Name :- Deepawali. B. Mhaisagar

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 are True and False. In Python, these values are written as keywords with an initial capital letter: True and False.

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

ANS

- Logical AND (and): The logical AND operator returns True if both operands are True, and False otherwise. It evaluates to True only if all conditions are true.
- Logical OR (or): The logical OR operator returns True if at least one of the operands is True, and False if both operands are False. It evaluates to True if any of the conditions are true.
- Logical NOT (not): The logical NOT operator negates the value of an operand. It returns True if the operand is False, and False if the operand is True. It flips the truth value of the expression.

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

Logical AND

Operand1	Operand2	Result
True	True	True
True	False	False
False	True	False
False	False	False

Logical OR

Operand1	Operand2	Result

True	True	True
True	False	True
False	True	True
False	False	False

Logical NOT

Operand1	Result
True	False
False	True

4.. What are the values of the following expressions?

(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 (==)**: The equal to operator compares if two operands are equal and returns True if they are equal, and False otherwise.
- **Not equal to (!=)**: The not equal to operator compares if two operands are not equal and returns True if they are not equal, and False if they are equal.
- **Greater than (>):** The greater than operator compares if the left operand is greater than the right operand and returns True if it is, and False otherwise.
- **Less than (<):** The less than operator compares if the left operand is less than the right operand and returns True if it is, and False otherwise.
- **Greater than or equal to (>=):** The greater than or equal to operator compares if the left operand is greater than or equal to the right operand and returns True if it is, and False otherwise.
- **Less than or equal to (<=):** The less than or equal to operator compares if the left operand is less than or equal to the right operand and returns True if it is, and False otherwise.

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 to compare two values for equality. It checks if the values on both sides of the operator are equal and returns a Boolean value (True or False) indicating the result of the comparison.

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.

To differentiate between the equal to operator and the assignment operator, consider the context in which they are used. If you are comparing two values for equality, you need to use the equal to operator (==). On the other hand, if you want to assign a value to a variable, you need to use the assignment operator (=). A condition is an expression that evaluates to either True or False. When comparing two values for equality or any other comparison, you would use the equal to operator (==). For example, you might use it in an if statement to execute certain code when a condition is met:

```
x = 5

if x == 5:

    print("x is equal to 5")
```

In this case, the condition (x == 5) checks if x is equal to 5, and if it is, the code inside the if statement will be executed and the message "x is equal to 5" will be printed.

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

#Block1

if spam == 10:

        print(‘eggs’)

#Block2

if spam > 5:

        print(‘bacon’)

#Block3

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 = 1

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

On Windows/Linux: Press Ctrl + C

On macOS: Press Command + . (Command key and period key together)

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

ANS

In Python, break and continue are two different keywords used in loop control flow. They have distinct functions and purposes:

break statement: The break statement is used to terminate the execution of the nearest enclosing loop (i.e., for or while loop). When the break statement is encountered within a loop, the loop is immediately exited, and the program continues with the next statement after the loop.

```
for i in range(1, 5):

    if i == 4:

        break

    print(i)
```

In this example, the loop will iterate from 1 to 4. However, when i becomes 4 the break statement is encountered, and the loop is terminated. Therefore, only the numbers 1,2 and 3 will be printed, and the loop will not continue to iterate further.

continue statement: The continue statement is used to skip the remaining code within a loop for the current iteration and move to the next iteration. It allows you to skip certain parts of the loop's code block based on a specific condition without terminating the loop entirely.

```
for i in range(1, 5):

    if i == 4:

        continue

    print(i)
```

In this example, when i becomes 4, the continue statement is encountered. It causes the loop to skip the remaining code for that iteration and move to the next iteration. Therefore, when i is 4, it will not be printed, and the loop will continue with the next value.

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

ANS

range(10)

```
for i in range(10):

   print(i)
```

range(0, 10)

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

   print(i)
```

range(0, 10, 1)

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

   print(i)
```

All three variations of range produce the same sequence of numbers from 0 to 9. The difference lies in how the range values are specified, but their behavior in a for loop remains the same.

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 for loop

```
for i in range(0,11):

        print(i)
```

Using while loop

```
i= 1

While i<=10:

        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

```
import spam

spam.bacon()
```

The module spam is imported using the import statement. The function bacon() within the spam module can be accessed using dot notation (spam.bacon()). This syntax specifies the module name (spam) followed by the function name (bacon()) to call the function.