

Chapter 1: Python Fundamentals

Lesson 5: Operators and Expressions

1. What Are Operators and Expressions?

- **Operators:** Special symbols or keywords in Python that perform operations on one or more operands (values or variables). For example, `+` adds two numbers.
 - **Operands:** The values or variables that operators act upon. In `5 + 3`, `5` and `3` are operands.
 - **Expressions:** Combinations of operators and operands that Python evaluates to produce a result. For example, `5 + 3` is an expression that evaluates to `8`.
-

2. Types of Operators in Python

A. Arithmetic Operators

These operators perform mathematical operations.

Operator	Description	Example	Result
<code>+</code>	Addition	<code>5 + 3</code>	<code>8</code>
<code>-</code>	Subtraction	<code>5 - 3</code>	<code>2</code>
<code>*</code>	Multiplication	<code>5 * 3</code>	<code>15</code>
<code>/</code>	Division (returns float)	<code>5 / 2</code>	<code>2.5</code>
<code>//</code>	Floor Division (integer)	<code>5 // 2</code>	<code>2</code>
<code>%</code>	Modulus (remainder)	<code>5 % 2</code>	<code>1</code>
<code>**</code>	Exponentiation	<code>2 ** 3</code>	<code>8</code>

Example:

```
a = 10
b = 3
print(a + b)    # 13
print(a / b)    # 3.333...
print(a // b)   # 3
print(a % b)    # 1
print(a ** 2)   # 100
```

B. Comparison (Relational) Operators

These compare two values and return a boolean (True or False).

Operator	Description	Example	Result
==	Equal to	5 == 5	True
!=	Not equal to	5 != 3	True
>	Greater than	5 > 3	True
<	Less than	5 < 3	False
>=	Greater than or equal	5 >= 5	True
<=	Less than or equal	3 <= 5	True

Example:

```
x = 7
y = 4
print(x > y)    # True
print(x == y)   # False
print(x <= 7)   # True
```

C. Logical Operators

These combine boolean expressions.

Operator	Description	Example	Result
and	True if both are True	True and False	False
or	True if at least one is True	True or False	True
not	Inverts the boolean value	not True	False

Example:

```
a = True
b = False
print(a and b)  # False
print(a or b)   # True
print(not a)    # False
```

D. Assignment Operators

These assign values to variables, often with an operation.

Operator	Description	Example	Equivalent
----------	-------------	---------	------------

Operator	Description	Example	Equivalent
=	Assign	x = 5	x = 5
+=	Add and assign	x += 3	x = x + 3
-=	Subtract and assign	x -= 2	x = x - 2
*=	Multiply and assign	x *= 4	x = x * 4
/=	Divide and assign	x /= 2	x = x / 2
//=	Floor divide and assign	x //= 2	x = x // 2
%=	Modulus and assign	x %= 3	x = x % 3
**=	Exponent and assign	x **= 2	x = x ** 2

Example:

```
x = 10
x += 5 # x is now 15
x *= 2 # x is now 30
print(x) # 30
```

E. Bitwise Operators

These operate on binary representations of integers.

Operator	Description	Example	Result
&	Bitwise AND	5 & 3	1
	Bitwise OR	5 3	7
^	Bitwise XOR	5 ^ 3	6
~	Bitwise NOT	~5	-6
<<	Left shift	5 << 1	10
>>	Right shift	5 >> 1	2

Example:

```
a = 5 # Binary: 0101
b = 3 # Binary: 0011
print(a & b) # 1 (Binary: 0001)
print(a | b) # 7 (Binary: 0111)
print(a << 1) # 10 (Binary: 1010)
```

F. Identity Operators

These check if two variables refer to the same object in memory.

Operator	Description	Example	Result
<code>is</code>	True if same object	<code>x is y</code>	Varies
<code>is not</code>	True if different object	<code>x is not y</code>	Varies

Example:

```
x = [1, 2, 3]
y = x
z = [1, 2, 3]
print(x is y)      # True (same object)
print(x is z)      # False (different objects)
print(x == z)      # True (same values)
```

G. Membership Operators

These test if a value is in a sequence (like a list, string, or tuple).

