Python: Operators

Connect with me: Youtube | LinkedIn | WhatsApp Channel | Web | Facebook | Twitter

- Download PDF
- To access the updated handouts, please click on the following link: https://yasirbhutta.github.io/python/docs/operators.html

What is Operators

- Operators in Python are special symbols that perform specific operations on values and variables.
- They are essential for calculations, comparisons, assignments, and logical operations within your code.

Major operator categories

1. Arithmetic Operators:

- Used for performing basic mathematical calculations.
- Operators: `+` (addition), `-` (subtraction), `*` (multiplication), `/` (division), `//` (floor division), `%` (modulo), `**` (exponentiation)

video: Basic Mathematical Operations with Examples

- Examples:

```
result = 10 + 5  # Addition
difference = 15 - 7  # Subtraction
product = 4 * 6  # Multiplication
quotient = 12 / 3  # Division
integer_quotient = 17 // 4  # Floor division
remainder = 25 % 4  # Modulo
square = 5 ** 2  # Exponentiation
```

For more details on floor division, see What is the `//` Operator in Python?

2. Comparison Operators:

- Used to compare values and return a Boolean result (True or False).
- Operators: `==` (equal to), `!=` (not equal to), `>` (greater than), `<` (less than), `>=` (greater than or equal to), `<=` (less than or equal to)

video: How to Use Comparision Operators in Python

Examples:

```
is_equal = 7 == 7  # True
is_greater = 12 > 9  # True
is_less_or_equal = 5 <= 5  # True
```

3. Assignment Operators:

- Used to assign values to variables.
- Operators: `=`
- Examples:

```
x = 10  # Simple assignment
```

Compound assignment:

A **compound assignment** is a combination of an operator and an assignment that performs an operation on a variable and then assigns the result back to that variable in a single step.

For example, in the code:

```
message += " Welcome!"
```

This is a **compound assignment** using the `+=` operator. Here's what it does:

1. It concatenates (adds) `" Welcome!"` to the existing value of the `message` variable.
2. Then, it assigns the result back to `message`.

In general, the format of compound assignment operators is:

- `+=` : Adds and assigns
- `-=` : Subtracts and assigns
- `*=` : Multiplies and assigns
- `/=` : Divides and assigns
- `%=` : Takes the modulus and assigns

This shorthand avoids writing out the operation in a longer form, such as:

```
message = message + " Welcome!"
```

The `+=` operator helps to keep the code more concise and readable.

Example:

```
x += 5  # Add 5 to x
x *= 2  # Multiply x by 2
```

4. Logical Operators:

- Used to combine Boolean expressions.
- Operators: `and`, `or`, `not`
- Examples:

```
is_valid = (age >= 18) and (has_license == True)
is_allowed = (is_member) or (has_ticket)
```

5. Identity Operators:

- In Python, the `is` operator is used to compare the identities of two objects. It checks whether two references point to the same object in memory, not whether the values of the two objects are equal.
- Operators: `is`, `is not`

Here's a simple example to illustrate the difference between `is` and `==`:

```
a = [1, 2, 3]
b = [1, 2, 3]
c = a

print(a == b)  # Output: True, because the lists have the same content
print(a is b)  # Output: False, because they are different objects in memory
print(a is c)  # Output: True, because both references point to the same object
```

In this example:

- `a == b` is `True` because the values in both lists are the same.
- `a is b` is `False` because they are two different objects in memory, even though their contents are identical.
- `a is c` is `True` because `c` is assigned to the same object as `a`, so they reference the same memory location.

Example 2

```
x = 5
y = 5
result = x is y  # True (they refer to the same integer object)
```

Object Caching in Python: Understanding Memory Optimization for Small Integers

- In Python, small integer objects (typically between -5 and 256) are cached to optimize memory usage and performance. This means that if you create multiple variables with the same value within this range, they all point to the same memory location, instead of creating new objects each time.
- This behavior is part of Python's optimization to save memory and improve performance, as small integers are frequently used. [1]

Here's an example to explain the concept:

```
# Assigning variables within the cached range
a = 100
b = 100

# Checking if the IDs are the same
print(id(a))  # Outputs the memory address of 'a'
print(id(b))  # Outputs the memory address of 'b'

# Since both 'a' and 'b' are within the cached range, their IDs are the same
print(a is b)  # True, as both refer to the same cached object

# Assigning variables outside the cached range
x = 300
y = 300

# Checking if the IDs are the same
print(id(x))  # Outputs the memory address of 'x'
print(id(y))  # Outputs the memory address of 'y'

# Since 'x' and 'y' are outside the cached range, they are different objects
print(x is y)  # False, as 'x' and 'y' refer to different objects
```

Explanation:

1. **Cached Objects**: The values `a` and `b` are both set to `100`, which falls within the default cached range of small integers. As a result, Python uses the same memory location for both, which is why `id(a)` and `id(b)` are the same, and `a is b` returns `True`.
2. **Non-Cached Objects**: The values `x` and `y` are both set to `300`, which is outside the cached range. Therefore, Python creates separate objects for each, and `id(x)` and `id(y)` are different, meaning `x is y` returns `False`.

