**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 that can be either true or false. In programming languages, the two possible values of the Boolean data type are typically represented as:

True : It show that the value is true or positive condition.

False: It show that the value is false or negative condition.

Ex: Declaring a boolean variable

is\_raining = True

is\_sunny = False

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

**Answer->**The three different types of Boolean operators are **AND operator, OR operator, NOT** **operator.**

**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->** the truth tables for each Boolean operator:

1. AND Operator (**and**):

| **Operand 1** | **Operand 2** | **Result** |
| --- | --- | --- |
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

1. OR Operator (**or**):

| **Operand 1** | **Operand 2** | **Result** |
| --- | --- | --- |
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

1. NOT Operator (**not**):

| **Operand** | **Result** |
| --- | --- |
| True | False |
| False | True |

Basically the NOT operator reverse the true into false and false into true.

**4. What are the values of the following expressions?**

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

true

**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?**

**Answer->**

1. Equal to (==): It checks if the operands on both sides are equal and returns True if they are, and False otherwise.
2. Not equal to ( != ) : It checks if the operands on both sides are not equal and returns True if they are not, and False otherwise.
3. Greater than (>): It checks if the left operand is greater than the right operand and returns True if it is, and False otherwise.
4. Less than (<): It checks if the left operand is less than the right operand and returns True if it is, and False otherwise.
5. Greater than or equal to (>=): It checks if the left operand is greater than or equal to the right operand and returns True if it is, and False otherwise.
6. Less than or equal to (<=): It checks if the left operand is less than or equal to the right operand and returns True if it is, and False otherwise.

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

Answer-> Equal to Operator (==): The equal to operator is used for comparison to check if two values are equal.

Assignment Operator (**=**): The assignment operator is used to assign a value to a variable. It assigns the value on the right side of the operator to the variable on the left side.

the equal to operator (**==**) is used for comparison, while the assignment operator (**=**) is used for assigning values to variables

**7. Identify the three blocks in this code:**

**spam = 0**

**if spam == 10:**

**print('eggs')**

#This block is the first **if** statement block. It checks if the value of **spam** is equal to 10. If it is, the code within the block, which is the **print('eggs')** statement, will be executed.

**if spam > 5:**

**print('bacon')**

**else:**

**print('ham')**

This block is the second **if-else** statement block. It checks if the value of **spam** is greater than 5. If it is, the code within the **if** block, which is the **print('bacon')** statement, will be executed. Otherwise, if the condition is not met, the code within the **else** block, which is the **print('ham')** statement, will be executed.

**print('spam')**

**print('spam')**

This block consists of two **print** statements. These statements are not conditional and will always be executed regardless of the outcomes of the preceding **if** and **if-else** statements.

**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->**  here is my code

spam = 1

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->**  **Ctrl + C (Windows/Linuix)-** Pressing Ctrl and C simultaneously interrupts the running program and typically terminates its execution

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

**Answer->**  break statement:

* The **break** statement is used to immediately exit the loop and terminate its execution.
* When the **break** statement is encountered within a loop, the control jumps out of the loop, and the program continues with the next statement after the loop.
* It is often used when a specific condition is met, and you want to exit the loop prematurely.
* After encountering a **break** statement, the loop is not executed further, and the program continues with the code following the loop.

**continue** statement:

* The **continue** statement is used to skip the rest of the current iteration and move to the next iteration of the loop.
* When the **continue** statement is encountered within a loop, the control jumps to the beginning of the loop for the next iteration, skipping the remaining code in the current iteration.
* It is often used when you want to skip certain iterations based on a specific condition but continue looping.

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

**Answer->**  **range(10):** This specifies a range starting from 0 (default start value) and ending at 10 (exclusive). The step value is implicitly set to 1 (default step value). So, range(10) generates the numbers 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

**range(0, 10):** This explicitly specifies the start value as 0 and the end value as 10 (exclusive). Similar to range(10), the step value is implicitly set to 1. Therefore, range(0, 10) also generates the numbers 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

**range(0, 10, 1):** This explicitly specifies the start value as 0, the end value as 10 (exclusive), and the step value as 1. Since the step value is explicitly set to 1 (same as the default), it behaves the same as the previous examples. It generates the numbers 0, 1, 2, 3, 4, 5, 6, 7, 8, and 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.**

**Answer->**

for i in range(1, 11):

print(i)

output 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?**

**Answer->**

import spam

spam.bacon()