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

**Ans:**

The Boolean data type has two values: True and False. In Python, these values are case-sensitive, so they must be written exactly as shown here—beginning with an uppercase "T" for True and an uppercase "F" for False. It's important to note that these are reserved keywords in Python, and attempting to use lowercase versions like true or false will result in a 'NameError'.

is\_true = True

is\_false = False

# Using Boolean values in expressions

result = is\_true and is\_false #{This will evaluate to False}

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

**Ans:**

**AND (Logical AND):**

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

Example:

result = True and False # This will evaluate to False

**OR (Logical OR):**

The or operator returns True if at least one of the operands is True, and False if both are False.

Example:

result = True or False # This will evaluate to True

**NOT (Logical NOT):**

The not operator returns the opposite Boolean value of its operand. If the operand is True, not will return False, and vice versa.

Example:

result = not True # This will evaluate to False

**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:**

**AND (Logical AND) Truth Table:**

| A | B | A AND B |

|-------|-------|---------|

| True | True | True |

| True | False | False |

| False | True | False |

| False | False | False |

The result is True only if both operands are True.

**OR (Logical OR) Truth Table:**

| A | B | A OR B |

|-------|-------|---------|

| True | True | True |

| True | False | True |

| False | True | True |

| False | False | False |

The result is True if at least one of the operands is True.

**NOT (Logical NOT) Truth Table:**

| A | NOT A |

|-------|-------|

| 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)**

**Ans:**

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

not (5 > 4) : False

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

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

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

(not False) or (not True) : True

**5. What are the six comparison operators?**

**Ans:**

**Equal to (==):**

Checks if two values are equal.

Example: 'x == y'

**Not equal to (!=):**

Checks if two values are not equal.

Example: 'x != y'

**Greater than (>):**

Checks if the value on the left is greater than the value on the right.

Example: 'x > y'

**Less than (<):**

Checks if the value on the left is less than the value on the right.

Example: 'x < y'

**Greater than or equal to (>=):**

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

Example: 'x >= y'

**Less than or equal to (<=):**

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

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.**

**Ans:**

The equal to operator (==) and the assignment operator (=) serve different purposes in programming, and they are used in different contexts.

**Equal To Operator (==):**

The equal to operator is a comparison operator used to check if two values are equal.

Example:

x = 5

y = 10

result = (x == y) # This checks if x is equal to y

The expression (x == y) evaluates to False in this example because x is not equal to y.

**Assignment Operator (=):**

The assignment operator is used to assign a value to a variable.

Example:

x = 5 # Assigning the value 5 to the variable x

y = x # Assigning the value of x to the variable y

In this example, the value 5 is assigned to the variable x, and then the value of x is assigned to the variable y.

**Distinguishing Between Them:**

The key difference is in their usage and the context in which they appear.

The equal to operator (==) is used in conditional statements, such as if statements, to compare values and make decisions based on their equality.

The assignment operator (=) is used to assign values to variables, updating or creating variables with new values.

Example of Condition:

# Assign values to variables

a = 10

b = 20

# Check if a is equal to b

if a == b:

print("a is equal to b")

else:

print("a is not equal to b")

**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:**

Issue is with indentation error.

Correct Code:

spam = 0

if spam == 10:

print('eggs')

if spam > 5:

print('bacon')

else:

print('ham')

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 = 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

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

**Ans:  
break statement:**

The break statement is used to exit a loop prematurely, terminating the entire loop when a certain condition is met.

Example:

for i in range(5):

if i == 3:

break

print(i)

Output:

0

1

2

In this example, the loop terminates when i becomes equal to 3. The print(i) statement is not executed for i = 3.

**continue statement:**

The continue statement is used to skip the rest of the code inside a loop for the current iteration and move to the next iteration.

Example:

for i in range(5):

if i == 2:

continue

print(i)

Output:

0

1

3

4

In this example, the print(i) statement is skipped for i = 2, and the loop moves on to the next iteration.

**Summary:**

break is used to completely exit a loop when a specific condition is met.

continue is used to skip the rest of the loop's code for the current iteration and move to the next iteration.

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

**Ans:  
range(10):**

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

It is equivalent to range(0, 10, 1) with the default step size of 1.

Example:

for i in range(10):

print(i)

Output:

0

1

2

3

4

5

6

7

8

9

**range(0, 10):**

This expression explicitly specifies the start and end values of the sequence (start = 0, end = 10), with the default step size of 1.

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 expression explicitly specifies the start, end, and step values of the sequence (start = 0, end = 10, step = 1).

The step size is the difference between consecutive numbers in the sequence.

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**

counter = 1

while counter <= 10:

print(counter)

counter += 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()