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

The two values of the Boolean data type are "true" and "false".

In some programming languages, the Boolean values can also be represented as "1" for true and "0" for false.

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

Boolean operators are used to combine or manipulate Boolean values (true or false). The most common Boolean operators are:

1. AND operator: The AND operator returns true if both operands are true. It is represented by the symbol "&&" or "and". For example, "true && false" would return false.

2. OR operator: The OR operator returns true if at least one operand is true. It is represented by the symbol "||" or "or". For example, "true || false" would return true.

3. NOT operator: The NOT operator is used to negate a Boolean value. It returns the opposite of the Boolean value. It is represented by the symbol "!" or "not". For example, "!true" would return 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)**

Values of the given Boolean expressions:

1. 1.5 > 4) and (3 == 5) -> (True and False) -> False

2. not (5 > 4) -> not True -> False

3. (5 > 4) or (3 == 5) -> (True or False) -> True

4. not ((5 > 4) or (3 == 5)) -> not (True or False) -> False

5. (True and True) and (True == False) -> (True and False) -> False

6. (not False) or (not True) -> True or False -> True

**5.** **What are the six comparison operators?**

Less than, greater than, less than or equal to, greater than or equal to, equal to **and** not equal to.

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

The equal to operator (==) is used to compare two values and returns true if they are equal, false otherwise. The assignment operator (=), on the other hand, is used to assign a value to a variable

To differentiate between the two, it is important to remember that the equal to operator uses two equal signs (==), while the assignment operator uses only one equal sign (=).

Here's an example to illustrate the difference:

x = 5 # Assigning the value 5 to the variable x

y = 10 # Assigning the value 10 to the variable y

if x == y:

print("x and y are equal")

else:

print("x and y are not equal")

In the code above, we are assigning the value 5 to the variable x and the value 10 to the variable y using the assignment operator. Then, we are using the equal to operator in the if statement to compare the values of x and y. Since x and y are not equal, the code will print "x and y are not equal".

In general, we use the equal to operator (==) when we want to compare two values or expression to check if they are equal. We use the assignment operator (=) when we want to assign a value to a variable or update the value of 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')**

**spam = 0**

if spam == 10:

print('eggs')

if spam > 5:

print('bacon')

else:

print('ham')

print('spam')

print('spam')

Block 1 only contains a single line of code, which is the print('eggs') statement. However, since the condition spam == 10 is not true (since spam is assigned the value 0), this block is skipped and the print('eggs') statement is not executed.

Block 2 starts with the if spam > 5: line and ends with the print('ham') statement. If spam is greater than 5, the print('bacon') statement is executed; otherwise, the print('ham') statement is executed.

Block 3 contains the two print('spam') statements, which are executed regardless of whether the conditions in the previous blocks were true or false.

**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**.

spam = 3

if spam == 1:

print('Hello')

elif spam == 2:

print('Howdy')

else:

print('Greetings!')

In the code above, we first assign the value 3 to the variable spam. Then, we use an if-elif-else statement to check the value of spam and print a message accordingly.

If spam is equal to 1, the first if statement is true and the code inside it (print('Hello')) is executed. If spam is equal to 2, the first if statement is false, but the elif statement is true and the code inside it (print('Howdy')) is executed. If spam is not equal to either 1 or 2, both the if and elif statements are false, and the else statement is executed, printing 'Greetings!'.

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

Ans – we will press CTRL + C

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

**break statement**: It is used to terminate the loop immediately and transfer control to the statement following the loop. Once the break statement is encountered, the loop will not executeany further and the program will continue with the next line of code after the loop.

**continue statement:** It is used to skip the current iteration of the loop and move on to the next iteration. When the continue statement is encountered, the current iteration of the loop is terminated immediately, and the loop proceeds with the next iteration.

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

**range(10):** This specifies a range that starts at 0 and goes up to (but does not include) 10, incrementing by 1 at each step. This is the most common way to use range() and is equivalent to range(0, 10, 1).

**range(0, 10):** This specifies a range that starts at 0 and goes up to (but does not include) 10, incrementing by 1 at each step. The first argument (0) specifies the starting value of the range, and the second argument (10) specifies the ending value.

**range(0, 10, 1):** This specifies a range that starts from 0 and goes up to 11 (but does not include) 10, incrementing by 1 at each step. The first argument (0) specifies the starting value of the range, the second argument (10) specifies the ending value, and the third argument (1) specifies the step size.

**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.**

for I in range(1,11):

print(i)

1

2

3

4

5

6

7

8

9

10

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

**Ans –** import spam

spam.bacon()

Spam is the name of the module and bacon() is the name of the function.