Operator	Description	Example	Result
<code>in</code>	True if value is found	<code>3 in [1, 2, 3]</code>	True
<code>not in</code>	True if value not found	<code>4 not in [1, 2, 3]</code>	True

Example:

```
fruits = ["apple", "banana"]
print("apple" in fruits)      # True
print("orange" not in fruits) # True
```

3. Operator Precedence

When multiple operators are in an expression, Python follows a precedence order (similar to math's PEMDAS). Here's a simplified list (highest to lowest):

1. `**` (Exponentiation)
2. `~, +, -` (Unary operators)
3. `*, /, //, %`
4. `+, -` (Addition/Subtraction)
5. `<<, >>`
6. `&`
7. `^, |`

- 8. Comparison operators (>, <, ==, etc.)
- 9. not
- 10. and
- 11. or

Example:

```
result = 5 + 3 * 2 # Multiplication first: 5 + (3 * 2) = 11
print(result)      # 11

result = (5 + 3) * 2 # Parentheses override: (5 + 3) * 2 = 16
print(result)      # 16
```

4. Building Expressions

Expressions combine operators and operands. Python evaluates them based on precedence and associativity (left-to-right for most operators, except `**` which is right-to-left).

Examples:

```
# Simple expression
x = 5 + 3 * 2 # 11

# Complex expression
y = (10 / 2) ** 2 + 3 > 15 and 7 % 2 == 1
# Step-by-step:
# 10 / 2 = 5
# 5 ** 2 = 25
# 25 + 3 = 28
# 28 > 15 = True
# 7 % 2 = 1
# 1 == 1 = True
# True and True = True
print(y) # True
```

5. Practical Examples

Let's apply what we've learned:

Example 1: Calculate Area

```
length = 5
width = 3
area = length * width
print(f"Area: {area}") # Area: 15
```

Example 2: Check Even or Odd

```
num = 10
is_even = num % 2 == 0
print(f"{num} is even: {is_even}") # 10 is even: True
```

Example 3: Compound Condition

```
age = 25
has_id = True
can_enter = age >= 21 and has_id
print(f"Can enter: {can_enter}") # Can enter: True
```

6. Common Pitfalls

- **Division:** `/` always returns a float; use `//` for integer division.
- **Precedence:** Use parentheses to enforce evaluation order if unsure.
- **Type Errors:** Ensure operands are compatible (e.g., can't add a string and int directly).

7. Exercises for Practice

1. Write an expression to calculate the perimeter of a rectangle.
2. Check if a number is divisible by both 3 and 5.
3. Write a program that:
 - Asks the user to input two numbers.
 - Prints the results of addition, subtraction, multiplication, division, and modulus operations.
4. Write a program that:
 - Asks the user to input a base number.
 - Calculates and prints the result of raising the base to the powers of 1, 2, and 3 using the `**` operator.
5. Write a program that:
 - Asks the user if it's sunny (yes/no) and if it's warm (yes/no).
 - Converts responses to booleans (`True` for "yes", `False` for "no").
 - Uses `and`, `or`, and `not` operators to print:
 - Is it sunny AND warm?
 - Is it sunny OR warm?
 - Is it NOT sunny?
6. Create a program that:
 - Asks the user to input a starting score.
 - Uses assignment operators (`+=`, `-=`, `*=`) to:
 - Add 5 to the score.
 - Subtract 3 from the score.
 - Multiply the score by 2.
 - Prints the score after each operation.

7. Write a program that:

- Defines a list of fruits (e.g., ["apple", "banana", "grape"]).
- Asks the user to input a fruit name.
- Uses `in` and `not in` operators to check and print:

8. Create a program that:

- Asks the user to input their age and whether they have permission (yes/no).
- Converts the permission response to a boolean (`True` for "yes", `False` for "no").
- Checks if the person can enter a restricted area (age \geq 18 AND has permission).
- Prints a message like "Entry allowed: False".

9. Create a program that:

- Asks the user to input two numbers.
 - Uses comparison operators (`>`, `<`, `==`, `!=`) to check and print whether the first number is greater than, less than, equal to, or not equal to the second.
-