Learn Python Programming from Scratch

Topic: Basic Operators in Python

1. What are Operators?

Operators are special symbols in Python that perform operations on variables and values (operands). They are the building blocks for creating expressions and performing computations in your programs.

Think of operators as the "action words" of programming:

- They tell Python what operation to perform
- They work with one or more operands (values or variables)
- They return a result based on the operation

2. Why Operators are Important

Operators are essential because they allow you to:

- Perform mathematical calculations
- Compare values and make decisions
- Assign and modify variable values
- Combine logical conditions
- Manipulate data efficiently

3. Types of Operators in Python

Python provides several categories of operators:

Arithmetic Operators: Perform mathematical operations

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

Comparison Operators: Compare values and return True/False

- `==` (Equal), `!=` (Not Equal), `>` (Greater), `<` (Less)
- `>=` (Greater or Equal), `<=` (Less or Equal)

Assignment Operators: Assign values to variables

- `=` (Assign), `+=` (Add and Assign), `-=` (Subtract and Assign)
- `*=`, `/=`, `//=`, `%=`, `**=` (Operation and Assign)

Logical Operators: Combine boolean expressions

- `and`, `or`, `not`

Let's explore each type with detailed examples!

In [1]:
```python
# Arithmetic Operators - Basic Mathematical Operations

print("=== ARITHMETIC OPERATORS ===")
print("Performing basic mathematical operations\n")

# Sample values for demonstrations
a = 15
b = 4

print(f"Given: a = {a}, b = {b}")
print("-" * 35)

# Addition (+)
result_add = a + b
print(f"Addition:        {a} + {b} = {result_add}")

# Subtraction (-)
result_sub = a - b
print(f"Subtraction:     {a} - {b} = {result_sub}")

# Multiplication (*)
result_mul = a * b
print(f"Multiplication:  {a} * {b} = {result_mul}")

# Division (/) - Always returns float
result_div = a / b
print(f"Division:        {a} / {b} = {result_div}")

# Floor Division (//) - Returns integer part only
result_floor = a // b
print(f"Floor Division:  {a} // {b} = {result_floor}")

# Modulus (%) - Returns remainder
result_mod = a % b
print(f"Modulus:         {a} % {b} = {result_mod}")

# Exponentiation (**) - Power operation
result_exp = a ** b
print(f"Exponentiation:  {a} ** {b} = {result_exp}")

print("\n" + "="*45)
```

```
=== ARITHMETIC OPERATORS ===
Performing basic mathematical operations

Given: a = 15, b = 4
-----------------------------------
Addition:        15 + 4 = 19
Subtraction:     15 - 4 = 11
Multiplication:  15 * 4 = 60
Division:        15 / 4 = 3.75
Floor Division:  15 // 4 = 3
Modulus:         15 % 4 = 3
Exponentiation:  15 ** 4 = 50625

=============================================
```

4. Comparison Operators

Comparison operators compare two values and return a boolean result (`True` or `False`). They are essential for making decisions in your programs.

