These operators are used to assign values to variables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operator | Example | Same As | Example with complete Python code | Output |
| = | a = 5 | a = 10 | a=10; print(a) | 10 |
| += | a += 3 | a = a + 3 | a=3; a+=3; print(a) | 6 |
| -= | a -= 3 | a = a - 3 | a=10; a = a - 3; print(a) | 7 |
| \*= | a \*= 3 | a = a \* 3 | a=10; a = a \* 3; print(a) | 30 |
| /= | a /= 3 | a = a / 3 | a=10; a = a / 3; print(a) | 3.333333333 |
| %= | a %= 3 | a = a % 3 | a=10; a = a % 3; print(a) | 1 |
| //= | a //= 3 | a = a // 3 | a=10; a = a // 3; print(a) | 3 |
| \*\*= | a \*\*= 3 | a = a \*\* 3 | a=10; a = a \*\* 3; print(a) | 1000 |
| &= | a &= 3 | a = a & 3 | a=10; a= a & 3; print(a) | 2 |
| |= | a |= 3 | a = a | 3 | a=10; a = a | 3; print(a) | 11 |
| ^= | a ^= 3 | a = a ^ 3 | a=10; a = a ^ 3; print(a) | 9 |
| >>= | a >>= 3 | a = a >> 3 | a=10; a = a >> 3; print(a) | 1 |
| <<= | a <<= 3 | a = a << 3 | a=10; a = a << 3; print(a) | 80 |
| := | print(a := 3) | a = 3 | a=10; a = 3; print(a) | 3 |
| print(a) |  |  |

1. **Comparison operators:** These operators are used to compare two values.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operator** | **Name** | **Example** | **Example with Python Code** | **Output** |
| == | Equal | a == b | a=10; b=20; print (a==b) | FALSE |
| != | Not equal | a != b | a=10; b=20; print(a != b) | TRUE |
| > | Greater than | a > b | a=10; b=20; print(a > b) | FALSE |
| < | Less than | a < b | a=10; b=20; print(a < b) | TRUE |
| >= | Greater than or equal to | a >= b | a=10; b=20; print(a >= b) | FALSE |
| <= | Less than or equal to | a <= b | a=10; b=20; print(a <= b) | TRUE |

1. **Logical Operators:** These operators are used to combining conditional statements.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Example with Python Code** | **Output** |
| and | Returns True if both statements are true | x < 5 and  x < 10 | print (x < 5 and x < 10); | FALSE |
| or | Returns True if one of the statements is true | x < 5 or x < 4 | print (x < 5 or x < 10); | FALSE |
| not | Reverse the result, returns False if the result is true | not(x < 5 and x < 10) | print (not(x < 5 and x < 10)); | TRUE |

1. **Identity Operators:** These operators are used to compare objects, not if they are equal, but if they are actually the same object, with the same memory location.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operator | Description | Example | Example with Python Code | Output |
| is | Returns True if both variables are the same object | x is y | x=10; y=5; print(x is y) | FALSE |
| is not | Returns True if both variables are not the same object | x is not y | x=10; y=5; print(x is not y) | TRUE |

1. **Membership Operators:** These operators are used to test if a sequence is presented in an object.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Operator | Description | Example | Example with Python Code | Output |
| in | Returns True if a sequence with the specified value is present in the object | x in y | a=[1,4,8,0]; print(1 in a) | TRUE |
| not in | Returns True if a sequence with the specified value is not present in the object | x not in y | a=[1,4,8,0];print(10 not in a) | TRUE |

1. **Bitwise Operators:** These operators are used to compare (binary) numbers.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operator | Name | Description | Example | Example with Python Code | Output |
| & | AND | Sets each bit to 1 if both bits are 1 | x & y | print(6&4) | 4 |
| | | OR | Sets each bit to 1 if one of two bits is 1 | x | y | print(6|4) | 6 |
| ^ | XOR | Sets each bit to 1 if only one of two bits is 1 | x ^ y | print(15^32) | 47 |
| ~ | NOT | Inverts all the bits | ~x | a= 5 print('~a:', ~a) | -6 |
| << | Zero fill left shift | Shift left by pushing zeros in from the right and let the leftmost bits fall off | x << 2 | print (5<<3) | 40 |
| >> | Signed right shift | Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off | x >> 2 | print (10>>3) | 1 |

**Understand Binary calculation with the help of TRUE and FALSE**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X | Y | X AND Y | X OR Y | | X XOR Y | 0= FALSE  1= TRUE |
| TRUE | TRUE | TRUE | TRUE | FALSE | |  |
| TRUE | FALSE | FALSE | TRUE | TRUE | |  |
| FALSE | TRUE | FALSE | TRUE | TRUE | |  |
| FALSE | FALSE | FALSE | FALSE | FALSE | |  |

