1.What are the two values of the Boolean data type? How do you write them?  
Answer:- Boolean data type in Python has two values: True and False. These values represent the two possible truth values in logic.  
These values are case-sensitive in Python, so you need to write them exactly as shown: True and False. They are keywords in Python, and using lowercase (true or false) will result in a Name Error.

2. What are the three different types of Boolean operators?  
Answer:- The three different types of Boolean operators in Python are:  
AND Operator (and):Returns True if both operands are true.  
OR Operator (or):Returns True if at least one of the operands is true.  
NOT Operator (not):Returns True if the operand is false (negates the operand).

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:-Certainly! Here are the truth tables for each Boolean operator:

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

| **x** | **y** | **x and y** |
| --- | --- | --- |
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

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

| **x** | **y** | **x or y** |
| --- | --- | --- |
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

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

| **x** | **not x** |
| --- | --- |
| True | False |
| False | True |

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)

Answer:-the values of the given expressions are:

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

5. What are the six comparison operators?

Answer:-The six comparison operators in Python are:

1. **Equal to (==):**
   * Checks if the values of two operands are equal.
   * Example: **x == y**
2. **Not equal to (!=):**
   * Checks if the values of two operands are not equal.
   * Example: **x != y**
3. **Greater than (>):**
   * Checks if the value of the left operand is greater than the value of the right operand.
   * Example: **x > y**
4. **Less than (<):**
   * Checks if the value of the left operand is less than the value of the right operand.
   * Example: **x < y**
5. **Greater than or equal to (>=):**
   * Checks if the value of the left operand is greater than or equal to the value of the right operand.
   * Example: **x >= y**
6. **Less than or equal to (<=):**
   * Checks if the value of the left operand is less than or equal to the value of the right operand.
   * Example: **x <= y**

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

1. Answer:- Equal to Operator (==):
   * Purpose: Used to compare the equality of two values or expressions.
   * Example: **x == y** checks if the value of **x** is equal to the value of **y**.
   * Returns: **True** if the values are equal; otherwise, **False**.

Ex:- x = 5 y = 3 result = (x == y) # Evaluates to False

1. **Assignment Operator (=):**
   * Purpose: Used to assign a value to a variable.
   * Example: **x = 5** assigns the value **5** to the variable **x**.
   * Returns: Does not return a value; it is used for assignment.

Ex:- x = 5 # Assigns the value 5 to the variable x

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')

Answer:- spam = 0

if spam == 10:

print('eggs') # Block 1

if spam > 5:

print('bacon') # Block 2

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.

Answer:-

spam = 1 # You can change the value of spam to test different cases

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**

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

Asnwer:-

1. **break Statement:**
   * Purpose: The **break** statement is used to exit a loop prematurely, before its normal termination.
   * When encountered: When the **break** statement is encountered inside a loop, the loop is immediately terminated, and the program continues with the next statement after the loop.
   * Common use: Used when a specific condition is met, and you want to exit the loop early.
2. **continue Statement:**
   * Purpose: The **continue** statement is used to skip the rest of the code inside a loop for the current iteration and proceed to the next iteration.
   * When encountered: When the **continue** statement is encountered, the remaining code inside the loop for the current iteration is skipped, and the loop proceeds with the next iteration.
   * Common use: Used when you want to skip certain iterations based on a condition.

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

Answer:-

1. **range(10):**
   * This generates a sequence of numbers starting from 0 (default start) up to, but not including, 10 (default stop).
   * The default step is 1.
2. **range(0, 10):**
   * This explicitly specifies the start as 0, and it generates a sequence of numbers from 0 up to, but not including, 10.
   * The default step is 1.
3. **range(0, 10, 1):**
   * This explicitly specifies the start as 0, stop as 10, and step as 1.
   * While the step is explicitly mentioned as 1, it is the default step value, and it could be omitted.

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:-

# Using a for loop

for i in range(1, 11):

print(i)

Output: 1 2 3 4 5 6 7 8 9 10

# Using a while loop

counter = 1

while counter <= 10:

print(counter)

counter += 1

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()