```
# Operators
# Operators are used to perform operations on operands (variables or values).
"""
```

1. Arithmetic Operators
   # Used to perform Arithmetic Operations
   + => Addition
   - => Subtraction
   * => Multiplication
   / => Division => It will always result in a float value.
   // => Floor Division
   % (Modulus) => Remainder
   ** => Exponential

2. Assignment Operators
   To Assign values to variables
   =
   +=
   -=
   *=
   /=
   //=
   %=
   **=

3. Relational / Comparison Operators
   They are used to compare two operands.
   They always result in a boolean value.
   > -> Greater than ->
        Checks if left operand is greater than the right operand
   < -> Less than
        Checks if left operand is less than the right operand
   >= -> Greter than or equal to
         Checks if left operand is greater than or equal to
         the right operand
   <= -> Less than or equal to
         Checks if left operand is less than or equal to
         the right operand
   == -> Equal to
         Checks if both left operand and the right operand are equal.
   != -> Not Equal to
         Checks if both left operand and the right operand are not equal.

4. Logical Operators
   They are used to combine two or more conditions
   and, or, not
5. Bitwise Operators
   They are used to perform bitwise operations on operands.
   & -> Bitwise and
   | -> Bitwise or
   ~ -> Complement (Bitwise not)
   ^ -> Exor
   << -> Left shift
   >> -> Right shift

6. Membership Operators
   They are used to verify if an object is an item in the collection.
   in, not in
7. Identity Operators
   They are used to verify if both the objects refer to the same memory location.

```
   is, is not
"""
# Arithmetic Operators
a = 10
b = 20
c = a + b
print(c) # 30
c = b - a
print(c) # 10
c = a * b
print(c) # 200
c = b / a
print(c) # 2.0
c = b // a
print(500 / 23)
print(500 // 23)
print(c) # 2
c = b % a
print(c) # 0

from math import floor, ceil



print(floor(1.99))
print(floor(1.55))
print(floor(1.01))

print(ceil(1.99))
print(ceil(1.55))
print(ceil(1.01))


print(floor(69.56))
print(floor(69.89))
print(floor(45.01))

print(ceil(69.56))
print(ceil(69.89))
print(ceil(45.01))

print(round(78.5))
print(round(78.6))

print(round(78.55))
print(round(78.80))

print(round(78.49))
print(round(78.20))


print(b % a)
print(5 % 2)
print(12 % 5)

print(2 ** 3) # 2 * 2 * 2 => 8
print(5 ** 3) # 5 * 5 * 5 => 125
```

```
# Assignment Operators:
a = 10
print(a)

# 1. They improve the code readability.
# 2. Easy to write code.
a += 2 # a = a + 2
print(a) # 12
a -= 10 # a = a - 10
print(a) # 2
a *= 10 # a = a * 10
print(a) # 20
a /= 2 # a = a / 2
print(a) # 10.0
a = 100
a //= 23 # a = a // 23
print(a) # 4
a %= 4 # a = a % 4
print(a) # 0
a = 5
a **= 2 # a = a ** 2
print(a) # 25


# 3. Relational Operators / Comparison Operators:
print(10 < 20) # True
print(10 < 5) # False
print(50 > 10) # True
print(100 > 1000) # False
print(5 >= 5) # True
print(10 >= 5) # True
print(1 >= 5) # False
print(6 <= 6) # True
print(6 <= 10) # True
print(100 <= 50) # False
print(100 == 100) # True
print(30 == 40) # False
print(9 != 9) # False
print(9 != 7) # True


print("Logical Operators--------------")
a = 10
b = 20
print((a > 0) and (b < 100)) # True and True => True
print((a > 0) and (b > 100)) # True and False => False
print((a < 0) and (b < 100)) # False and True => False
print((a < 0) and (b > 100)) # False and False => False

print((a > 0) or (b < 100)) # True or True => True
print((a > 0) or (b > 100)) # True or False => True
print((a < 0) or (b < 100)) # False or True => True
print((a < 0) or (b > 100)) # False or False => False

print(not (a == 10)) # not True => False
print(not (a == 100)) # not False => True


# Number Systems:
```

```
"""
1. Binary (Bi - 2 => 0, 1) => Base 2
2. Octal (Octa - 8 => 0 - 7) => Base 8
3. Decimal (Deci - 10 => 0 - 9) => Base 10
4. Hexa Decimal (Hexa - 6, Deci - 10 (10 + 6) => (0 - 9, A = 10,
                 B = 11, C = 12, D = 13, E = 14, F = 15)) => Base 16
"""
print(25 & 57)
print(25 | 57)
print(25 ^ 57)

print(49 & 79)
print(49 | 79)
print(49 ^ 79)


print(9 << 1) # 18 (9 * (2 ** 1)) => (9 * 2)
print(9 << 2) # 36 (9 * (2 ** 2)) => (9 * 4)

# Left Shift
# (n >> x) => (n * (2 ** x))


print(9 >> 1) # 4 (9 // (2 ** 1)) = (9 // 2)
print(9 >> 2) # 2 (2 // (2 ** 2)) = (9 // 4)

# Right Shift
# (n >> x) => (n // (2 ** x))

print(19 << 1) # 38
print(19 << 2) # 76


print(19 >> 1)
print(19 >> 2)

print(~9) # -10

print(~-11) # +10

# ~n => -(n + 1)

# Membership Operators
# in, not in

# Identity Operators
# is, is not

# Walrus Operator (>= 3.8)
# :=


# (1827382732 // 10)
# (1827382732 // 100)
# (1827382732 // 1000)
# (1827382732 // 10000)

# (1827382732 % 10)
# (1827382732 % 100)
```

```
# (1827382732 % 1000)
# (1827382732 % 10000)
```