# Boolean Expression

## Topic 03 Key concept

## 1. Boolean Data Type and Boolean Expression

### 1.1 True and False

We learned that the bool data type consists of True and False, which represent true and false, respectively.

## 1. Boolean Data Type and Boolean Expression

### 1.1 True and False

- Data of a non-bool type can also be **converted to bool type.**

```
1 print(bool(1))
2 print(bool(0))
3 print(bool(''))
4 print(bool(None))
```

```
True
False
False
False
```

## 1. Boolean Data Type and Boolean Expression

### 1.1 True and False

- Data of a non-bool type can also be **converted to bool type.**

```
1 print(bool(1))
2 print(bool(0))
3 print(bool(''))
4 print(bool(None))
```

```
True
False
False
False
```

**Line 1**

- When converting any number except 0 to bool type, it becomes True.

# 1. Boolean Data Type and Boolean Expression

## 1.1 True and False

- Data of a non-bool type can also be **converted to bool type.**

```
1 print(bool(1))
2 print(bool(0))
3 print(bool(''))
4 print(bool(None))
```

```
True
False
False
False
```

**Line 2**

- Converting the number 0 to bool type results in False.

# 1. Boolean Data Type and Boolean Expression

## 1.1 True and False

- Data of a non-bool type can also be **converted to bool type.**

```
1 print(bool(1))
2 print(bool(0))
3 print(bool(''))
4 print(bool(None))
```

```
True
False
False
False
```

**Line 3**

- Converting empty strings ('') to bool type results in False, and any other non-empty character results in True.

# 1. Boolean Data Type and Boolean Expression

## 1.1 True and False

- Data of a non-bool type can also be **converted to bool type.**

```
1 print(bool(1))
2 print(bool(0))
3 print(bool(''))
4 print(bool(None))
```

```
True
False
False
False
```

**Line 4**

- None is a keyword that represents the absence of a value, so it becomes False.

# 1. Boolean Data Type and Boolean Expression

## 1.1 True and False

- It is important to note that non-bool types can be interpreted as False if they represent 0 or an empty value, and as True if they represent a non-zero value or something is present.

```
1 print(bool(-1))
2 print(bool('Hello'))
```

```
True
True
```

# 1. Boolean Data Type and Boolean Expression

## 1.1 True and False

- It is important to note that non-bool types can be interpreted as False if they represent 0 or an empty value, and as True if they represent a non-zero value or something is present.

```
1 print(bool(-1))
2 print(bool('Hello'))
```

```
True
True
```

**Line 1**

- -1 is considered True.

# 1. Boolean Data Type and Boolean Expression

## 1.1 True and False

- It is important to note that non-bool types can be interpreted as False if they represent 0 or an empty value, and as True if they represent a non-zero value or something is present.

```
1 print(bool(-1))
2 print(bool('Hello'))
```

```
True
True
```

**Line 2**

- Non-empty strings are considered True.

AnB_Prime

——>4<——

# 1. Boolean Data Type and Boolean Expression

## 1.2 Boolean Expression


- A Boolean expression is an expression that evaluates to either True or False, representing true or false, respectively.
  - Boolean expressions return Boolean values, either True or False.

# 1. Boolean Data Type and Boolean Expression

## 1.2 Boolean Expression

- Various operators and operations are used to create different Boolean expressions.

| Operation | Description | Operator |
|---|---|---|
| Comparison operation | Comparing values | Comparison operators (==, !=, >, >=, <, <=) |
| Logical operation | Combining Boolean values | Logical operators(and, or, not) |
| Membership operation | Determining if a value is present in data | in, not in |
| Identity operation | Determining if two objects are the same | is, not is |

# 1. Boolean Data Type and Boolean Expression

## 1.2 Boolean Expression

- Additionally, evaluating values to Boolean (true or false) **using the bool function** is also considered a **Boolean expression**.

