Operators in Python

Premanand S

Assistant Professor
School of Electronics Engineering
Vellore Institute of Technology
Chennai Campus

premanand.s@vit.ac.in

November 26, 2024

Introduction to Operators

- **Definition**: Operators are symbols or keywords used to perform operations on variables and values.
- **Importance**: Essential for calculations, comparisons, logical decisions, and more.

Types of Operators in Python

- Arithmetic Operators
- Comparison Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators
- Membership Operators
- Identity Operators
- Ternary (Conditional) Operators

Arithmetic Operators

- Perform basic mathematical operations.
- Operators: +, −, *, /, //, %, **.

Example (Python Code)

```
a, b = 10, 3
print(a + b)  # Addition
print(a // b)  # Floor division
print(a ** b)  # Exponentiation
```

Comparison (Relational) Operators

- Compare two values.
- Operators: ==, !=, >, <, >=, <=.

Example (Python Code)

```
a, b = 10, 20
print(a == b)  # False
print(a < b)  # True
```

Logical Operators

- Combine conditional statements.
- Operators: `and`, `or`, `not`.

Example (Python Code)

```
x, y = True, False
print(x and y)  # False
print(x or y)  # True
print(not x)  # False
```

Bitwise Operators

- Perform bit-level operations.
- Operators: &, |, ^, ~, <<, >>.

Example (Python Code)

```
a, b = 5, 3
print(a & b)  # Bitwise AND
print(a << 1)  # Left shift
```

Assignment Operators

- Assign values to variables and perform operations simultaneously.
- Operators: =, +=, −=, ∗=, /=, %=, ∗∗=, //=.

Example (Python Code)

```
a = 10
a += 5  # Same as a = a + 5
print(a)  # 15
```

Membership Operators

- Check membership in sequences like strings, lists, or tuples.
- Operators: `in`, `not in`.

Example (Python Code)

```
nums = [1, 2, 3]
print(2 in nums)  # True
print(4 not in nums)  # True
```

Identity Operators

- Compare memory locations of objects.
- Operators: `is`, `is not`.

Example (Python Code)

```
a = [1, 2, 3]
b = a
print(a is b)  # True
print(a is not b)  # False
```

Ternary (Conditional) Operators

- Conditionally assign values in a single line.
- Syntax: `[value_if_true] if [condition] else [value_if_false]`.

Example (Python Code)

```
x, y = 10, 20
max_val = x if x > y else y
print(max_val)  # 20
```

Overloading Operators

- Customize the behavior of operators for user-defined objects.
- Use special methods like `__add__`, `__sub__`, `__eq__`.

Example (Python Code)

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, other):
        return Point(self.x + other.x, self.y + other.y)

p1 = Point(1, 2)
p2 = Point(3, 4)
print(p1 + p2)  # Outputs: (4, 6)
```

Augmented Assignment Operators

- Combine binary operation and assignment in one step.
- Faster than separate operations.
- Operators: +=, −=, ∗=, /=, %=, ∗∗=, //=.

Example (Python Code)

```
a = 10
a += 5  # Same as a = a + 5
print(a)  # Output: 15
```

Boolean Operators with Numbers

- Python treats numbers as boolean values (0 is False, non-zero is True).

Example (Python Code)

```
print(1 and 0)  # Output: 0 (False)
print(1 or 0)  # Output: 1 (True)
```

Operator Behavior with Strings

- Operators like + (concatenation), * (repetition), and `in` (membership) work with strings.

Example (Python Code)

```
print("Hello" + " World")  # Concatenation
print("Hi " * 3)  # Repetition
print("P" in "Python")  # Membership
```

Chaining Comparisons

- Combine multiple comparison operators for readable conditions.

Example (Python Code)

```
x = 10
print(5 < x < 20)  # Output: True
```

Short-Circuit Behavior of Logical Operators

- Logical operators (`and`, `or`) stop evaluation as soon as the result is determined.

Example (Python Code)

```
def check():
    print("Checking...")
    return True

print(False and check())  # Output: False (short-circuited)
print(True or check())  # Output: True (short-circuited)
```

Overloading Operators with Special Methods

- Customize operator behavior for user-defined objects.
- Methods: __add__, __sub__, __eq__, etc.

Example (Python Code)

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, other):
        return Point(self.x + other.x, self.y + other.y)

p1 = Point(1, 2)
p2 = Point(3, 4)
print(p1 + p2)  # Outputs: (4, 6)
```

Debugging and Precedence

- Parentheses can override operator precedence.
- Use precedence tables to debug complex expressions.

Example (Python Code)

```
print(10 + 2 * 3)  # Multiplication first
print((10 + 2) * 3)  # Parentheses override precedence
```

Using operator Module

- Provides functional equivalents of operators.
- Common functions: add, mul, eq, gt, etc.

Example (Python Code)

```
import operator
print(operator.add(10, 5))  # Output: 15
print(operator.gt(10, 5))  # Output: True
```

Understanding Floating-Point Arithmetic

- Floating-point operations may lead to precision issues.
- Use `decimal.Decimal` for high precision.

