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

Answer

The Boolean data type represents a logical value and has two possible values: true and false. These values are used to indicate the truth or falsity of a condition.

In most programming languages, the keywords "true" and "false" are used to represent these Boolean values. Some programming languages also provide shortcuts, such as numeric representations, where 1 can be used for true and 0 for false.

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

Answer

The three basic types of Boolean operators are:

(i) AND Operator: The AND operator is used to combine two or more conditions or expressions. It returns true only if all the conditions or expressions are true.

(ii) OR Operator: The OR operator is used to combine two or more conditions or expressions. It returns true if at least one of the conditions or expressions is true.

(iii) NOT Operator: The NOT operator, also known as the negation operator, is used to invert the truth value of a condition or expression. It returns the opposite of the original truth value.

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

Below listed are a few of the Boolean operator’s truth tables:

1. AND Operator

| A | B | A AND B |

|---|---|---------|

| 0 | 0 | 0 |

| 0 | 1 | 0 |

| 1 | 0 | 0 |

| 1 | 1 | 1 |

1. OR Operator

| A | B | A OR B |

|---|---|--------|

| 0 | 0 | 0 |

| 0 | 1 | 1 |

| 1 | 0 | 1 |

| 1 | 1 | 1 |

1. NOT Operator

| A | NOT A |

|---|-------|

| 0 | 1 |

| 1 | 0 |

1. XOR (exclusive OR) Operator

| A | B | A XOR B |

|---|---|---------|

| 0 | 0 | 0 |

| 0 | 1 | 1 |

| 1 | 0 | 1 |

| 1 | 1 | 0 |

1. NAND (NOT AND) Operator

| A | B | A NAND B |

|---|---|----------|

| 0 | 0 | 1 |

| 0 | 1 | 1 |

| 1 | 0 | 1 |

| 1 | 1 | 0 |

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

1. (5 > 4) and (3 == 5)

False

1. not (5 > 4)

False

1. (5 > 4) or (3 == 5)

True

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

False

1. (True and True) and (True == False)

False

1. (not False) or (not True)

True

5. What are the six comparison operators?

Answer

The six comparison operators in most programming languages are:

1. Equality (==): Checks if two values are equal and returns true if they are, and false otherwise.
2. Inequality (!=): Checks if two values are not equal and returns true if they are not, and false otherwise.
3. Greater than (>): Checks if the value on the left is greater than the value on the right and returns true if it is, and false otherwise.
4. Less than (<): Checks if the value on the left is less than the value on the right and returns true if it is, and false otherwise.
5. Greater than or equal to (>=): Checks if the value on the left is greater than or equal to the value on the right and returns true if it is, and false otherwise.
6. Less than or equal to (<=): Checks if the value on the left is less than or equal to the value on the right and returns true if it is, and false otherwise.

These operators are commonly used in conditional statements, loops, and other programming constructs to compare values and make decisions based on the results.

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

Answer

In many programming languages, the equal to operator and the assignment operator are represented by different symbols and serve distinct purposes. Here's how you can tell the difference:

Equal To Operator: The equal to operator is typically represented by two consecutive equal signs (==) in most programming languages. It is used to compare two values or expressions to check if they are equal. The equal to operator evaluates to a Boolean value, either true or false, depending on the comparison result.

For example:

x = 5

y = 7

if x == y:

print("x is equal to y")

else:

print("x is not equal to y")

In this example, the equal to operator (==) compares the values of x and y. Since x is not equal to y, the output will be "x is not equal to y."

Assignment Operator: The assignment operator is used to assign a value to a variable. In most programming languages, the assignment operator is represented by a single equal sign (=). It assigns the value on the right-hand side of the operator to the variable on the left-hand side.

For example:

x = 5

y = x + 2

In this example, the assignment operator (=) is used to assign the value 5 to the variable x. Then, the value of x is used in an expression to calculate a new value for y.

To summarize, you would use the equal to operator (==) when you want to compare two values or expressions to determine if they are equal. On the other hand, you would use the assignment operator (=) when you want to assign a value to a variable.

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

Blocks in a code refer to sections of code that are grouped together and are typically delimited by specific syntax rules.

The three blocks that can be clearly detected here are :

1. between line 1 and line 3
2. between line 4 and line 5
3. between line 6 and line 7

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

def greeting\_based\_on\_spam(spam):

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

On Windows:

* Press "Ctrl + C" in the terminal or command prompt where the program is running.
* If it's a GUI application, you can try closing the window or using the Task Manager to end the process.

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

Answer

In programming, "break" and "continue" are control flow statements used to alter the flow of execution within loops (e.g., for loops, while loops). They serve different purposes and can be easily distinguished based on their functionality:

1. Break:

* The "break" statement is used to exit or terminate the loop prematurely.
* When the "break" statement is encountered within a loop, the loop's execution is immediately halted, and the program continues with the statement following the loop.
* It is typically used to break out of a loop when a specific condition is met or when you want to stop the loop early based on certain criteria.

1. Continue:

* The "continue" statement is used to skip the rest of the loop's body for the current iteration and proceed to the next iteration.
* When the "continue" statement is encountered within a loop, the remaining statements inside the loop for the current iteration are skipped, and the loop moves on to the next iteration.
* It is typically used when you want to skip some specific elements or conditions in the loop but continue with the rest of the iterations.

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

Answer

In Python, the range() function is commonly used in for loops to generate a sequence of numbers. All three of the expressions range(10), range(0, 10), and range(0, 10, 1) are used to generate the same sequence of numbers from 0 to 9, inclusive. However, there are slight differences in their syntax and usage.

1. range(10): This generates a sequence of numbers from 0 up to (but not including) 10. The step value is assumed to be 1 by default. It is equivalent to range(0, 10, 1).
2. range(0, 10): This explicitly specifies the start and end values of the sequence. It generates numbers starting from 0 and ending at (but not including) 10, with a step value of 1.
3. range(0, 10, 1): This explicitly specifies the start, end, and step values of the sequence. It generates numbers starting from 0 and ending at (but not including) 10, with a step value of 1. This is the most explicit form of specifying the range.

In summary, all three forms range(10), range(0, 10), and range(0, 10, 1) are used to generate the same sequence of numbers. The difference lies in how explicitly you want to specify the start, end, and step values of the sequence. The default step value is 1, so if you don't specify a step value, it will be assumed as 1.

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

First using for loop:

for i in range(1, 11):

print(i)

Now using while loop:

num = 1

while num <= 10:

print(num)

num += 1

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

Answer

If you have a function named bacon() inside a module named spam, and you want to call it after importing the spam module, you would use the following syntax:

import spam

spam.bacon()

This assumes that both the spam module and the bacon() function are defined correctly. The import spam statement imports the spam module, and then you can call the bacon() function using spam.bacon() notation.