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

Ans. You can evaluate any expression in Python, and get one of two answers, True or False. When you compare two values, the expression is evaluated and Python returns the Boolean answer.

When you run a condition in an if statement, Python returns True or False:

Boolean data type evaluates values and variables.

print(10 > 9) - True  
print(10 == 9) - False  
print(10 < 9) - False

print(bool("Hello")) - True  
print(bool(15)) - True

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

Ans. A Boolean expression is an expression that yields just the two outcomes: ***true or false***. When we work with multiple Boolean expressions or perform some action on them, we make use of the Boolean operators. Since the Boolean expression reveals true or false, the operations on these expressions also result in either ***"true"*** or ***"false".*** Consequently, there are three types of Boolean operators:

* *The AND operator (&& or "and")*
* *The OR operator (|| or "or")*
* *The NOT operator (not)*

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 Boolean Operator in Python*

The ***AND Boolean operator*** is similar to the bitwise ***AND operator*** where the operator analyses the expressions written on both sides and returns the output.

* *True and True = True*
* *True and False = False*
* *False and True = False*
* *False and False = False*

a = 30

b = 45

if (a > 30 and b == 45):

 print("True")

else:

print("False")

Output- False

***OR Boolean Operator in Python***

The ***OR operator*** is similar to the ***OR bitwise operator***. In the bitwise OR, we were focussing on either of the bit being 1. Here, we take into account if either of the expression is true or not. If at least one expression is true, consequently, the result is true.

* *True or True = True*
* *True or False = True*
* *False or True = True*
* *False or False = False*

a = 25

b = 30

if (a > 30 or b < 45):

 print("True")

else:

print("False")

Output- True

***NOT Boolean Operator in Python***

The ***NOT operator*** reverses the result of the Boolean expression that follows the operator. It is important to note that the NOT operator will only reverse the final result of the expression that ***immediately follows.*** Moreover, the NOT operator is denoted by the keyword ***"not".***

* *not(True) = False*
* *not(False) = True*

a = 2

b = 2

if(not(a == b)):

  print("If Executed")

else:

  print("Else Executed")

output- Else Executed.

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) -

(not False) or (not True) - True

5. What are the six comparison operators?

Ans. We have six of these, including and limited to-

**less than,** <

**greater than, >**

**less than or equal to, <=**

**greater than or equal to, >=**

**equal to, ==**

**and not equal to**. !=

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 mathematics and algebra, = is an **equal to** operator. In programming = is an [assignment operator](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Assignment), which means that it assigns a value to a variable.

X = 5 assigning a value of 5 to the variable.

The equality operator returns true or false based on whether the operands (the values being compared) are equal.

X == 8 ## Returns false

7. Identify the three blocks in this code:

Ans spam = 0

if spam == 10:

print('eggs') # Block - 1

if spam > 5:

print('bacon') # Block -2

else:

print('ham')

print('spam')

print('spam') # Block -3

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 **=** int(input("Input a no."))

**if** spam **==** 1:

print("Hello")

**elif** spam **==** 2:

print("Howdy")

**else** :

print("Greetings!")

Input a no.2

Howdy

9.If your programme is stuck in an endless loop, what keys you’ll press?

Ans-  If program is stuck in endless loop we will press ctrl+c

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

Ans*# use of break*

**for** i **in** range(10):

**if**(i**==**7):

**break**

print(i)

print('Breaked')

*#use of continue*

**for** i **in** range(10):

**if**(i**==**7):

**continue**

print(i)

0

1

2

3

4

5

6

Breaked

0

1

2

3

4

5

6

8

9

The break statement will move the execution outside and just after a loop. The continue statement will move the execution to the start of the loop.

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

for i in range(10):

  print(i)

0

1

2

3

4

5

6

7

8

9

From the above output we can conclude that they all do the same thing. The range(10) call ranges from 0 up to (but not including) 10, range(0, 10) tells the loop to start at 0, and range(0, 10, 1) tells the loop to increase the variable by 1 on each iteration.

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.

*#Use of While Loop*

print("While Loop")

a **=**1

**while** a **<=** 10:

print(a)

a**+=**1

print(i)

While Loop

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?