In [2]:
```
# Comparison Operators - Comparing Values

print("=== COMPARISON OPERATORS ===")
print("Comparing values and returning True/False\n")

# Sample values for comparison
x = 10
y = 20
z = 10

print(f"Given: x = {x}, y = {y}, z = {z}")
print("-" * 40)

# Equal to (==)
print(f"Equal to:              x == y  →  {x} == {y}  = {x == y}")
print(f"Equal to:              x == z  →  {x} == {z}  = {x == z}")

# Not equal to (!=)
print(f"Not equal to:          x != y  →  {x} != {y}  = {x != y}")
print(f"Not equal to:          x != z  →  {x} != {z}  = {x != z}")

# Greater than (>)
print(f"Greater than:          x > y   →  {x} > {y}   = {x > y}")
print(f"Greater than:          y > x   →  {y} > {x}   = {y > x}")

# Less than (<)
print(f"Less than:             x < y   →  {x} < {y}   = {x < y}")
print(f"Less than:             y < x   →  {y} < {x}   = {y < x}")

# Greater than or equal (>=)
print(f"Greater or equal:      x >= z  →  {x} >= {z}  = {x >= z}")
print(f"Greater or equal:      x >= y  →  {x} >= {y}  = {x >= y}")

# Less than or equal (<=)
print(f"Less or equal:         x <= z  →  {x} <= {z}  = {x <= z}")
print(f"Less or equal:         x <= y  →  {x} <= {y}  = {x <= y}")

print("\n" + "="*50)

# Comparing different data types
print("\n=== COMPARING DIFFERENT DATA TYPES ===")
num = 5
text = "5"
boolean = True

print(f"Number vs String:      {num} == '{text}'     = {num == text}")
print(f"Number vs Boolean:     {num} == {boolean}    = {num == boolean}")
print(f"Boolean as number:     {boolean} == 1        = {boolean == 1}")
print(f"Boolean as number:     False == 0       = {False == 0}")
```

```
=== COMPARISON OPERATORS ===
Comparing values and returning True/False

Given: x = 10, y = 20, z = 10
----------------------------------------
Equal to:              x == y  →  10 == 20  = False
Equal to:              x == z  →  10 == 10  = True
Not equal to:          x != y  →  10 != 20  = True
Not equal to:          x != z  →  10 != 10  = False
Greater than:          x > y   →  10 > 20   = False
Greater than:          y > x   →  20 > 10   = True
Less than:             x < y   →  10 < 20   = True
Less than:             y < x   →  20 < 10   = False
Greater or equal:      x >= z  →  10 >= 10  = True
Greater or equal:      x >= y  →  10 >= 20  = False
Less or equal:         x <= z  →  10 <= 10  = True
Less or equal:         x <= y  →  10 <= 20  = True

==================================================

=== COMPARING DIFFERENT DATA TYPES ===
Number vs String:      5 == '5'     = False
Number vs Boolean:     5 == True    = False
Boolean as number:     True == 1        = True
Boolean as number:     False == 0       = True
```

5. Assignment Operators

Assignment operators are used to assign values to variables. They provide shortcuts for performing operations and assignments in one step.

In [3]:
```
# Assignment Operators - Assigning and Modifying Values

print("=== ASSIGNMENT OPERATORS ===")
print("Assigning values and performing operations in one step\n")

# Basic assignment (=)
num = 5
print(f"Initial assignment:    num = {num}")
print("-" * 45)

# Add and assign (+=)
num += 3    # Equivalent to: num = num + 3
print(f"Add and assign (+=):   num += 3   →  num = {num}")

# Subtract and assign (-=)
num -= 2    # Equivalent to: num = num - 2
print(f"Subtract assign (-=):  num -= 2   →  num = {num}")

# Multiply and assign (*=)
num *= 4    # Equivalent to: num = num * 4
print(f"Multiply assign (*=):  num *= 4   →  num = {num}")

# Divide and assign (/=)
num /= 3    # Equivalent to: num = num / 3
print(f"Divide assign (/=):    num /= 3   →  num = {num}")

# Reset to integer for remaining examples
```

```
num = int(num)
print(f"Reset to integer:      num = {num}")

# Floor divide and assign (//=)
num //= 2   # Equivalent to: num = num // 2
print(f"Floor div assign (//=):num //= 2  →  num = {num}")

# Modulus and assign (%=)
num %= 5    # Equivalent to: num = num % 5
print(f"Modulus assign (%=):   num %= 5   →  num = {num}")

# Power and assign (**=)
num **= 2   # Equivalent to: num = num ** 2
print(f"Power assign (**=):    num **= 2  →  num = {num}")

print("\n" + "="*50)

# Multiple assignment examples
print("\n=== MULTIPLE ASSIGNMENT EXAMPLES ===")
a, b, c = 1, 2.5, "Hello"
print(f"Multiple assignment:   a, b, c = 1, 2.5, 'Hello'")
print(f"Results:               a = {a}, b = {b}, c = '{c}'")

# Chain assignment
x = y = z = 10
print(f"Chain assignment:      x = y = z = 10")
print(f"Results:               x = {x}, y = {y}, z = {z}")
```

```
=== ASSIGNMENT OPERATORS ===
Assigning values and performing operations in one step

Initial assignment:    num = 5
---------------------------------------------
Add and assign (+=):   num += 3   →  num = 8
Subtract assign (-=):  num -= 2   →  num = 6
Multiply assign (*=):  num *= 4   →  num = 24
Divide assign (/=):    num /= 3   →  num = 8.0
Reset to integer:      num = 8
Floor div assign (//=):num //= 2  →  num = 4
Modulus assign (%=):   num %= 5   →  num = 4
Power assign (**=):    num **= 2  →  num = 16

==================================================

=== MULTIPLE ASSIGNMENT EXAMPLES ===
Multiple assignment:   a, b, c = 1, 2.5, 'Hello'
Results:               a = 1, b = 2.5, c = 'Hello'
Chain assignment:      x = y = z = 10
Results:               x = 10, y = 10, z = 10
```

6. Logical Operators

Logical operators are used to combine conditional statements and work with boolean values. They are essential for creating complex conditions.