Example (Python Code)

```
from decimal import Decimal
print(Decimal('0.1') + Decimal('0.2'))  # Output: 0.3
```

Unary Operators

- Operate on a single operand.
- Operators: +, −, ˜.

Example (Python Code)

```
x = 5
print(-x)  # -5
print(~x)  # -6 (bitwise NOT)
```

To brush it up!

Calculate Restaurant Bill with Tax and Tip

Problem: Write a program to calculate the total amount of a restaurant bill, including tax and tip. **Solution:**

Example (Python Code)

```
bill_amount = float(input("Enter bill amount: "))
tax = bill_amount * 0.10  # 10% tax
tip = bill_amount * 0.15  # 15% tip
total_amount = bill_amount + tax + tip
print(f"Total amount: ${total_amount:.2f}")
```

Check Leap Year

Problem: Write a program to check if a given year is a leap year.
Solution:

Example (Python Code)

```
year = int(input("Enter a year: "))
if (year % 4 == 0 and year % 100 != 0) or
                                (year % 400 == 0):
    print(f"{year} is a leap year.")
else:
    print(f"{year} is not a leap year.")
```

Check Number Within Range

Problem: Determine if a number is within a range (inclusive).
Solution:

Example (Python Code)

```
num = int(input("Enter a number: "))
lower = 10
upper = 20
if lower <= num <= upper:
    print(f"{num} is within the range {lower}-{upper}.")
else:
    print(f"{num} is not within the range {lower}-{upper}.")
```

Swap Two Numbers Without Using a Third Variable

Problem: Swap two numbers using bitwise XOR. **Solution:**

Example (Python Code)

```
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
a = a ^ b
b = a ^ b
a = a ^ b
print(f"Swapped values: a = {a}, b = {b}")
```

Calculate Compound Interest

Problem: Calculate the compound interest on an amount. **Solution:**

Example (Python Code)

```
principal = float(input("Enter principal amount: "))
rate = float(input("Enter annual interest rate (%): ")) / 100
time = int(input("Enter time in years: "))
amount = principal * (1 + rate) ** time
interest = amount - principal
print(f"Compound interest: ${interest:.2f}")
```

Check Word in Sentence

Problem: Check if a word exists in a given sentence. **Solution:**

Example (Python Code)

```
sentence = input("Enter a sentence: ").lower()
word = input("Enter a word to search: ").lower()
if word in sentence:
    print(f"The word '{word}' exists in the sentence.")
else:
    print(f"The word '{word}' does not exist in the
                                                    sentence.")
```

Compare Memory Locations of Lists

Problem: Compare two lists and check if they share the same memory location. **Solution:**

Example (Python Code)

```
list1 = [1, 2, 3]
list2 = list1
list3 = [1, 2, 3]

print(list1 is list2)  # True, same memory location
print(list1 is list3)  # False, different objects
```

Check Even or Odd Using Ternary Operator

Problem: Check if a number is even or odd. **Solution:**

Example (Python Code)

```
num = int(input("Enter a number: "))
result = "Even" if num % 2 == 0 else "Odd"
print(f"The number {num} is {result}.")
```

Calculate Area of Circle and Rectangle

Problem: Write a program to calculate the area of a circle and a rectangle. **Solution:**

Example (Python Code)

```python
import math

radius = float(input("Enter radius of the circle: "))
length = float(input("Enter length of the rectangle: "))
width = float(input("Enter width of the rectangle: "))

circle_area = math.pi * radius ** 2
rectangle_area = length * width

print(f"Circle area: {circle_area:.2f}")
print(f"Rectangle area: {rectangle_area:.2f}")
```

Scholarship Eligibility

Problem: Determine if a student qualifies for a scholarship.
Solution:

Example (Python Code)

```
age = int(input("Enter student age: "))
grade = float(input("Enter student grade: "))
if age <= 18 and grade >= 90:
    print("Eligible for scholarship.")
else:
    print("Not eligible for scholarship.")
```

Count Set Bits in an Integer

Problem: Count the number of set bits (1s) in an integer. **Solution:**

Example (Python Code)

```
def count_set_bits(n):
    count = 0
    while n:
        count += n & 1
        n >>= 1
    return count

num = int(input("Enter a number: "))
print(f"Number of set bits: {count_set_bits(num)}")
```

Check if String Contains Only Vowels

Problem: Check if a string contains only vowels. **Solution:**

Example (Python Code)

```
vowels = "aeiou"
string = input("Enter a string: ").lower()
if all(char in vowels for char in string):
    print("The string contains only vowels.")
else:
    print("The string contains other characters.")
```

Compare Dictionaries for Memory Location

Problem: Compare two dictionaries for memory location. **Solution:**

Example (Python Code)

```
dict1 = {"key1": "value1"}
dict2 = dict1
dict3 = {"key1": "value1"}

print(dict1 is dict2)  # True, same memory location
print(dict1 is dict3)  # False, different objects
```