| Operation | Description | Operator |
|---|---|---|
| Comparison operation | Comparing values | Comparison operators (==, !=, >, >=, <, <=) |
| Logical operation | Combining Boolean values | Logical operators(and, or, not) |
| Membership operation | Determining if a value is present in data | in, not in |
| Identity operation | Determining if two objects are the same | is, not is |

# 1. Boolean Data Type and Boolean Expression

## 1.2 Boolean Expression

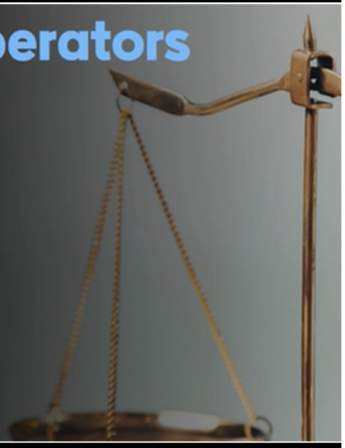
- Additionally, evaluating values to Boolean (true or false) **using the bool function** is also considered a **Boolean expression**.

| Operation | | Operator |
|---|---|---|
| Comparison operation | Comparing values | Comparison operators (==, !=, >, >=, <, <=) |
| Logical operation | | operators(and, or, not) |
| Membership | | in |

These **Boolean expressions** can **be used to evaluate conditions and control the flow of execution based on those conditions.**

# 2. Comparison Operators

## 2.1 What is Comparison Operator?



**Comparison operator**

- **Comparison operator** compares two operands, which can be numeric or string data, and return True or False based on the comparison results.

# 2. Comparison Operators

## 2.1 What is Comparison Operator?

- **Comparison operator** compares two operands, which can be numeric or string data, to determine their relative order, such as 'greater than' or 'less than'.
- They return True or False as the result of the comparison. This type of value, either True or False, is called a **bool value**.

```
1 print('Results of 10 > 5 :', 10 > 5)
2 print('Results of 5 < 1 :', 5 < 1)
3 print('Results of 5 == 5 :', 5 == 5)
4 print('Results of 5 != 5 :', 5 != 5)
5 print("Results of 'a' > 'b' :", 'a' > 'b')
```

```
Results of 10 > 5 : True
Results of 5 < 1 : False
Results of 5 == 5 : True
Results of 5 != 5 : False
Results of 'a' > 'b' : False
```

# 2. Comparison Operators

## 2.1 What is Comparison Operator?

- Let's examine the detailed explanation and results of comparison operators.

| Operator | Explanation | a = 100<br>b = 200 |
|---|---|---|
| == | Returns True if the two operands are equal. | a == b is False |
| != | Returns True if the two operands are not equal. | a != b is True |
| > | Returns True if the left operand is greater than the right operand. | a > b is False |
| < | Returns True if the left operand is less than the right operand. | a < b is True |

# 2. Comparison Operators

## 2.1 What is Comparison Operator?

- Let's examine the detailed explanation and results of comparison operators.

| Operator | Explanation | a = 100<br>b = 200 |
|---|---|---|
| >= | Returns True if the left operand is greater than or equal to the right operand. | a >= b is False |
| <= | Returns True if the left operand is less than or equal to the right operand. | a <= b is True |

# 2. Comparison Operators

## 2.2 Numeric Type and Comparison Operator

```
1 a = 100
2 b = 200
3 print(a == b)
4 print(a != b)
5 print(a > b)
6 print(a < b)
7 print(a >= b)
8 print(a <= b)
```

```
False
True
False
True
False
True
```

# 2. Comparison Operators

## 2.2 Numeric Type and Comparison Operator

```
1  a = 100
2  b = 200
3  print(a == b)
4  print(a != b)
5  print(a > b)
6  print(a < b)
7  print(a >= b)
8  print(a <= b)
```

**Focus**

The value returned when a comparison operator is applied is shown as a boolean value.

```
False
```

# 2. Comparison Operators

## 2.2 Numeric Type and Comparison Operator

```
1  a = 100
2  b = 200
3  print(a == b)
4  print(a != b)
5  print(a > b)
6  print(a < b)
7  print(a >= b)
8  print(a <= b)
```