In [4]:
```
# Logical Operators - Working with Boolean Logic

print("=== LOGICAL OPERATORS ===")
print("Combining boolean expressions and conditions\n")
```

```python
# Sample boolean variables
is_sunny = True
is_weekend = False
is_holiday = True
temperature = 25

print(f"Given conditions:")
print(f"is_sunny = {is_sunny}")
print(f"is_weekend = {is_weekend}")
print(f"is_holiday = {is_holiday}")
print(f"temperature = {temperature}")
print("-" * 50)

# AND operator (and)
print("AND OPERATOR (and) - Returns True only if BOTH conditions are True:")
print(f"is_sunny and is_weekend      = {is_sunny and is_weekend}")
print(f"is_sunny and is_holiday      = {is_sunny and is_holiday}")
print(f"is_weekend and is_holiday    = {is_weekend and is_holiday}")

# OR operator (or)
print("\nOR OPERATOR (or) - Returns True if AT LEAST ONE condition is True:")
print(f"is_sunny or is_weekend       = {is_sunny or is_weekend}")
print(f"is_weekend or is_holiday     = {is_weekend or is_holiday}")
print(f"False or False               = {False or False}")

# NOT operator (not)
print("\nNOT OPERATOR (not) - Returns the opposite boolean value:")
print(f"not is_sunny                 = {not is_sunny}")
print(f"not is_weekend               = {not is_weekend}")
print(f"not (is_sunny and is_weekend)= {not (is_sunny and is_weekend)}")

print("\n" + "="*60)

# Complex logical expressions
print("\n=== COMPLEX LOGICAL EXPRESSIONS ===")
good_weather = is_sunny and (temperature > 20)
perfect_day = good_weather and (is_weekend or is_holiday)
need_umbrella = not is_sunny

print(f"Good weather (sunny AND temp > 20):           {good_weather}")
print(f"Perfect day (good weather AND weekend/holiday): {perfect_day}")
print(f"Need umbrella (NOT sunny):                    {need_umbrella}")

# Truth table demonstration
print("\n=== TRUTH TABLES ===")
print("AND Truth Table:")
print("True  and True  =", True and True)
print("True  and False =", True and False)
print("False and True  =", False and True)
print("False and False =", False and False)

print("\nOR Truth Table:")
print("True  or True  =", True or True)
print("True  or False =", True or False)
print("False or True  =", False or True)
print("False or False =", False or False)
```

```
=== LOGICAL OPERATORS ===
Combining boolean expressions and conditions

Given conditions:
is_sunny = True
is_weekend = False
is_holiday = True
temperature = 25
--------------------------------------------------
AND OPERATOR (and) - Returns True only if BOTH conditions are True:
is_sunny and is_weekend      = False
is_sunny and is_holiday      = True
is_weekend and is_holiday    = False

OR OPERATOR (or) - Returns True if AT LEAST ONE condition is True:
is_sunny or is_weekend       = True
is_weekend or is_holiday     = True
False or False               = False

NOT OPERATOR (not) - Returns the opposite boolean value:
not is_sunny                 = False
not is_weekend               = True
not (is_sunny and is_weekend)= True

============================================================

=== COMPLEX LOGICAL EXPRESSIONS ===
Good weather (sunny AND temp > 20):           True
Perfect day (good weather AND weekend/holiday): True
Need umbrella (NOT sunny):                    False

=== TRUTH TABLES ===
AND Truth Table:
True  and True  = True
True  and False = False
False and True  = False
False and False = False

OR Truth Table:
True  or True  = True
True  or False = True
False or True  = True
False or False = False
```

7. Operator Precedence

Understanding operator precedence is crucial for writing correct expressions. Python follows specific rules about which operators are evaluated first.

