Python Operators - Interview Questions with Answers

1. What is an operator & purpose of the operator?
An operator is a symbol that performs operations on variables and values (e.g., +, -, *, /).

2. What is an operand & purpose of operands?
Operands are the values or variables on which the operator performs operations. Example: In `a + b`, `a` and `b` are operands.

3. What are the main types of operators in Python?

- Arithmetic Operators
- Relational/Comparison Operators
- Logical Operators
- Assignment Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

4. What are the arithmetic operators and explain about them?

- `+` Addition
- `-` Subtraction
- `*` Multiplication
- `/` Division
- `//` Floor Division
- `%` Modulus
- `**` Exponentiation

5. What are the relational or comparison operators?

- `==` Equal
- `!=` Not Equal
- `>` Greater than
- `<` Less than
- `>=` Greater than or equal to
- `<=` Less than or equal to These return `True` or `False`.

6. What are the logical operators?

- `and`: True if both operands are true.
- `or`: True if at least one operand is true.
- `not`: Reverses the result.

7. What is the difference between / and // in Python?

- `/` returns float division.
- `//` returns floor (integer) division.

8. How are the logical operators and, or, and not used?
Example:

```
x = 10
y = 5
print(x > 5 and y < 10)  # True
print(x > 20 or y < 10)  # True
print(not(x > 5))        # False
```

9. Difference between == and is?

- `==` checks value equality.
- `is` checks memory location (identity).

10. How do you overload operators in Python?
By defining special methods like `__add__()`, `__eq__()`, etc., in a class.

11. What is operator overloading? Give an example.
Allows the same operator to have different meanings based on context.

```
class Point:
    def __init__(self, x):
        self.x = x
    def __add__(self, other):
        return Point(self.x + other.x)
```

12. Purpose of `in` and `not in` operators?
They check membership in sequences. Example:

```
print('a' in 'apple')  # True
print(3 not in [1,2,3])  # False
```

13. Purpose of `is` and `is not`?
Check whether two references point to the same object.

```
a = [1,2]
b = a
print(a is b)  # True
```

14. Differences between * and ** operators?

- `*` is multiplication; `**` is exponentiation.

15. Use of * and ** for packing/unpacking?

- `*args` packs variable positional arguments.
- `**kwargs` packs variable keyword arguments.

def func(*args, **kwargs): pass

16. What is operator precedence in Python?

Determines the order in which operations are performed. Example: `2 + 3 * 4` evaluates as `2 + (3 * 4)`.

17. How can you compare two lists in Python?

Use `==` to compare values, `is` to compare identities.

```
[1,2] == [1,2]  # True
[1,2] is [1,2]  # False
```

18. What are the membership operators?

- `in`: Checks if a value exists in a sequence
- `not in`: Checks if a value does not exist Example: `'a' in 'apple'` → True

19. What are identity operators?

- `is`: True if both variables refer to the same object
- `is not`: True if they refer to different objects Example:

```
a = [1,2]; b = a; print(a is b)  # True
```