This caching behavior saves memory for frequently used small integers and makes the comparison of such objects faster.

6. Membership Operators:

- Used to check if a value is present in a sequence (like a list or string).
- Operators: `in`, `not in`
- Examples:

```
numbers = [1, 3, 5, 7]
is_present = 5 in numbers  # True
is_missing = 2 not in numbers  # True
```

7. Bitwise Operators:

- Used to perform operations on the binary representation of numbers.
- Operators: `&` (bitwise AND), `|` (bitwise OR), `^` (bitwise XOR), `~` (bitwise NOT), `<<` (left shift), `>>` (right shift)
- These are less common in general-purpose programming, but useful in certain domains like low-level programming or cryptography.

`id()` function

In Python, the `id()` function returns the "identity" of an object, which is a unique integer that serves as a constant for the object during its lifetime. The `id` is often the memory address of the object. Here's a simple example:

```
x = 10
print(id(x))

y = "hello"
print(id(y))
```

This code will print the unique identifiers for the variables `x` and `y`. If you have specific objects or variables in mind that you would like to see the `id` for, please let me know!

Python operators examples:

- Walrus Operator in Python

Key Terms

True/False (Mark T for True and F for False)

Answer Key (True/False):

Multiple Choice (Select the best answer)

1. **Arithmetic Operators:**

1. **What is the output of the following Python code?** Python Quiz #2

```
x = 10
y = 5

x = x + y
y = x - y
x = x - y

print(x, y)
```

- A) 5 0
- B) 5 -5
- C) 10 5
- D) 5 10

Watch this video for answers: https://youtube.com/shorts/gz4YuXmZvYo?si=VK50ZYFnvh5WljmT

2. **What is the output of the following expression?** Python Quiz #12

```
2 ** 3 + 4 // 2 - 1
```

- A) 8
- B) 9
- C) 10
- D) 11

Watch this video for the answer: https://youtube.com/shorts/_cHsABqmmcM

3. **What is the output of the following expression?** Python Quiz #24

```
x = 10
y = 5
print(x % y)
```

- A) 2
- B) 5
- C) 0
- D) 1

Watch this video for the answer: https://youtube.com/shorts/-mPZMfg_uJ8?si=F6Hk4m4C_HVmHZ9A

4. **What is the output of the following code?** Python Quiz #33

```
print(11 % 3 == (11 - 3 * (11 // 3)))
```

- A) True
- B) False

Watch the video for the answer: https://youtube.com/shorts/weXLcIrx2Ko?si=JlDU0qMLPgYedatJ

5. **What will be the output of the following code?** [Python Quiz #61]

```
age = 25
message = "You are " + str(age) + " years old."
```

```
message += " Welcome!"
print(message)
```

- A) `You are 25 years old. Welcome!`
- B) `You are 25 years old.`
- C) `You are 25 Welcome!`
- D) `You are Welcome!`

Watch video for the answer: https://youtu.be/Q2J1EEedb9E

6. **What is the output of 45 // 7?** [Python Quiz #62]

- A) 5.0
- B) 6
- C) 5
- D) 6.428571428571429

2. **Comparison Operators:**

7. **What is the output of the following Python code?** Python Quiz #10

```
x = 5
y = 10

result = x > 3 or y < 5
print(result)
```

- A) True
- B) False
- C) SyntaxError
- D) None of these

Watch video for the answer: https://youtube.com/shorts/FRa0r4UxyXM

8. **What is the output of the following PYTHON code?** [Python Quiz #82]

```
flag = not (True and False)
print(flag)
```

- A) True
- B) False
- C) None
- D) Error

3. Assignment Operators: 4. Logical Operators: 5. Identity Operators: 6. Membership Operators: 7. Bitwise Operators:

1. **What is the difference between and and or operators in Python?**
    - A) and returns True if both operands are True and or returns True if either operand is True
    - B) and returns True if either operand is True and or returns True if both operands are True
    - C) and returns False if both operands are False and or returns False if either operand is False
    - D) both A and C are correct

Answer: D

What is the difference between == and = in Python?

- A) == is the comparison operator, = is the assignment operator
- B) == is the assignment operator, = is the comparison operator
- C) They are the same operator
- D) There is no difference

Which operator is used to raise a number to a power?

- A) **
- B) ^
- C) pow()
- D) ** and ^ are both correct

What is the result of the expression 3 + 5 * 2 in Python? [Python Quiz #63]

- A) 16
- B) 13
- C) 11
- D) 26

What is the purpose of the % operator in Python?

- A) Exponential
- B) Modulus
- C) Floor Division
- D) Addition

Which operator adds a value to a variable?

- A) +=
- B) -=
- C) *=
- D) /=

What is the output of x = 10; x //= 3?

