# Operators Operands

## Operators and Operands in Python

In Python, **operators** are symbols used to perform operations on variables and values (known as **operands**). Python supports various types of operators, including **arithmetic**, **comparison**, **logical**, **assignment**, **boolean**, and **membership** operators.

## 1. Arithmetic Operators

- + : Addition
- - : Subtraction
- * : Multiplication
- / : Division (float)
- // : Integer Division (floor division)
- % : Modulus (remainder)
- ** : Exponentiation (power)

```
x = 10
y = 3

addition = x + y          # 13
subtraction = x - y       # 7
multiplication = x * y    # 30
division = x / y          # 3.333...
integer_division = x // y # 3
modulus = x % y           # 1
power = x ** y            # 1000
```

## 2. Comparison Operators

- == : Equal to
- != : Not equal to
- > : Greater than
- < : Less than
- >= : Greater than or equal to
- <= : Less than or equal to

```
x = 10
y = 5

print(x == y)   # False
print(x != y)   # True
print(x > y)    # True
print(x < y)    # False
print(x >= y)   # True
print(x <= y)   # False
```

## 3. Logical Operators

- and : Logical AND
- or : Logical OR
- not : Logical NOT

```
x = 10
y = 5

print(x > 3 and y < 10)   # True, both
conditions are True
print(x > 10 or y < 10)   # True, one
condition is True
print(not(x > 5))         # False, since x >
5 is True, `not` makes it False
```

## 4. Assignment Operators

- = : Assign value
- += : Add and assign
- -= : Subtract and assign
- *= : Multiply and assign
- /= : Divide and assign
- //= : Floor divide and assign
- %= : Modulus and assign
- **= : Exponent and assign

```
x = 10

x += 5   # Equivalent to: x = x + 5 (x becomes 15)
x -= 3   # Equivalent to: x = x - 3 (x becomes 12)
x *= 2   # Equivalent to: x = x * 2 (x becomes 24)
x /= 4   # Equivalent to: x = x / 4 (x becomes 6.0)
```

## 5. Boolean Operators

- True : Boolean True value
- False : Boolean False value

```
is_sunny = True
is_raining = False

print(is_sunny)     # Output: True
print(is_raining)   # Output: False
```

6. Membership Operators

- in : Returns True if a value is found in a sequence
- not in : Returns True if a value is not found in a sequence

```
fruit = "apple"
print("a" in fruit)       # True, 'a' is in "apple"
print("b" not in fruit)   # True, 'b' is not in
"apple"
```

# Day 7 : Loops – Introduction

## Conditional Statements in Python

Conditional statements allow you to execute specific blocks of code based on certain conditions. These conditions typically involve comparison or logical operators, which return either True or False.

### 1. if Statement

The if statement allows you to execute a block of code only if a condition is True.

```
x = 10
if x > 5:
    print("x is greater than 5")
```

**Explanation**: In this example, the condition x > 5 evaluates to True, so the code inside the if block runs, printing the message.

### 2. `else` Statement

The else statement is used to run a block of code if the condition in the if statement is False

```
x = 3
if x > 5:
    print("x is greater than 5")
else:
    print("x is less than or equal to 5")
```

**Explanation**: Since x > 5 is False, the else block is executed, printing "x is less than or equal to 5."

### 3. `elif` Statement

The elif (short for "else if") statement allows you to check multiple conditions. If the if condition is False, Python checks the elif condition.

```
x = 5
if x > 5:
    print("x is greater than 5")
elif x == 5:
    print("x is equal to 5")
else:
    print("x is less than 5")
```

**Explanation**: The elif statement checks if x == 5. Since this condition is True, the corresponding block of code runs, and "x is equal to 5" is printed.

### 4. Nested Conditionals

You can nest if, elif, and else statements within one another to check multiple conditions in a hierarchical manner.

**Explanation**: In this example, the outer if checks whether x > 5. Since this is True, the inner if checks if y < 10. Since both conditions are True, "x is greater than 5 and y is less than 10" is printed.

```python
x = 10
y = 5

if x > 5:
    if y < 10:
        print("x is greater than 5 and y is less than 10")
    else:
        print("y is 10 or greater")
else:
    print("x is 5 or less")
```

Summary:

- **if:** Executes a block of code if the condition is True.
- **else:** Executes a block of code if the condition in the if statement is False.
- **elif**: Checks additional conditions if the first if condition is False.
- **Nested conditionals**: Allows combining multiple conditions within one another.