**==**

- The == operator only returns True when the values are equal.
- In the example, the variable a is 100 and b is 200, so the values are not the same.
- Therefore, it returns False.

```
False
True
False
True
False
True
```

# 2. Comparison Operators

## 2.2 Numeric Type and Comparison Operator

```
1  a = 100
2  b = 200
3  print(a == b)
4  print(a != b)
5  print(a > b)
6  print(a < b)
7  print(a >= b)
8  print(a <= b)
```

**!=**

- The second comparison operator (!=) returns True only when the values are not equal.
- Therefore, in the example, while variable a is 100 and b is 200, they are not the same.
- Therefore, a != b would return True as the result.

```
False
True
False
True
False
True
```

# 2. Comparison Operators

## 2.2 Numeric Type and Comparison Operator

```
1 a = 100
2 b = 200
3 print(a == b)
4 print(a !
5 print(a >
6 print(a <
7 print(a >
8 print(a <
```

```
False
```

Comparison operators allow us to compare values using operators such as 'greater than', 'less than', 'greater than or equal to', and 'less than or equal to'.

# 2. Comparison Operators

## 2.2 Numeric Type and Comparison Operator

- When comparison operators are applied, they return bool values.

```
1 n = int(input('Enter an integer: '))
2 print('Is this an even number?', n % 2 == 0)
```

```
Enter an integer: 60
Is this an even number? True
```

# 2. Comparison Operators

## 2.2 Numeric Type and Comparison Operator

- When comparison operators are applied, they return bool values.

```
1 n = int(input('Enter an integer: '))
2 print('Is this an even number?', n % 2 == 0)
```

```
Enter an integer: 60
Is this an even number? True
```

**Line 2**

- If the remainder of dividing a number by 2 is 0, it returns True; otherwise, it returns False.

## One Step Further

- Python allows the use of more than one comparison operator, as shown in the following code.
- In the code below, since n is 100, 0 < n < 200 evaluates to True.

```
1 n = 100
2 print('n = ', n)
3 print(0 < n < 200)
```

```
n =  100
True
```

# 2. Comparison Operators

## 2.3 String and Comparison Operator

- The == operator for strings returns True only if the two strings are exactly the same.

```
1 print('aaa' == 'aaa')
2 print('aaa' == 'bbb')
```

```
True
False
```

# 2. Comparison Operators

## 2.3 String and Comparison Operator

- The == operator for strings returns True only if the two strings are exactly the same.

```
1 print('aaa' == 'aaa')
2 print('aaa' == 'bbb')
```

```
True
False
```

**Line 1**

- The two strings are exactly the same, so it returns True.

# 2. Comparison Operators

## 2.3 String and Comparison Operator

- The == operator for strings returns True only if the two strings are exactly the same.

```
1 print('aaa' == 'aaa')
2 print('aaa' == 'bbb')
```

```
True
False
```

**Line 2**

- The two strings are not the same, so it returns False.

# 2. Comparison Operators

## 2.3 String and Comparison Operator

- The != operator for strings returns True if the two strings are not exactly the same.

```
1 print('aaa' != 'aaa')
2 print('aaa' != 'bbb')
```

```
False
True
```

# 2. Comparison Operators

## 2.3 String and Comparison Operator

- The != operator for strings returns True if the two strings are not exactly the same.

```
1 print('aaa' != 'aaa')
2 print('aaa' != 'bbb')
```

```
False
True
```

**Line 1**
- The two strings are exactly the same, so it returns False.

# 2. Comparison Operators

## 2.3 String and Comparison Operator

- The != operator for strings returns True if the two strings are not exactly the same.

```
1 print('aaa' != 'aaa')
2 print('aaa' != 'bbb')
```

```
False
True
```

