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

The Boolean data type represents logical values, and in Python, there are two possible Boolean values: True and False. These values are used to express the truth or falsity of a condition or statement. In Python, the Boolean values are written as True and False, with the first letter capitalized.

is\_raining = True

is\_sunny = False

if is\_raining:

print("Take an umbrella.")

if not is\_sunny:

print("Wear a jacket.")

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

AND Operator: The and operator returns True if both operands evaluate to True, and False otherwise. It performs a logical AND operation on the two Boolean expressions.

OR Operator: The or operator returns True if at least one of the operands evaluates to True, and False if both operands are False. It performs a logical OR operation on the two Boolean expressions.

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 ).

AND Operator (&& or and):

Operand 1 Operand 2 Result

True True True

True False False

False True False

False False False

OR Operator (|| or or):

Operand 1 Operand 2 Result

True True True

True False True

False True True

False False False

NOT Operator (not):

Operand Result

True False

False True

4. What are the values of the following expressions?

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

(5 > 4) is True because 5 is greater than 4.

(3 == 5) is False because 3 is not equal to 5.

The and operator returns False if any operand is False.

Therefore, (5 > 4) and (3 == 5) evaluates to False.

not (5 > 4)

(5 > 4) is True because 5 is greater than 4.

The not operator negates the Boolean value.

Therefore, not (5 > 4) evaluates to False.

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

(5 > 4) is True because 5 is greater than 4.

(3 == 5) is False because 3 is not equal to 5.

The or operator returns True if any operand is True.

Therefore, (5 > 4) or (3 == 5) evaluates to True.

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

(5 > 4) or (3 == 5) evaluates to True (as explained in the previous step).

The not operator negates the Boolean value.

Therefore, not ((5 > 4) or (3 == 5)) evaluates to False

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

(True and True) is True because both operands are True.

(True == False) is False because True is not equal to False.

The and operator returns False if any operand is False.

Therefore, (True and True) and (True == False) evaluates to False.

(not False) or (not True)

not False is True because it negates the value False.

not True is False because it negates the value True.

The or operator returns True if any operand is True.

Therefore, (not False) or (not True) evaluates to True.

5. What are the six comparison operators?

Equal to (==), Not equal to (!=), Greater than (>), Less than (<), Greater than or equal to (>=), Less than or 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.

To differentiate between the equal to (==) operator and the assignment (=) operator in Python, consider the following:

The equal to operator (==) is used to compare two values and checks if they are equal.

The assignment operator (=) is used to assign a value to a variable. It assigns the value on the right-hand side to the variable on the left-hand side.

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')

Block 1:

if spam == 10:

print('eggs')

This block is an if statement block. It checks if the value of the variable spam is equal to 10. If it is, it will execute the indented code block, which contains the print('eggs') statement.

Block 2:

if spam > 5:

print('bacon')

This block is also an if statement block. It checks if the value of the variable spam is greater than 5. If it is, it will execute the indented code block, which contains the print('bacon') statement.

Block 3:

else:

print('ham')

print('spam')

print('spam')

This block is an else statement block associated with the previous if statement block. It executes when the condition of the if statement (spam > 5) is not met. It includes the indented code block with the print('ham') statement.

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.

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?

Ctrl + C: This key combination is commonly used to send an interrupt signal to the running program. Pressing Ctrl + C usually terminates the program and brings you back to the command prompt or stops the execution in the development environment.

Ctrl + Break: On some systems, such as Windows, you can use Ctrl + Break to interrupt the running program.

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

Break Statement:

The break statement is used to exit or terminate the current loop entirely. When the break statement is encountered within a loop, the loop is immediately exited, and the program continues with the next statement after the loop.

The break statement is often used when a specific condition is met, and there is no need to continue the loop further.

Continue Statement:

The continue statement is used to skip the rest of the current iteration and move on to the next iteration of the loop. When the continue statement is encountered within a loop, the remaining code within the loop for that iteration is skipped, and the program proceeds to 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 expression creates a sequence of numbers starting from 0 (inclusive) and ending at 10 (exclusive) with a step size of 1. The default start value is 0, and the default step size is 1, so range(10) is equivalent to range(0, 10, 1). It generates the numbers 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

range(0, 10): This expression specifies both the start and end values explicitly. It creates a sequence of numbers starting from 0 (inclusive) and ending at 10 (exclusive) with a step size of 1. It generates the same numbers as range(10): 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

range(0, 10, 1): This expression specifies the start, end, and step values explicitly. It creates a sequence of numbers starting from 0 (inclusive) and ending at 10 (exclusive) with a step size of 1. Since the step size is explicitly set to 1, it generates the same numbers as range(10): 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.

for i in range(10):

print(i)

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?

from spam import bacon

bacon()