In [6]:
```python
# Operator Precedence - Order of Operations

print("=== OPERATOR PRECEDENCE ===")
print("Understanding the order in which operators are evaluated\n")

print("Precedence Order (highest to lowest):")
print("1. Parentheses ()")
print("2. Exponentiation **")
print("3. Unary +, -, not")
print("4. Multiplication *, Division /, Floor Division //, Modulus %")
print("5. Addition +, Subtraction -")
print("6. Comparison operators ==, !=, <, >, <=, >=")
print("7. Logical operators: not, and, or")
print("-" * 60)

# Examples of precedence in action
print("\n=== PRECEDENCE EXAMPLES ===")

# Arithmetic precedence
result1 = 2 + 3 * 4
result2 = (2 + 3) * 4
print(f"Without parentheses: 2 + 3 * 4     = {result1}")
print(f"With parentheses:    (2 + 3) * 4   = {result2}")

# Exponentiation precedence
result3 = 2 ** 3 ** 2  # Right associative: 2**(3**2) = 2**9
result4 = (2 ** 3) ** 2
print(f"Right associative:   2 ** 3 ** 2   = {result3}")
print(f"Left associative:    (2 ** 3) ** 2 = {result4}")

# Mixed operations
result5 = 10 + 5 * 2 ** 3 / 4
result6 = ((10 + 5) * 2) ** 3 / 4
print(f"Complex expression:  10 + 5 * 2 ** 3 / 4     = {result5}")
print(f"With parentheses:    ((10 + 5) * 2) ** 3 / 4 = {result6}")

print("\n=== LOGICAL PRECEDENCE ===")
# Logical operator precedence
a, b, c = True, False, True

result7 = a or b and c
result8 = (a or b) and c
result9 = a or (b and c)

print(f"Original:      a or b and c      = {result7}")
print(f"Force left:    (a or b) and c    = {result8}")
print(f"Force right:   a or (b and c)    = {result9}")

# Comparison with logical
x, y = 5, 10
result10 = x < y and y > 0 or x == 0
result11 = ((x < y) and (y > 0)) or (x == 0)

print(f"\nComplex condition: x < y and y > 0 or x == 0")
print(f"Evaluated as:      {result10}")
print(f"Explicit grouping: {result11}")

print("\nTIP: When in doubt, use parentheses to make your intentions clear!")
```

```
=== OPERATOR PRECEDENCE ===
Understanding the order in which operators are evaluated

Precedence Order (highest to lowest):
1. Parentheses ()
2. Exponentiation **
3. Unary +, -, not
4. Multiplication *, Division /, Floor Division //, Modulus %
5. Addition +, Subtraction -
6. Comparison operators ==, !=, <, >, <=, >=
7. Logical operators: not, and, or
------------------------------------------------------------

=== PRECEDENCE EXAMPLES ===
Without parentheses: 2 + 3 * 4     = 14
With parentheses:    (2 + 3) * 4   = 20
Right associative:   2 ** 3 ** 2   = 512
Left associative:    (2 ** 3) ** 2 = 64
Complex expression:  10 + 5 * 2 ** 3 / 4     = 20.0
With parentheses:    ((10 + 5) * 2) ** 3 / 4 = 6750.0

=== LOGICAL PRECEDENCE ===
Original:      a or b and c      = True
Force left:    (a or b) and c    = True
Force right:   a or (b and c)    = True

Complex condition: x < y and y > 0 or x == 0
Evaluated as:      True
Explicit grouping: True

TIP: When in doubt, use parentheses to make your intentions clear!
```

8. Practical Examples and Applications

Let's see how operators work together in real-world scenarios and practical applications.