**Line 2**
- The two strings are not the same, so it returns True.

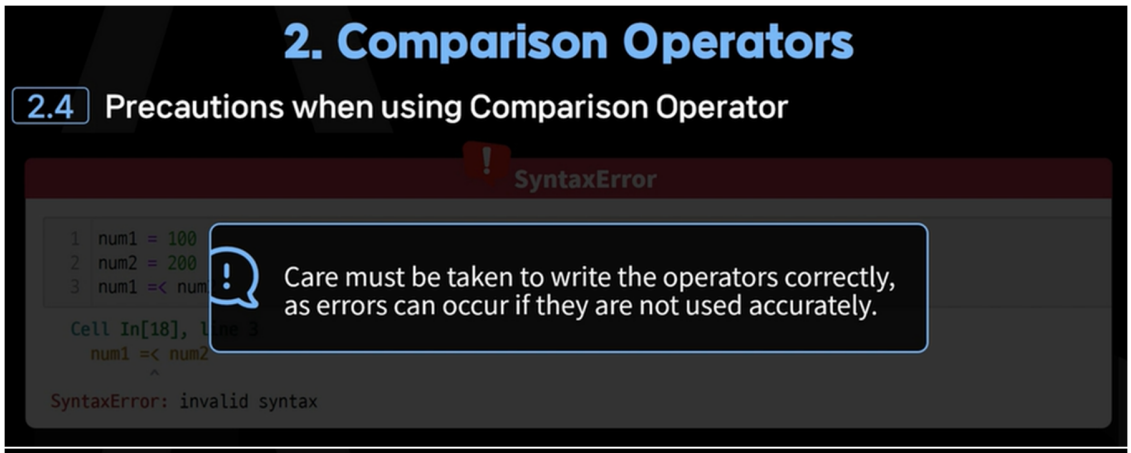
# 2. Comparison Operators

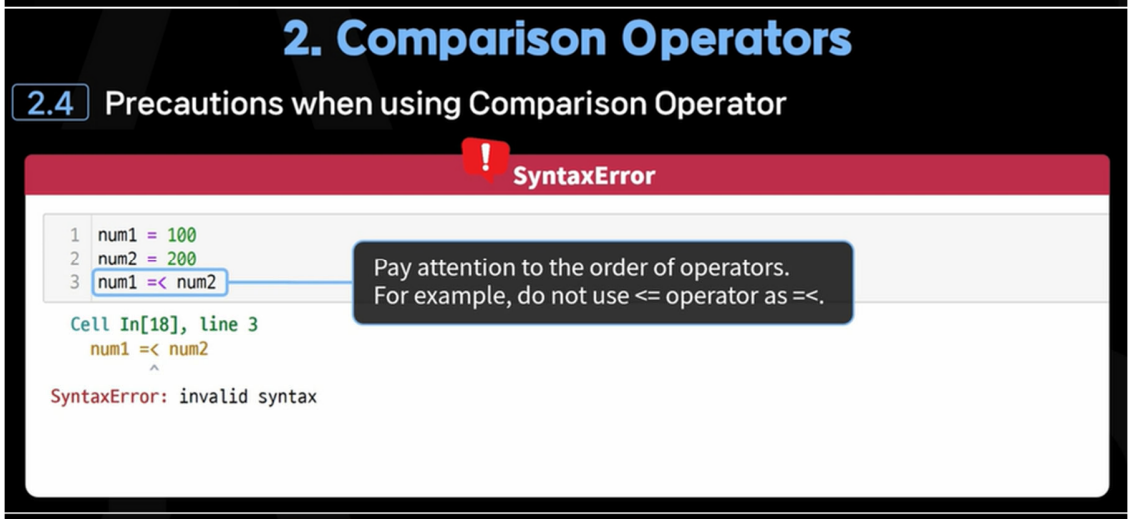
## 2.4 Precautions when using Comparison Operator

