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

Ans1)The Boolean data type has two values: True and False. In Python, these values are written as they are, with the first letter capitalized:

* True
* False

These values are used to represent logical truth values in programming.

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

Ans2)The three different types of Boolean operators are:

AND operator (&& or "and" in Python):

The AND operator returns True if both operands are True, otherwise, it returns False.

Example: True and True evaluates to True, True and False evaluates to False.

OR operator (|| or "or" in Python):

* The OR operator returns True if at least one of the operands is True, otherwise, it returns False.
* Example: True or False evaluates to True, False or False evaluates to False.

NOT operator (! or "not" in Python):

* The NOT operator returns the opposite of the operand's value. If the operand is True, NOT returns False; 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 ).

* AND operator (&& or "and" in Python):

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

* OR operator (|| or "or" in Python):

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

* NOT operator (! or "not" in Python):

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

These truth tables illustrate all possible combinations of Boolean values for each operator and the resulting evaluation.

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)

Ans4)(5 > 4) and (3 == 5)

* + (5 > 4) evaluates to True
  + (3 == 5) evaluates to False
  + True and False evaluates to False
* not (5 > 4)
  + (5 > 4) evaluates to True
  + not True evaluates to False
* (5 > 4) or (3 == 5)
  + (5 > 4) evaluates to True
  + (3 == 5) evaluates to False
  + True or False evaluates to True
* not ((5 > 4) or (3 == 5))
  + (5 > 4) evaluates to True
  + (3 == 5) evaluates to False
  + True or False evaluates to True
  + not True evaluates to False
* (True and True) and (True == False)
  + True and True evaluates to True
  + True == False evaluates to False
  + True and False evaluates to False
* (not False) or (not True)
  + not False evaluates to True
  + not True evaluates to False
  + True or False evaluates to True

So, the values of the given expressions are:

* False
* False
* True
* False
* False
* True

5. What are the six comparison operators?

Ans5)The six comparison operators are:

* Equal to (==): Returns True if the operands are equal, otherwise returns False.
* Not equal to (!=): Returns True if the operands are not equal, otherwise returns False.
* Greater than (>): Returns True if the left operand is greater than the right operand, otherwise returns False.
* Less than (<): Returns True if the left operand is less than the right operand, otherwise returns False.
* Greater than or equal to (>=): Returns True if the left operand is greater than or equal to the right operand, otherwise returns False.
* Less than or equal to (<=): Returns True if the left operand is less than or equal to the right operand, otherwise returns False.

These comparison operators are used to compare values and determine the relationship between them in conditional statements, loops, and other programming constructs.

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

Ans6)Equal to operator (==):

* + The equal to operator is used in comparisons to check if two values are equal.
  + It returns True if the values on both sides of the operator are equal, otherwise it returns False.
  + Example: if x == 5: compares the value of variable x with the integer 5.
* 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.
  + Example: x = 5 assigns the value 5 to the variable x.

Here's a condition where you would use each operator:

* If you want to compare two values to check if they are equal, you would use the equal to operator (==). For example, in an if statement to check if a variable x is equal to 5, you would write if x == 5:.
* If you want to assign a value to a variable, you would use the assignment operator (=``). For example, to assign the value 5to a variablex, you would write 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')

Ans7)Each block consists of one or more lines of code, and they are distinguished by their indentation levels. The first block initializes the variable spam and checks if it's equal to 10, the second block checks if spam is greater than 5 and prints 'bacon' if true, otherwise, it prints 'ham', and the third block prints 'spam' twice.

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.

Ans8)

spam = # Assign the value to spam variable

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?

Ans9)If your program is stuck in an endless loop and you need to stop it, you can typically press the following keys to interrupt the execution:

Ctrl + C: This key combination sends an interrupt signal (SIGINT) to the running program, causing it to terminate. In most terminals and command prompts, pressing Ctrl + C will immediately halt the execution of the program, allowing you to regain control.

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

Ans10)

break statement:

* The break statement is used to exit or terminate the loop prematurely.
* When a break statement is encountered inside a loop, the loop is immediately exited, and the program continues executing the code after the loop.
* for i in range(5):
* if i == 3:
* break
* print(i)

OUTPUT

0

1

2

continue statement:

* The continue statement is used to skip the rest of the code inside the loop for the current iteration and proceed to the next iteration of the loop.
* When a continue statement is encountered, the loop immediately jumps to the next iteration, skipping any remaining code in the loop body for the current iteration.
* It's commonly used to skip certain iterations of a loop based on a condition without terminating the entire loop.
* for i in range(5):
* if i == 2:
* continue
* print(i)

OUTPUT

0

1

3

4

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

Ans11)

n Python, the range() function is used to generate a sequence of numbers. Let's examine the differences between range(10), range(0, 10), and range(0, 10, 1) in a for loop:

* range(10):
  + This generates a sequence of numbers from 0 up to (but not including) 10.
  + It starts from 0 by default and increments by 1.
  + The start parameter is omitted, and only the stop parameter (10) is provided.
  + Example usage:
  + python
  + Copy code

for i in range(10):

print(i)

* + Output:
  + python
  + Copy code

0

1

2

...

8

9

* range(0, 10):
  + This also generates a sequence of numbers from 0 up to (but not including) 10.
  + It explicitly specifies the start parameter as 0 and the stop parameter as 10.
  + It increments by 1 by default.
  + Example usage:
  + python
  + Copy code

for i in range(0, 10):

print(i)

* + Output:
  + python
  + Copy code

0

1

2

...

8

9

* range(0, 10, 1):
  + This generates a sequence of numbers from 0 up to (but not including) 10, with a step size of 1.
  + It explicitly specifies the start parameter as 0, the stop parameter as 10, and the step parameter as 1.
  + It increments by 1 since the step parameter is provided as 1 (default).
  + Example usage:
  + python
  + Copy code

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

print(i)

* + Output:
  + python
  + Copy code

0

1

2

...

8

9

In summary, while all three forms of range() produce the same result, they differ in how their parameters are specified. The start and step parameters are optional, with default values of 0 and 1 respectively if not provided.

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.

Ans12)# Using a for loop

for i in range(1, 11):

print(i)

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

Ans13)

If you have a function named bacon() inside a module named spam, and you want to call it after importing the spam module, you would use the dot notation to access the function.

import spam

# Call the bacon() function from the spam module

spam.bacon()