In [7]:
```python
# Practical Applications of Operators

print("=== PRACTICAL EXAMPLES ===")

# Example 1: Grade Calculator
print("GRADE CALCULATOR")
math_score = 85
science_score = 92
english_score = 78

total_points = math_score + science_score + english_score
average = total_points / 3
is_passing = average >= 60

print(f"Math: {math_score}, Science: {science_score}, English: {english_score}")
print(f"Total Points: {total_points}")
print(f"Average: {average:.1f}")
print(f"Passing Grade (≥60): {is_passing}")

# Grade classification using comparison operators
if average >= 90:
    grade = "A"
elif average >= 80:
    grade = "B"
elif average >= 70:
    grade = "C"
elif average >= 60:
    grade = "D"
else:
    grade = "F"
print(f"Letter Grade: {grade}")

print("\n" + "-"*50)

# Example 2: Shopping Cart Calculator
print("\nSHOPPING CART CALCULATOR")
item1_price = 29.99
item2_price = 15.50
item3_price = 8.75
tax_rate = 0.08  # 8% tax
discount_threshold = 50.00
discount_rate = 0.10  # 10% discount

subtotal = item1_price + item2_price + item3_price
qualifies_for_discount = subtotal >= discount_threshold

if qualifies_for_discount:
    discount_amount = subtotal * discount_rate
    subtotal_after_discount = subtotal - discount_amount
else:
    discount_amount = 0
    subtotal_after_discount = subtotal

tax_amount = subtotal_after_discount * tax_rate
final_total = subtotal_after_discount + tax_amount

print(f"Item 1: ${item1_price:.2f}")
print(f"Item 2: ${item2_price:.2f}")
print(f"Item 3: ${item3_price:.2f}")
print(f"Subtotal: ${subtotal:.2f}")
print(f"Qualifies for discount (≥$50): {qualifies_for_discount}")
print(f"Discount amount: ${discount_amount:.2f}")
print(f"After discount: ${subtotal_after_discount:.2f}")
print(f"Tax (8%): ${tax_amount:.2f}")
print(f"Final Total: ${final_total:.2f}")

print("\n" + "-"*50)

# Example 3: Time and Age Calculator
print("\nTIME & AGE CALCULATOR")
current_year = 2024
birth_year = 1995
hours_per_day = 24
minutes_per_hour = 60
seconds_per_minute = 60

age = current_year - birth_year
is_adult = age >= 18
is_senior = age >= 65

# Calculate total seconds in a day
seconds_per_day = hours_per_day * minutes_per_hour * seconds_per_minute
```

```python
print(f"Birth year: {birth_year}")
print(f"Current year: {current_year}")
print(f"Age: {age} years")
print(f"Is adult (≥18): {is_adult}")
print(f"Is senior (≥65): {is_senior}")
print(f"Seconds in a day: {seconds_per_day:,}")

# Age groups using logical operators
is_child = age < 13
is_teenager = age >= 13 and age < 20
is_young_adult = age >= 20 and age < 30
is_middle_aged = age >= 30 and age < 60

print(f"\nAge Group Classification:")
print(f"Child (< 13): {is_child}")
print(f"Teenager (13-19): {is_teenager}")
print(f"Young Adult (20-29): {is_young_adult}")
print(f"Middle Aged (30-59): {is_middle_aged}")
print(f"Senior (≥ 65): {is_senior}")

print("\n" + "="*60)
```

```
=== PRACTICAL EXAMPLES ===
GRADE CALCULATOR
Math: 85, Science: 92, English: 78
Total Points: 255
Average: 85.0
Passing Grade (≥60): True
Letter Grade: B

--------------------------------------------------

SHOPPING CART CALCULATOR
Item 1: $29.99
Item 2: $15.50
Item 3: $8.75
Subtotal: $54.24
Qualifies for discount (≥$50): True
Discount amount: $5.42
After discount: $48.82
Tax (8%): $3.91
Final Total: $52.72

--------------------------------------------------

TIME & AGE CALCULATOR
Birth year: 1995
Current year: 2024
Age: 29 years
Is adult (≥18): True
Is senior (≥65): False
Seconds in a day: 86,400

Age Group Classification:
Child (< 13): False
Teenager (13-19): False
Young Adult (20-29): True
Middle Aged (30-59): False
Senior (≥ 65): False

============================================================
```

Key Takeaways

- **Arithmetic operators** (+, -, *, /, //, %, **) perform mathematical calculations
- **Comparison operators** (==, !=, <, >, <=, >=) compare values and return True/False
- **Assignment operators** (=, +=, -=, *=, etc.) assign and modify variable values efficiently
- **Logical operators** (and, or, not) combine boolean expressions
- **Operator precedence** determines evaluation order - use parentheses for clarity
- **Type compatibility** matters when using operators with different data types

Practice Exercises

Try these exercises to strengthen your understanding:

1. **Calculator Challenge**: Create expressions using all arithmetic operators with two numbers
2. **Grade Evaluator**: Use comparison operators to classify test scores into letter grades
3. **Eligibility Checker**: Use logical operators to determine eligibility based on multiple criteria
4. **Assignment Practice**: Use compound assignment operators to update a counter variable
5. **Precedence Puzzles**: Predict the results of complex expressions before running them

Common Pitfalls to Avoid

- **Integer vs Float Division**: Remember `/` always returns float, `//` returns integer
- **Assignment vs Comparison**: Don't confuse `=` (assignment) with `==` (comparison)
- **Operator Precedence**: When in doubt, use parentheses to make your intentions clear
- **Boolean Context**: Remember that 0, empty strings, and empty collections are "falsy"

Course Information

Learn Python Programming from Scratch
Author: Prakash Ukhalkar
Topic: Python Fundamentals - Basic Operators

Built with ❤️ for the Python learning community