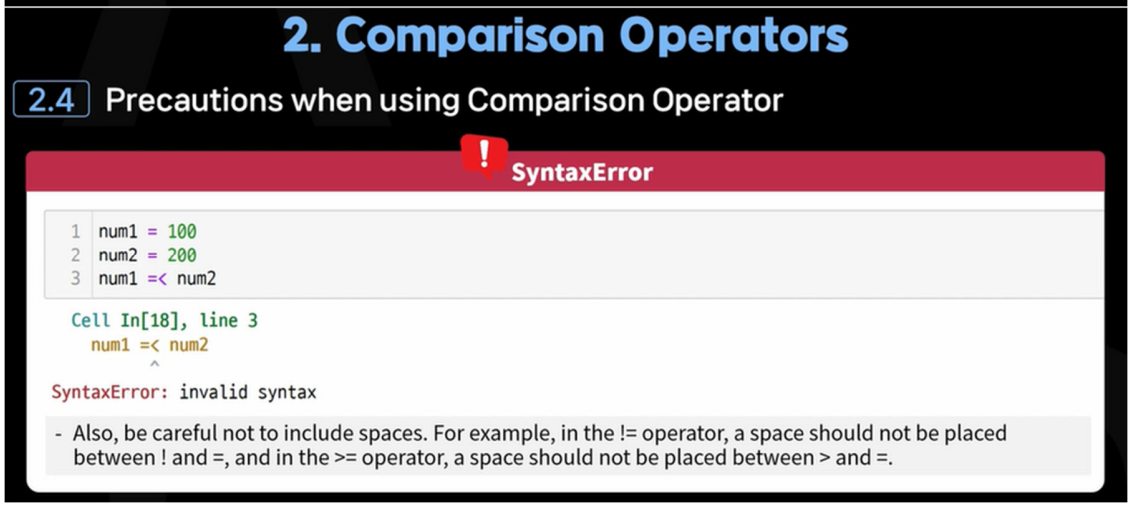
**SyntaxError**

```
1 num1 = 100
2 num2 = 200
3 num1 =< num2
```

```
Cell In[18], line 3
  num1 =< num2
       ^
SyntaxError: invalid syntax
```


2. Comparison Operators
2.4 Precautions when using Comparison Operator
SyntaxError
num1 = 100
num2 = 200
num1 =< num
Care must be taken to write the operators correctly, as errors can occur if they are not used accurately.
num1 =< num2
SyntaxError: invalid syntax


2. Comparison Operators
2.4 Precautions when using Comparison Operator
SyntaxError
num1 = 100
num2 = 200
num1 =< num2
Pay attention to the order of operators. For example, do not use <= operator as =<.
Cell In[18], line 3
num1 =< num2
SyntaxError: invalid syntax


2. Comparison Operators
2.4 Precautions when using Comparison Operator
SyntaxError
num1 = 100
num2 = 200
num1 =< num2
Cell In[18], line 3
num1 =< num2
SyntaxError: invalid syntax
- Also, be careful not to include spaces. For example, in the != operator, a space should not be placed between ! and =, and in the >= operator, a space should not be placed between > and =.

# 2. Comparison Operators

## 2.4 Precautions when using Comparison Operator

- We learned that '=' is an assignment operator. It does not mean equality.
  Be careful not to confuse it with the comparison operator '=='!

**SyntaxError**

```
1 300 = 300
```

```
Cell In[3], line 1
  300 = 300
  ^
SyntaxError: cannot assign to literal here. Maybe you meant '==' instead of '='?
```

- '=' means that the value on the right side is assigned to the left side. It does not indicate equality.
- Therefore, an error occurs because you cannot assign the value 300 to the value 300 in the example.

# 3. Logical Operators

## 3.1 What is Logical Operator?

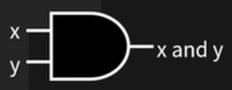
- Logical operators evaluate logical expressions and return True or False.