**Below are the simple way to convert numeric value into Binary using Formula- 32 16 8 4 2 1**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Operator | AND (&) | |  |  |  |  |  |  |
| Formula | **16** | **8** | | **4** | **2** | **1** |  |  |
|  |  |  | |  |  |  |  |  |
| Binary number of 6 |  |  | | 1 | 1 | 0 |  |  |
| Binary number of 4 |  |  | | 1 | 0 | 0 | **ANSWER** |  |
| Binary of 6 and 4 in Binary |  |  | | 1 | 0 | 0 | 4 |  |
|  |  |  | |  |  |  |  |  |
| Binary Number of 23 | 1 | 0 | | 1 | 1 | 1 |  |  |
| Binary Number of 17 | 1 | 0 | | 0 | 0 | 1 | **ANSWER** |  |
| Binary of 23 and 17 | 1 | 0 | | 0 | 0 | 1 | 17 |  |
|  |  |  | |  |  |  |  |  |
| Formula | **32** | **16** | | **8** | **4** | **2** | **1** |  |
| Binary number of 39 | 1 | 0 | | 0 | 1 | 1 | 1 |  |
| Binary number of 3 | 0 | 0 | | 0 | 0 | 1 | 1 | **ANSWER** |
| Binary of 39 and 3 in Binary | 0 | 0 | | 0 | 0 | 1 | 1 | 3 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Operator | OR (|) | | |  |  |  |  |  |  |
| Formula | **16** | **8** | | | **4** | **2** | **1** |  |  |
| Binary number of 6 |  |  | | | 1 | 1 | 0 |  |  |
| Binary number of 4 |  |  | | | 1 | 0 | 0 | **ANSWER** |  |
| Binary of 6 and 4 Using OR Operator |  |  | | | 1 | 1 | 0 | 6 |  |
|  |  |  | | |  |  |  |  |  |
| Binary Number of 23 | 1 | 0 | | | 1 | 1 | 1 |  |  |
| Binary Number of 17 | 1 | 0 | | | 0 | 0 | 1 | **ANSWER** |  |
| Binary of 23 and 17 Using OR Operator | 1 | 0 | | | 1 | 1 | 1 | 23 |  |
|  |  |  | | |  |  |  |  |  |
| Formula | **32** | **16** | | | **8** | **4** | **2** | **1** |  |
| Binary number of 39 | 1 | 0 | | | 0 | 1 | 1 | 1 |  |
| Binary number of 3 | 0 | 0 | | | 0 | 0 | 1 | 1 | **ANSWER** |
| Binary of 39 and 3 in Binary Using OR Operator | 1 | 0 | | | 0 | 1 | 1 | 1 | 39 |
|  |  |  | | |  |  |  |  |  |
| Binary number of 21 | 1 | 0 | | | 1 | 0 | 1 |  |  |
| Binary Number of 13 | 0 | 1 | | | 1 | 0 | 1 |  |  |
| Binary of 21 and 23 Using OR Operator | 1 | 1 | | | 1 | 0 | 1 | 29 |  |
| Operator | **XOR (^)** | |  | |  |  |  |  | **ANSWER** |
| Formula | **32** | **16** | | | **8** | **4** | **2** | **1** |  |
|  |  |  | | |  |  |  |  |  |
| Binary number of 15 |  |  | | | 1 | 1 | 1 | 1 |  |
| Binary number of 32 | 1 | 0 | | | 0 | 0 | 0 | 0 |  |
| Binary of 15 and 32 Using XOR Operator | 1 | 0 | | | 1 | 1 | 1 | 1 | 47 |
|  |  |  | | |  |  |  |  |  |
| Binary Number of 13 |  |  | | | 1 | 1 | 0 | 1 |  |
| Binary Number of 17 |  | 1 | | | 0 | 0 | 0 | 1 |  |
| Binary of 13 and 17 Using XOR Operator |  | 1 | | | 1 | 1 | 0 | 0 | 28 |
|  |  |  | | |  |  |  |  |  |
| Binary Number of 21 |  | 1 | | | 0 | 1 | 0 | 1 |  |
| Binary Number of 26 |  | 1 | | | 1 | 0 | 1 | 0 |  |
| Binary of 21 and 26 Using XOR Operator |  | 0 | | | 1 | 1 | 1 | 1 | 15 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Formula | 32 | 16 | 8 | 4 | 2 | 1 |  |  |  |  |
| Binary of 5 |  |  |  |  | 1 | 0 | 1 | **Answer** |  |  | **Answer** |
| 5<<3 | **Left shift** | 1 | 0 | 1 | 0 | 0 | 0 | 40 |  |  |  |
| 5<<2 |  |  | 1 | 0 | 1 | 0 | 0 | 20 |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
| Binary of 10 |  |  |  | 1 | 0 | 1 | 0 |  |  |  |  |
| 10>>2 | **Right Shift** |  |  | 0 | 0 | 1 | 0 | 1 | 0 |  | 2 |
| 10>>3 |  |  |  | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |