1.What are the two values of the Boolean data type? How do you write them?

Answer:

The Boolean data type represents logical values and has two possible values: true and false.

2. What are the three different types of Boolean operators?

Answer:

1. AND operator: This operator is denoted by "&&" (double ampersand) or the word "AND". It returns true if both operands are true, and false otherwise. For example, in the expression "A && B", the result is true only if both A and B are true.
2. OR operator: This operator is denoted by "||" (double vertical bar) or the word "OR". It returns true if at least one of the operands is true, and false if both operands are false. For example, in the expression "A || B", the result is true if either A or B (or both) are true.
3. NOT operator: This operator is denoted by "!" (exclamation mark) or the word "NOT". It is a unary operator that negates the boolean value of its operand. If the operand is true, the NOT operator returns false, and if the operand is false, it returns true. For example, in the expression "!A", if A is true, the result is false; if A is false, the result is true.

3. Make a list of each Boolean operator's truth tables (i.e. every possible combination of Boolean values for the operator and what it evaluate ).

Answer:

1. AND operator:

| **Operand A** | **Operand B** | **A && B** |
| --- | --- | --- |
| false | false | false |
| false | true | false |
| true | false | false |
| true | true | true |

1. OR operator:

| **Operand A** | **Operand B** | **A || B** |
| --- | --- | --- |
| false | false | false |
| false | true | true |
| true | false | true |
| true | true | true |

1. NOT operator:

| **Operand A** | **!A** |
| --- | --- |
| false | true |
| true | false |

4. What are the values of the following expressions?

(5 > 4) and (3 == 5)

not (5 > 4)

(5 > 4) or (3 == 5)

not ((5 > 4) or (3 == 5))

(True and True) and (True == False)

(not False) or (not True)

Answer:

, the values of the given expressions are as follows:

1. (5 > 4) and (3 == 5) => False
2. not (5 > 4) => False
3. (5 > 4) or (3 == 5) => True
4. not ((5 > 4) or (3 == 5)) => False
5. (True and True) and (True == False) => False
6. (not False) or (not True) => True

5. What are the six comparison operators?

Answer:

1. Equal to (==): This operator checks if the values of two operands are equal and returns true if they are equal, and false otherwise. For example, 5 == 5 evaluates to true.
2. Not equal to (!=): This operator checks if the values of two operands are not equal and returns true if they are not equal, and false if they are equal. For example, 5 != 3 evaluates to true.
3. Greater than (>): This operator checks if the value of the left operand is greater than the value of the right operand and returns true if it is, and false otherwise. For example, 5 > 3 evaluates to true.
4. Less than (<): This operator checks if the value of the left operand is less than the value of the right operand and returns true if it is, and false otherwise. For example, 3 < 5 evaluates to true.
5. Greater than or equal to (>=): This operator checks if the value of the left operand is greater than or equal to the value of the right operand and returns true if it is, and false otherwise. For example, 5 >= 5 evaluates to true.
6. Less than or equal to (<=): This operator checks if the value of the left operand is less than or equal to the value of the right operand and returns true if it is, and false otherwise. For example, 3 <= 5 evaluates to true.

6. How do you tell the difference between the equal to and assignment operators?Describe a condition and when you would use one.

Answer:

the equal to operator (==) is used for comparison, while the assignment operator (=) is used for assigning values to variables. The equal to operator is used in conditions to make decisions based on equality, while the assignment operator is used to store and update values in variables.

7. Identify the three blocks in this code:

spam = 0

if spam == 10:

print('eggs')

if spam > 5:

print('bacon')

else:

print('ham')

print('spam')

print('spam')

Answer:

if spam == 10:

print('eggs')

if spam > 5:

print('bacon')

else:

print('ham')

print('spam')

print('spam')

8. Write code that prints Hello if 1 is stored in spam, prints Howdy if 2 is stored in spam, and prints Greetings! if anything else is stored in spam.

Answer:

spam = # Assign the desired value to spam variable

if spam == 1:

print("Hello")

elif spam == 2:

print("Howdy")

else:

print("Greetings!")

9.If your programme is stuck in an endless loop, what keys you’ll press?

Answer:  
If your program is stuck in an endless loop and you want to stop its execution, you can typically press the following key combination:

Ctrl + C

10. How can you tell the difference between break and continue?

Answer:

the "break" statement is used to exit the loop entirely, while the "continue" statement is used to skip the remaining statements in the current iteration and move on to the next iteration of the loop.

11. In a for loop, what is the difference between range(10), range(0, 10), and range(0, 10, 1)?

Answer:

In a for loop, the three expressions **range(10)**, **range(0, 10)**, and **range(0, 10, 1)** generate the same sequence of numbers. Here's the breakdown of their differences:

1. **range(10)**: This expression generates a sequence of numbers starting from 0 (inclusive) up to 10 (exclusive) with a default step size of 1. In other words, it generates numbers from 0 to 9. The start parameter is omitted, so it assumes a default start value of 0.
2. **range(0, 10)**: This expression also generates a sequence of numbers starting from 0 (inclusive) up to 10 (exclusive), just like **range(10)**. The start parameter is explicitly set to 0, indicating that the sequence starts from 0. The step size is omitted, so it assumes a default step size of 1.
3. **range(0, 10, 1)**: Similar to the previous expressions, this one generates a sequence of numbers starting from 0 (inclusive) up to 10 (exclusive). The start parameter is explicitly set to 0, and the step size is explicitly set to 1. In this case, the step size is explicitly mentioned as 1, although 1 is the default value. Specifying the step size explicitly makes the intention clear.

12. Write a short program that prints the numbers 1 to 10 using a for loop. Then write an equivalent program that prints the numbers 1 to 10 using a while loop.

Answer:

# Using a for loop

for i in range(1, 11):

print(i)

# Using a while loop

i = 1

while i <= 10:

print(i)

i += 1

13. If you had a function named bacon() inside a module named spam, how would you call it after importing spam?

Answer:

import spam

spam.bacon()