The symbols used in this figure represent logic gates commonly used in electrical and computer engineering. Each symbol represents 'and', 'or', and 'not', respectively.

| x | y | x and y |
|---|---|---|
| False | False | False |
| False | True | False |
| True | False | False |
| True | True | True |

| x | y | x or y |
|---|---|---|
| False | False | False |
| False | True | True |
| True | False | True |
| True | True | True |

| x | not x |
|---|---|
| False | True |
| True | False |



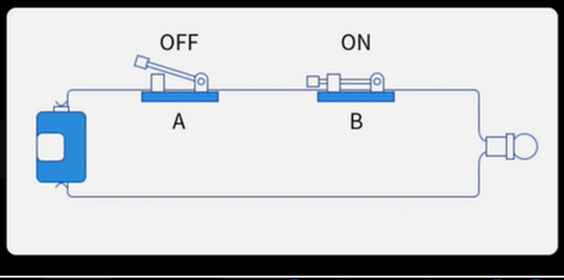


# 3. Logical Operators

## 3.1 What is Logical Operator?

- The 'and' logical operator takes into account the state of False among the input values and returns True.

If any of the input values are OFF, it is False (extinguished).




and

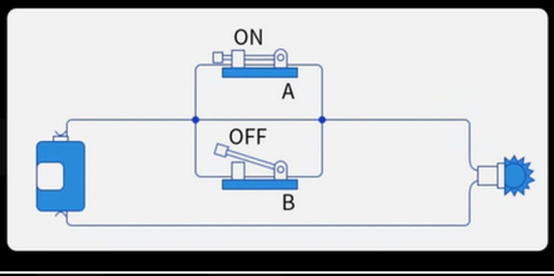
| Input | | Output |
|---|---|---|
| A | B | |
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

# 3. Logical Operators

## 3.1 What is Logical Operator?

- On the other hand, the 'or' logical operator takes into account the state of True among the input values and returns True.

If any of the input values are ON, it is True (illuminated).




or

| Input | | Output |
|---|---|---|
| A | B | |
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

# 3. Logical Operators

## 3.1 What is Logical Operator?

| Operator | Meaning |
|---|---|
| x and y | Returns True if x and y are both True; otherwise, returns False. |
| x or y | Returns True if either x or y is True; returns False only if both are False. |
| not x | Returns False if x is True; returns True if x is False. |

# 3. Logical Operators

## 3.1 What is Logical Operator?

Focus

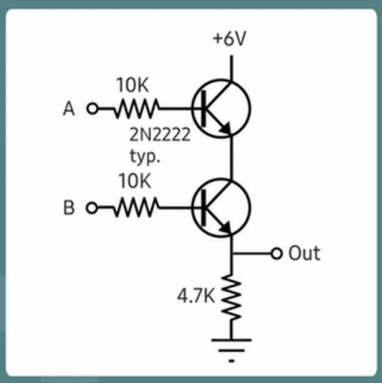
- Logical operators return bool values.

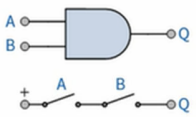
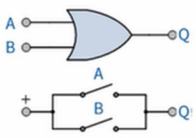
```
1 x = True
2 y = False
3 print(x and y)
4 print(x or y)
5 print(not x)
6 print(not y)
```

```
False
True
False
True
```

## One Step Further

▶ 'and', 'or' logical gate circuits




| Symbol | Truth Table | | |
|---|---|---|---|
|  | B | A | Q |
| | 0 | 0 | 0 |
| | 0 | 1 | 0 |
| | 1 | 0 | 0 |
| | 1 | 1 | 1 |
|  | B | A | Q |
| | 0 | 0 | 0 |
| | 0 | 1 | 1 |
| | 1 | 0 | 1 |
| | 1 | 1 | 1 |

# 3. Logical Operators

## 3.2 Logical Operation

- Logical operations combine bool values to create new bool values.

```
1 num = int(input('Enter an integer : '))
2 result = num >= 0 and num <= 100 and num % 2 == 0
3 print('Is this integer an even number within the range of 0 to 100?', result)
```

```
Enter an integer : 99
Is this integer an even number within the range of 0 to 100? False
```

# 3. Logical Operators

## 3.2 Logical Operation

- Logical operations combine bool values to create new bool values.

```
1 num = int(input('Enter an integer : '))
2 result = num >= 0 and num <= 100 and num % 2 == 0
3 print('Is this integer an even number within the range of 0 to 100?', result)
```

```
Enter an integer : 99
Is this integer an even number within the range of 0 to 100? False
```

**Line 2**

- If the input number is 99, num is greater than 0 and less than 100, so the first and second conditions return True. In the third condition, the remainder of num is 1, so the operation 1 == 0 returns False.
- Since the 'and' logical operator requires all conditions to be True, it ultimately returns False.

