Operators in Python

1. Assignment Operators

Assignment operators are used to assign values to variables. The simplest one is `=` which assigns the value on the right to the variable on the left. There are also compound assignment operators that combine arithmetic operations with assignment.

Common Assignment Operators:

- `=` : Assigns value on the right to the variable on the left.
- `+=` : Adds right operand to the left operand and assigns the result to the left operand.
- `-=` : Subtracts the right operand from the left operand and assigns the result to the left operand.
- `*=` : Multiplies the left operand by the right operand and assigns the result to the left operand.
- `/=` : Divides the left operand by the right operand and assigns the result to the left operand.
- `%=` : Takes modulus of left operand by right operand and assigns the result to the left operand.

Examples:

```
x = 5  # Assigns 5 to x
x += 3  # Equivalent to x
x -= 2  # Equivalent to x
x *= 4  # Equivalent to x
x /= 6  # Equivalent to x
```

2. Comparison Operators

Comparison operators are used to compare two values. They return either `True` or `False` depending on the condition.

Common Comparison Operators:

- `==` : Checks if two values are equal.
- `!=` : Checks if two values are not equal.
- `>` : Checks if the left operand is greater than the right operand.
- `<` : Checks if the left operand is less than the right operand.
- `>=` : Checks if the left operand is greater than or equal to the right operand.

- <= : Checks if the left operand is less than or equal to the right operand.

Examples:

```
a = 10
b = 20

print(a == b)  # Output:
print(a != b)  # Output:
print(a > b)  # Output: F
print(a < b)  # Output: T
print(a >= 10)  # Output:
print(b <= 25)  # Output:
```

3. Logical Operators

Logical operators are used to combine conditional statements. They evaluate expressions and return either `True` or `False`.

Common Logical Operators:

- `and` : Returns `True` if both conditions are true.
- `or` : Returns `True` if at least one condition is true.
- `not` : Reverses the logical state of its operand (True becomes False, and vice versa).

Examples:

```
x = 5
y = 10
z = 15

# and operator
print(x > 0 and y > 5)  #

# or operator
print(x > 10 or z > 10)

# not operator
print(not(x > 10))  # Out
```

4. Membership Operators

Membership operators test for membership within a sequence, such as a list, string, or tuple. They return `True` or `False` based on whether the value is found in the sequence.

Membership Operators:

- `in` : Returns `True` if the specified value is found in the sequence.
- `not in` : Returns `True` if the specified value is not found in the sequence.

Examples:

```
my_list = [1, 2, 3, 4, 5]
my_string = "Python"

print(3 in my_list)  # Ou
print(6 not in my_list)
print("P" in my_string)
print("z" not in my_strin
```

5. Bitwise Operators

Bitwise operators perform operations on binary representations of integers. These operators are useful for low-level programming tasks like working with bits and bytes.

Common Bitwise Operators:

- `&` : Bitwise AND (sets each bit to 1 if both bits are 1).
- `|` : Bitwise OR (sets each bit to 1 if one of the bits is 1).
- `^` : Bitwise XOR (sets each bit to 1 if only one of the bits is 1).
- `~` : Bitwise NOT (inverts all the bits).
- `<<` : Left shift (shifts bits to the

- `<<` : Left shift (shifts bits to the left by a specified number of positions).
- `>>` : Right shift (shifts bits to the right by a specified number of positions).

Examples:

```
a = 5  # In binary: 101
b = 3  # In binary: 011

# Bitwise AND
print(a & b)  # Output: 1

# Bitwise OR
print(a | b)  # Output: 7

# Bitwise XOR
print(a ^ b)  # Output: 6

# Bitwise NOT
print(~a)  # Output: -6 (

# Left shift
print(a << 1)  # Output:

# Right shift
print(a >> 1)  # Output:
```

6. Arithmetic Operators (Already Covered in Chapter 2)

Python supports basic arithmetic operations like addition, subtraction, multiplication, division, and more.

Common Operators:

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

Examples:

```
a = 10
b = 3
print(a + b)  # Output: 1
print(a - b)  # Output: 7
print(a * b)  # Output: 3
print(a / b)  # Output: 3
print(a // b)  # Output:
print(a % b)  # Output: 1
print(a ** b)  # Output:
```

Homework

1. **Logical Operator Practice**: Write a Python program that takes two numbers as input from the user and checks if:
    - Both numbers are greater than 10 (using `and`).
    - At least one of the numbers is less than 5 (using `or`).
    - The first number is not greater than the second (using `not`).
2. **Comparison Operator Challenge**: Create a Python program that asks the user for their age and prints:
    - "You are an adult" if the age is greater than or equal to 18.
    - "You are a minor" if the age is less than 18.
    - Use `>=` and `<` comparison operators.

3. **Membership Operator Exercise**: Write a Python program that:
    - Takes a string as input from the user.
    - Checks if the letter 'a' is in the string (using `in`).
    - Checks if the string doesn't contain the word "Python" (using `not in`).
4. **Bitwise Operator Task**: Given two integers, write a Python program that:
    - Prints the result of `a & b`, `a | b`, and `a ^ b`.
    - Shifts the bits of `a` two positions to the left (`a << 2`).
    - Shifts the bits of `b` one position to the right (`b >> 1`).