- A) 3
- B) 3.33
- C) 3.5
- D) 4

Which operator returns True if both operands are True?

- A) and
- B) ror
- C) not
- D) xor

Which operator checks if a value is present in a sequence?

- A) in
- B) not in
- C) Both in and not in
- D) None of the above

What is the output of the following PYTHON Code? [Python Quiz #83]

```
num = 4
result = num in [1, 3, 4]
print(result)
```

```
- A) `True`
- B) `False`
- C) `None`
- D) `Error`
```

What will be the output of the following PYTHON Code? [Python Quiz #84]

```
str = 'x'
result = str in 'python'
print(result)
```

- A) `True`
- B) `False`
- C) `'x'`
- D) `Error`

Which operator checks if two objects refer to the same memory location?

- A) ==
- B) is
- C) in
- D) Both is and in

What will be the output of the following PYTHON Code? [Python Quiz #85]

```
x = [1, 2]; y = x; print(x is y)
```

- A) True
- B) False
- C) [1, 2]
- D) Error

What is the correct way to use the exponentiation operator in Python?

- A) x ^ y
- B) x ** y
- C) pow(x, y)
- D) either b or c

Answer: B) either b or c

In Python, What is the output of this code? [Python Quiz #86]

```
x = 10
y = 5
x = x + y
y = x - y
x = x - y
print(x, y)
```

- A) 10, 5
- B) 5, 10
- C) 15, -5
- D) -5, 15

Answer key (Mutiple Choice):

Fill in the Blanks

Answer Key (Fill in the Blanks):

Coding Exercises

Beginner Exercises

Exercise #1: Rectangle Area and Perimeter Calculation

Task: Write a program to calculate the area and perimeter of a rectangle.

Input:

- A floating-point number `length`, representing the length of the rectangle.
- A floating-point number `width`, representing the width of the rectangle.

Output:

- Print the area of the rectangle.
- Print the perimeter of the rectangle.

Example:

- **Input:**
  ```
  length = 5.0
  width = 3.0
  ```
- **Output:**
  ```
  Area: 15.0
  Perimeter: 16.0
  ```
- **Input:**
  ```
  length = 7.5
  width = 2.5
  ```
- **Output:**
  ```
  Area: 18.75
  Perimeter: 20.0
  ```

Exercise 2: Converts temperature from Celsius to Fahrenheit

Task: Create a program that converts temperature from Celsius to Fahrenheit using the formula: `(Celsius * 9/5) + 32`.

Exercise 3: Calculating Sum, Difference, Product, and Quotient

Task: Write a program that takes two numbers and prints their sum, difference, product, and quotient.

Exercise 4: Basic Assignment

Task: Assign the value `10` to the variable `a` and print the value of `a`.

Exercise 5: Compound Assignment with Addition (`+=`)

Task:

1. Assign the value `15` to a variable `x`.
2. Use the `+=` operator to add `5` to `x`.
3. Print the new value of `x`.

Exercise 6: Compound Assignment with Subtraction (`-=`) [Easy]

Task:

1. Assign the value `20` to a variable `y`.
2. Use the `-=` operator to subtract `7` from `y`.
3. Print the new value of `y`.

Exercise 7: Modulus Assignment (`%=`)

Task:

1. Assign `13` to a variable `m`.
2. Use the `%=` operator to assign the remainder when `m` is divided by `5`.
3. Print the result.

Exercise 8: Swap Variables Using Assignment Operators [Easy]

Task: Swap the values of two variables `a = 10` and `b = 20` without using a third variable or tuple assignment.

Exercise 9: Complex Assignment Expression

Task:

1. Create three variables: `a = 4`, `b = 5`, and `c = 6`.
2. Use a single line with assignment operators to update the values of all three variables, such that:
    - `a` is multiplied by `2`.
    - `b` is increased by `10`.
    - `c` is divided by `3`.
3. Print the values of all three variables.

Exercise 10:

Task: Write a program that checks if two variables point to the same object using the `is` operator.

Exercise 11:

Task: Create a list of fruits and write a program that checks if "apple" is in the list using the `in` operator.

Intermediate Exercises

Bitwise Operators

1. Write a program that takes two numbers as input and prints their bitwise AND, OR, and XOR.
2. Use the left shift operator (`<<`) to multiply a number by 4 and the right shift operator (`>>`) to divide a number by 2.

Advanced Exercises

Practical Mini-Project

1. **Shopping Cart**: Create a shopping cart program. Initialize a total cost as 0. Add the cost of items using `+=` as items are added. Print the total cost when all items have been added.
2. **Odd or Even Checker**: Write a program that takes a number and uses the modulo operator (`%`) to check if it is even or odd.

Review Questions

References and Bibliography

[1] “Python memory management,” Discussions on Python.org, Apr. 02, 2023. https://discuss.python.org/t/python-memory-management/25391 (accessed Jul. 27, 2024).