# 3. Logical Operators

## 3.3 Logical Operations and Parentheses

- Parentheses can also be used in logical operations to group logical expressions.
- In order to determine whether 'num is divisible by 4 and not by 100, or num is divisible by 400', the expression 'num is divisible by 4 and not by 100' **should be grouped with parentheses.**

```
1 num = 2023
2 (num % 4 == 0 and num % 100 != 0) or num % 400 == 0
```

```
False
```

# 3. Logical Operators

## 3.3 Logical Operations and Parentheses

```
1 num = 2023
2 (num % 4 == 0 and num % 100 != 0) or num % 400 == 0
```

```
False
```

**Line 2**

- The expression '(num % 4 == 0 and num % 100 != 0)' is created by grouping the condition 'num is divisible by 4 and not by 100' with parentheses.
- Then, the condition 'num is divisible by 400' is expressed as 'num % 400 == 0'.
- Finally, these two logical expressions are combined with 'or'.

# 4. Identity Operators

## 4.1 Difference between the 'is' Operator and the == Operator

- The 'is' operator compares the identity of the objects by comparing their identifiers and returns True or False indicating whether the two variables refer to the same object.

```
1 a = 1
2 b = 1.0
3 print(a == b)
4 print(a is b)
```

```
True
False
```

# 4. Identity Operators

## 4.1 Difference between the 'is' Operator and the == Operator

- The 'is' operator compares the identity of the objects by comparing their identifiers and returns True or False indicating whether the two variables refer to the same object.

```
1 a = 1
2 b = 1.0
3 print(a == b)
4 print(a is b)
```

```
True
False
```

**Line 3**

- It is a comparison of **values** using the == operator. The two **values** are equal, so it returns True.

# 4. Identity Operators

## 4.1 Difference between the 'is' Operator and the == Operator

- The 'is' operator compares the identity of the objects by comparing their identifiers and returns True or False indicating whether the two variables refer to the same object.

```
1 a = 1
2 b = 1.0
3 print(a == b)
4 print(a is b)
```

```
True
False
```

**Line 4**

- It is a comparison of whether the two **objects** are the same. Since 'a' is of integer type and 'b' is of float type, the two **objects** are different. They are located in different memory locations. Therefore, it returns False.

# 5. Membership Operators

## 5.1 The 'in' Operator

- The 'in' operator determines whether the left value is present in the right data and returns True or False.
- If you want to check if there is any matching substring in a string, you can use the in operator to obtain True or False values.

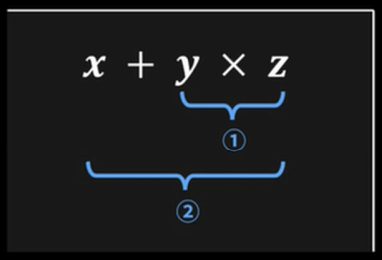
```
1 print('aaa' in 'aaa-bbb-ccc')
2 print('bbb' in 'aaa-bbb-ccc')
3 print('ddd' in 'aaa-bbb-ccc')
```

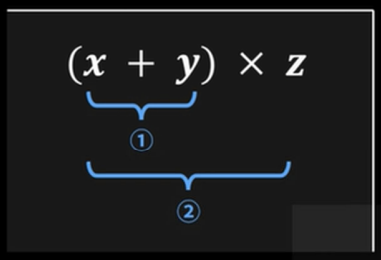
```
True
True
False
```

  - The string 'aaa' is a substring contained in 'aaa-bbb-ccc', so the in operator returns True.

