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

Ans:

The two values of the Boolean data type in Python are True and False. These values represent the two possible states of Boolean logic: true and false, respectively.

In Python, you write these Boolean values exactly as shown: True and False, with the first letter capitalized and the rest in lowercase. These values are keywords in Python and are used to represent truth values in logical expressions, conditions, and comparisons.

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

Ans:

The three different types of Boolean operators in Python are:

AND Operator (and):

The and operator returns True if both operands are true; otherwise, it returns False.

Example:

x = True

y = False

print(x and y) # Output: False

OR Operator (or):

The or operator returns True if at least one of the operands is true; otherwise, it returns False.

Example:

x = True

y = False

print(x or y) # Output: True

NOT Operator (not):

The not operator returns the opposite of the operand's value. If the operand is True, it returns False, and if the operand is False, it returns True.

Example:

x = True

print(not x) # Output: False

These Boolean operators are used to combine Boolean expressions and are fundamental in controlling the flow of program execution, making decisions, and implementing logical conditions in Python code.

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).

Ans:

Following are the truth tables for each Boolean operator:

AND Operator (and):

A B A and B

True True True

True False False

False True False

False False False

OR Operator (or):

A B A or B

True True True

True False True

False True True

False False False

NOT Operator (not):

A not A

True False

False True

These truth tables illustrate the result of applying each Boolean operator to all possible combinations of Boolean values (True and 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)

Ans:

Evaluation of each expression:

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

(5 > 4) is True.

(3 == 5) is False.

True and False evaluates to False.

not (5 > 4):

(5 > 4) is True.

not True evaluates to False.

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

(5 > 4) is True.

(3 == 5) is False.

True or False evaluates to True.

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

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

not True evaluates to False.

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

True == False is False.

True and False evaluates to False.

(not False) or (not True):

not False is True.

not True is False.

True or False evaluates to True.

So, the values of the expressions are:

1. False
2. False
3. True
4. False
5. False
6. True

5. What are the six comparison operators?

Ans:

The six comparison operators in Python are:

Equal to (==):

Checks if two values are equal.

Example: 5 == 5 evaluates to True.

Not equal to (!=):

Checks if two values are not equal.

Example: 5 != 3 evaluates to True.

Greater than (>):

Checks if the left operand is greater than the right operand.

Example: 5 > 3 evaluates to True.

Less than (<):

Checks if the left operand is less than the right operand.

Example: 3 < 5 evaluates to True.

Greater than or equal to (>=):

Checks if the left operand is greater than or equal to the right operand.

Example: 5 >= 5 evaluates to True.

Less than or equal to (<=):

Checks if the left operand is less than or equal to the right operand.

Example: 3 <= 5 evaluates to True.

These comparison operators are used to compare values and expressions, and they return Boolean values (True or False) based on the result of the comparison.

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

Ans:

In Python, the equal to operator (==) is used for comparison, checking if two values are equal, while the assignment operator (=) is used for assigning a value to a variable.

Here's how you can differentiate between them:

Equal to Operator (==):

This operator is used to compare two values to check if they are equal.

Example: x == y checks if the value of x is equal to the value of y.

It returns True if the values are equal, and False otherwise.

Assignment Operator (=):

This operator is used to assign a value to a variable.

Example: x = 5 assigns the value 5 to the variable x.

It does not return any value; it simply assigns the value on the right-hand side to the variable on the left-hand side.

Here's a scenario where you would use each:

Condition where you would use the equal to operator (==):

You might use the equal to operator in a conditional statement to check if a condition is met. For example:

x = 10

if x == 10:

print("x is equal to 10")

Here, the if statement checks if the value of x is equal to 10 using the equal to operator (==).

Condition where you would use the assignment operator (=):

You would use the assignment operator when you want to assign a value to a variable. For example:

x = 5

This statement assigns the value 5 to the variable x.

In summary, 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')

if spam > 5:

print('bacon')

else:

print('ham')

print('spam')

print('spam')

Ans:

spam = 0

if spam == 10:

print('eggs') # Block 1 (if statement block)

if spam > 5:

print('bacon') # Block 2 (if statement block)

else:

print('ham') # Block 3 (else statement block)

print('spam') # Not in any specific block

print('spam') # Not in any specific block

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.

Ans:

spam = 3

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?

Ans:

Ctrl + C

This key combination sends a KeyboardInterrupt signal to the Python interpreter, causing it to stop the execution of the program immediately, regardless of where it is in the code. This is a common method to break out of an infinite loop or halt a program that is not responding.

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

Ans:

"break" is used to exit the loop completely, while "continue" is used to skip the current iteration and proceed to the next iteration of the loop.

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

Ans:

In Python, range() is a built-in function used to generate a sequence of numbers, typically used in for loops. Let's look at the differences between range(10), range(0, 10), and range(0, 10, 1):

range(10):

This generates a sequence of numbers from 0 up to (but not including) 10.

It starts from 0 by default and goes up to 9.

The syntax is range(stop), where stop is the upper limit of the sequence (exclusive).

Example:

for i in range(10):

print(i)

Output: 0 1 2 3 4 5 6 7 8 9

range(0, 10):

This also generates a sequence of numbers from 0 up to (but not including) 10.

It explicitly specifies the starting point (0) and the ending point (10) of the sequence.

The syntax is range(start, stop), where start is the starting point (inclusive) and stop is the upper limit of the sequence (exclusive).

Example:

for i in range(0, 10):

print(i)

Output: 0 1 2 3 4 5 6 7 8 9

range(0, 10, 1):

This also generates a sequence of numbers from 0 up to (but not including) 10.

It explicitly specifies the starting point (0), the ending point (10), and the step size (1) of the sequence.

The syntax is range(start, stop, step), where start is the starting point (inclusive), stop is the upper limit of the sequence (exclusive), and step is the increment between each number in the sequence.

In this case, since the step size is 1, it generates consecutive numbers.

Example:

for i in range(0, 10, 1):

print(i)

Output: 0 1 2 3 4 5 6 7 8 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.

Ans:

# 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

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()

This code first imports the spam module, and then it calls the bacon() function from within the spam module using dot notation (spam.bacon()). This way, you explicitly specify that you're calling the bacon() function that belongs to the spam module.