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

Ans: In computer programs, there are three types of data: text, numbers and Booleans. A Boolean data type is a value that can only be either true or false.

A true Boolean value might indicate that the object is valid (e.g. an email address has been typed correctly). A false Boolean value indicates that the object is invalid and has not been done correctly (e.g. you’ve forgotten to fill out a required field).

Boolean values are true and false values.Boolean values have two possible states: true and false. In binary, these are represented by 1 and 0.

Boolean algebra is a type of math that deals with operations on logical values, including binary variables. It is the foundation for decisions in programs, so it’s important to understand how Booleans work.

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

Ans : AND, OR, NOT

There are three basic Boolean search commands: AND, OR and NOT.

AND searches find all of the search terms. For example, searching on dengue AND malaria AND zika returns only results that contain all three search terms. Very limited results.

OR searches find one term or the other. Searching on dengue OR malaria OR zika returns all items that contain any of the three search terms. Returns a large number of results.

NOT eliminates items that contain the specified term. Searching on malaria NOT zika returns items that are about malaria, but will specifically NOT return items that contain the word zika. This is a way to fine-tune results. Note: sometimes AND NOT is used; serves the same function as 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 :**

A and B

Then: C

This returns the value C, when the values A and B are true. We can represent this using something called a truth table. A truth table is a way of representing every possible input and it’s corresponding output. The truth table for this AND statement looks like this:

A B C

1 1 1

1 0 0

0 1 0

0 0 0

In the truth table, a 1 represents true while a 0 represents false. From looking at this table it is evident that the only time C is true, is when both A and B are true.

There is also an OR statement. The OR statement is true when A OR B is true:

If: A or B

Then: C

Truth table:

A B C

1 1 1

1 0 1

0 1 1

0 0 0

This truth table might be a little different then you were expecting. This is because an OR statement is only false when both input values (A and B) are False.

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

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

Ans : False

not (5 > 4)

Ans : False

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

Ans : True

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

Ans : False

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

Ans : False

(not False) or (not True)

Ans : True

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

**Ans Python has six comparison operators, which are as follows:**

**Less than ( < )**

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

**Greater than (>)**

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

**Equal to ( == )**

**Not equal to ( != )**

**These comparison operators compare two values and return a boolean value, either True or False.**

**these comparison operators to compare both numbers and strings.**

**Less than operator (<)**

**The Less Than operator (<) compares two values and returns True if the value on the left is less than the value on the right. Otherwise, it returns False:**

**left\_value < right\_value**

**The following example uses the Less Than (<) operator to compare two numbers:**

**>>> 10 < 20**

**True**

**>>> 30 < 20**

**False**

**The following example uses the less than operator (<) to compare two strings:**

**>>> 'apple' < 'orange'**

**True**

**>>> 'banana' < 'apple'**

**False**

**The expression 'apple' < 'orange' returns True because the letter a in apple is before the letter o in orange.**

**Similarly, the 'banana' < 'apple' returns False because the letter 'b' is after the letter 'a'.**

**The following example shows how to use the less than operator with variables:**

**>>> x = 10**

**>>> y = 20**

**>>> x < y**

**True**

**>>> y < x**

**False**

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

**The less than or equal to operator compares two values and returns True if the left value is less than or equal to the right value. Otherwise, it returns False:**

**left\_value <= right\_value**

**The following example shows how to use the less than or equal to operator to compare two numbers:**

**>>> 20 <= 20**

**True**

**>>> 10 <= 20**

**True**

**>>> 30 <= 30**

**True**

**And this example shows how to use the less than or equal to operator to compare the values of two variables:**

**>>> x = 10**

**>>> y = 20**

**>>> x <= y**

**True**

**>>> y <= x**

**False**

**Greater than operator (>)**

**The greater than operator (>) compares two values and returns True if the left value is greater than the right value. Otherwise, it returns False:**

**left\_value > right\_value**

**This example uses the greater than operator (>) to compare two numbers:**

**>>> 20 > 10**

**True**

**>>> 20 > 20**

**False**

**>>> 10 > 20**

**False**

**And the following example uses the greater than operator (>) to compare two strings:**

**>>> 'apple' > 'orange'**

**False**

**>>> 'orange' > 'apple'**

**True**

**Greater Than or Equal To operator (>=)**

**The greater than or equal to operator (>=) compares two values and returns True if the left value is greater than or equal to the right value. Otherwise, it returns False:**

**left\_value >= right\_value**

**The following example uses the greater than or equal to operator to compare two numbers:**

**>>> 20 >= 10**

**True**

**>>> 20 >= 20**

**True**

**>>> 10 >= 20**

**False**

**And the following example uses the greater than or equal to operator to compare two strings:**

**>>> 'apple' >= 'apple'**

**True**

**>>> 'apple' >= 'orange'**

**False**

**>>> 'orange' >= 'apple'**

**True**

**Equal To operator (==)**

**The equal to operator (==) compares two values and returns True if the left value is equal to the right value. Otherwise, it returns False :**

**left\_value = right\_value**

**The following example uses the equal to operator (==) to compares two numbers:**

**>>> 20 == 10**

**False**

**>>> 20 == 20**

**True**

**And the following example uses the equal to operator (==) to compare two strings:**

**>>> 'apple' == 'apple'**

**True**

**>>> 'apple' == 'orange'**

**False**

**Not Equal To operator (!=)**

**The not equal to operator (!=) compares two values and returns True if the left value isn’t equal to the right value. Otherwise, it returns False.**

**left\_value != right\_value**

**For example, the following uses the not equal to operator to compare two numbers:**

**>>> 20 != 20**

**False**

**>>> 20 != 10**

**True**

**And the following example uses the not equal to operator to compare two strings:**

**>>> 'apple' != 'apple'**

**False**

**>>> 'apple' != 'orange'**

**True**

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

**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')

**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=24

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:  can stop an infinite loop with CTRL + C .

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

Ans: Both “break” and “continue” are the ‘jump’ statements, that transfer control of the program to another part of the program. The main difference between break and continue is that break is used for immediate termination of loop. On the other hand, ‘continue’ terminate the current iteration and resumes the control to the next iteration of the loop.

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

Ans :

range(stop)

range(start, stop, step)

1) range(10) produce below out put

0

1

2

3

4

5

6

7

8

9

2) range(0, 10) produce same out put as above

0

1

2

3

4

5

6

7

8

9

But here 1st parameter of range function is start and 2nd

Parameter is stop , means 0 is star and 10 is stop

3) range(0, 10, 1) this function also produce same as above out put but here

3rd parameter is step.

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:

for i in range(1,11):

print(i)

With while loop

i = 1

while(i<=10):

print(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 :**

In spam.py

def bacon():

print(“Hello world”)

into another file

from spam import \*

bacon()