# 6. Order of Operation

## 6.1 Operators have precedence.

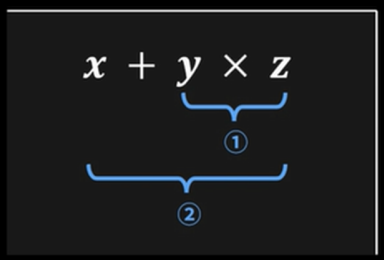


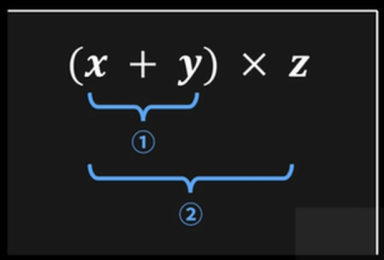




- When writing programs, using multiple operators is common, and the order in which these operators combine is important.
- As we learned in math class, multiplication and division should be performed before addition and subtraction. This is determined by precedence.

# 6. Order of Operation

## 6.1 Operators have precedence.




- It is the rule that determines which operation should be performed first among many operations. Each operator has a ranking.
- Multiplication and division have higher precedence than addition and subtraction.

# 6. Order of Operation

## 6.2 Operator Precedence Table

- Let's examine Python operators and their precedence. These are the concepts we will learn throughout this course.

| Operations | Meaning |
|---|---|
| () | Parentheses |
| ** | Exponentiation |
| ~ , + , - | Unary operators |
| * , / , % , // | Multiplication, division, modulo |
| + , - | Addition, subtraction |
| >> , << | Bit shift operators |

# 6. Order of Operation

## 6.2 Operator Precedence Table

- Let's examine Python operators and their precedence. These are the concepts we will learn throughout this course.

| Operations | Meaning |
|---|---|
| () | Parentheses |
| ** | Exponentiation |
| ~ , + , - | |
| * , / , % , // | division, modulo |
| + , - | Addition, subtraction |
| >> , << | Bit shift operators |

The higher the position in the table, the higher the precedence.

# 6. Order of Operation

## 6.2 Operator Precedence Table

- Let's examine Python operators and their precedence. These are the concepts we will learn throughout this course.

| | |
|---|---|
| & | Bitwise AND operator |
| ^ , \| | Bitwise XOR, bitwise OR operators |
| <= , < , > , >= | Comparison operators |
| == , != | Equality operators |
| = , %= , /= , //= , -= , += , *= , **= | Assignment operators, compound assignment operators |
| is , is not | Identity operators |
| in , not in | Membership operators |

| | |
|---|---|
| ^ , \| | Bitwise XOR, bitwise OR operators |
| <= , < , > , >= | Comparison operators |
| == , != | Equality operators |
| = , %= , /= , //= , -= , += , *= , **= | Assignment operators, compound assignment operators |
| is , is not | Identity operators |
| in , not in | Membership operators |
| not , or , and | Logical operators |

# 6. Order of Operation

## 6.3 Example Code for Operator Precedence

- Like the mathematical equations, parentheses have higher precedence in this case.

```
1 x = int(input("Enter the first number: "))
2 y = int(input("Enter the second number: "))
3 z = int(input("Enter the third number: "))
4
5 avg = (x + y + z) / 3
6 print("Average =", avg)
```

```
Enter the first number: 10
Enter the second number: 20
Enter the third number: 30
Average = 20.0
```

# 6. Order of Operation

## 6.3 Example Code for Operator Precedence

- Like the mathematical equations, parentheses have higher precedence in this case.

```
1 x = int(input("Enter the first number: "))
2 y = int(input("Enter the second number: "))
3 z = int(input("Enter the third number: "))
4
5 avg = (x + y + z) / 3
6 print("Average =", avg)
```

```
Enter the first number: 10
Enter the second number: 20
Enter the third number: 30
Average = 20.0
```

**Line 5**

- If you code x + y + z / 3, it will calculate z / 3 first.
- Therefore, you should use parentheses to perform addition first: (x + y + z) / 3.