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

The Boolean data type has two values: **True** and **False**.

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

**Logical AND (&&):** The logical AND operator returns **true** if both of its operands are **true**, and **false** otherwise.

**Logical OR (||):** The logical OR operator returns true if at least one of its operands is true, and false if both operands are false.

**Logical NOT (!):** The logical NOT operator negates the value of its operand. If the operand is true, it returns false, and if the operand is false, it returns 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 evaluates).

1. **Logical AND (&&):**

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

1. **Logical OR (||):**

| A | B | A || B |

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

1. **Logical NOT (!):**

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

4. What are the values of the following expressions?

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

not (5 > 4) = **False**

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

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

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

(not False) or (not True) = **True**

5. What are the six comparison operators?

There are six common comparison operators:

* Equal to (==): This operator checks if two values are equal. If they are equal, it returns true; otherwise, it returns false.
* Not equal to (!=): This operator checks if two values are not equal. If they are not equal, it returns true; otherwise, it returns false.
* Greater than (>): This operator checks if the left operand is greater than the right operand. If it is, it returns true; otherwise, it returns false.
* Less than (<): This operator checks if the left operand is less than the right operand. If it is, it returns true; otherwise, it returns false.
* Greater than or equal to (>=): This operator checks if the left operand is greater than or equal to the right operand. If it is, it returns true; otherwise, it returns false.
* Less than or equal to (<=): This operator checks if the left operand is less than or equal to the right operand. If it is, it returns true; otherwise, it returns false.

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

**Equal to (‘==’):** The equal to operator is used to compare two values for equality. It returns **true** if the values on the left and right sides of the operator are equal and **false** if they are not equal.

Condition:

We use the equal to operator when we need to check whether two values or variables have the same value. For example, in conditional statements, you might use it to determine if a condition is met, such as checking if a user’s input is equal to a predefined value.

Var1 = 5

var2 = 7

result = (var1 == var2)

# result will be **false** because var1 is not equal to var2

**Assignment (‘=’):** The assignment operator is used to assign a value to a variable.

Usage: It takes the value on the right and assigns it to the variable on the left.

Example: X = 10

We use the assignment operator whenever you need to store or update a value in a variable. It is used to initialize variables, update their values, and perform various operations.

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:**

**First block:**

spam = 0

if spam == 10:

print('eggs')

**Second block:**

if spam > 5:

print('bacon')

**Third block:**

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 = 2

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?

Windows: We can press Ctrl + C or Ctrl + Break to interrupt the running program in the Command Prompt or PowerShell. If you're using an integrated development environment (IDE), there might be a "Stop" or "Terminate" button that you can click.

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

**break** and **continue** are two control flow statements used in loops.

**break Statement**:

* Purpose: The **break** statement is used to exit the current loop prematurely. When a **break** statement is encountered, the loop is terminated, and the program continues with the code after the loop.
* Usage: It is typically used to exit a loop based on a certain condition or when a specific task has been accomplished.

**continue Statement:**

* Purpose: The **continue** statement is used to skip the current iteration of the loop and move on to the next iteration. It doesn't exit the loop but rather continues with the next iteration of the loop.
* Usage: It is typically used when you want to skip some iterations of the loop based on a certain condition but continue with the remaining iterations.

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

In a **for** loop, there is no practical difference between **range(10)**, **range(0, 10)**, and **range(0, 10, 1)** in Python.

They all generate the same output from 0 to 9, included. The default values for the **range** function are as follows:

range(start,end,step)

1. **range(10)**: This generates a sequence of numbers starting from 0 (default starting value) up to, but not including, 10. So, it includes values 0 through 9.
2. **range(0, 10)**: This explicitly specifies the starting value (0) and the stopping value (10), with the default step value of 1. It generates the same sequence as **range(10)**.
3. **range(0, 10, 1)**: This is the most explicit form, specifying the starting value (0), stopping value (10), and step value (1). Gives output 0 to 9.

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.

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

The **for** loop iterates over a range of values from 1 to 10 (inclusive), and the **while** loop continues as long as the value of **i** is less than or equal to 10, printing the numbers and incrementing **i** by 1 in each iteration.

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

Answer:

First import the **spam** module using the **import** statement, and then we can access the **bacon()** function

import spam

spam.bacon()