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.
* They are written without quotes, directly as keywords in Python.
* Examples:

a = True

b = False

* "0" and "1" are not Boolean values in Python. While 0 and 1 can be used in some contexts to represent False and True, respectively, they are not the same as the Boolean values False and True. In Python, False and True are specific keywords representing the Boolean values.

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

The three different types of Boolean operators in Python are:

1. **AND (and)**: Returns True if both operands are true, otherwise returns False.
2. **OR (or)**: Returns True if at least one operand is true, otherwise returns False.
3. **NOT (not)**: Returns the opposite boolean value of the operand. If the operand is True, not returns False, and if the operand is False, not 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 evaluate ).

Ans.

1. AND(‘and’):

|  |  |  |
| --- | --- | --- |
| A | B | A and B |
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

1. OR(‘or’):

|  |  |  |
| --- | --- | --- |
| A | B | A or B |
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

1. NOT(‘not’):

|  |  |
| --- | --- |
| A | Not A |
| True | False |
| False | True |

4. What are the values of the following expressions?

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

Ans.

(5 > 4) is True

(3 == 5) is False

Hence, (5 > 4) and (3 == 5) is False

not (5 > 4)

Ans.

(5 > 4) is True

Hence, not (5 > 4) is False

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

Ans.

(5 > 4) is True

(3 == 5) is False

Hence, (5 > 4) or (3 == 5) is True

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

Ans.

(5 > 4) is True

(3 == 5) is False

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

Hence, not ((5 > 4) or (3 == 5)) is False

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

Ans.

(True and True) is True

(True == False) is False

Hence, (True and True) and (True == False) is False

(not False) or (not True)

Ans.

Not False is True

Not True is False

Hence, (not False) or (not True) is True

1. What are the six comparison operators?

Ans.

The following six comparison operators are:  
  
a. Equal to (==): Verifies the equality of two operands.  
b. Dissimilar to (!=): determines whether two operands are equal.  
c. Over (>): determines whether the operands on the left and right are greater.  
d. Less than (\): Verifies whether the operand on the left is smaller than the operand on the right.  
e. Above or equivalent to (>=): determines whether the operands on the left and right are greater than or equal.  
f. Less than or equal to (\=): Verifies whether the operands on the left and right are the same or less than each other.

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

Ans.

* To compare two values and determine whether they are equal, use the equal to operator (==) (e.g., if x == 5).
* To assign a value to a variable (e.g., x = 5), use the assignment operator (=).
* Use ==, for instance, to determine whether a condition is true (if x == 5) and = to assign values to variables (x = 5).

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.

Block 1:

spam = 0

Block 2:

if spam == 10:

print('eggs')

if spam > 5:

print('bacon')

else:

print('ham')

Block 3:

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.

Ans.

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?

Ans.

* Press Ctrl + C on your keyboard.
* This key combination sends a "KeyboardInterrupt" signal to terminate the running program.
* Use this when your program is unresponsive due to a loop or infinite recursion.

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

Ans.

break:

* When encountered, break terminates the entire loop prematurely, regardless of any remaining iterations.
* It exits the loop entirely and continues with the code that follows the loop.
* Useful when you want to stop the loop as soon as a condition is met.

continue:

* When encountered, continue skips the current iteration of the loop and proceeds to the next iteration.
* It does not exit the loop but rather skips the rest of the code block for the current iteration and starts the next iteration.
* Useful when you want to skip certain elements or conditions within the loop without terminating the loop entirely.

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

Ans.

1.**range(10)**:

* This generates numbers from 0 to 9 (inclusive).
* It starts from 0 by default (if not specified otherwise) and ends at 9.
* It increments by 1 by default (if not specified otherwise).

Example:

for i in range(10):

print(i)

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

2.**range(0, 10)**:

* This explicitly specifies a start (0) and an end (10) for the sequence.
* It generates numbers from 0 up to 9 (not including 10).
* It increments by 1 by default (if not specified otherwise).

Example:

for i in range(0, 10):

print(i)

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

3.**range(0, 10, 1)**:

* This explicitly specifies a start (0), end (10), and step (1).
* It generates numbers from 0 up to 9 (not including 10), incrementing by 1 each time.
* The step parameter (1 in this case) can be changed to specify a different increment.

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.

For loop:

for i in range(1,11):

print(i)

While loop:

n=10

i=1

while(i<=n):

print(i)

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.

Following the `spam` module's import:   
1. Use dot notation to call the `bacon()` function: `spam.bacon()}.  
2. The `bacon()` function defined in the `spam` module is accessed by doing this.   
  
3. As an illustration, the `bacon()` function from the `spam` module is run by `spam.bacon()`.