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

The two values of Boolean data type are **True** and **False**

This data type can be used as a conditional statement in situations where we have to perform specific operations based on the trueness or falsity of a statement.

For example,

Example 1:

i=10

if type(i)==int:

print("Type of i is int")

In the above code, since the conditional statement type(i)==int is True, the above code will give the following output: **Type of i is int**

Example 2:

i="algorithm"

if type(i)==int:

print("Type of i is int")

else:

print("Type of i is not int")

In the above code, since the conditional statement type(i)==int is False, the above code will give the following output: **Type of i is not int**

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

**AND**, **OR** and **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 ).

| **AND operator** |  |  |
| --- | --- | --- |
| **X** | **Y** | **X AND Y** |
| TRUE | TRUE | TRUE |
| FALSE | TRUE | FALSE |
| TRUE | FALSE | FALSE |
| FALSE | FALSE | FALSE |

| **OR operator** |  |  |
| --- | --- | --- |
| **X** | **Y** | **X OR Y** |
| TRUE | TRUE | TRUE |
| FALSE | TRUE | TRUE |
| TRUE | FALSE | TRUE |
| FALSE | FALSE | FALSE |

| **NOT operator** |  |  |  |
| --- | --- | --- | --- |
| **X** | **Y** | **NOT X** | **NOT Y** |
| TRUE | TRUE | FALSE | FALSE |
| FALSE | TRUE | TRUE | FALSE |
| TRUE | FALSE | FALSE | TRUE |
| FALSE | FALSE | TRUE | 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?

< lesser than

> greater than

<= lesser than

>= greater than

== equal to

!= 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.

The comparison operator equal to is written as **==** whereas the assignment operator equal to is written as **=**

The following code shows the difference between the above 2 operators:

*#the below statement assigns value of 100 to variable i*

i = 100

*#the below statement compares the value of i to 100 and proceeds to next line of code in case the condition is True*

if i == 100:

print("Value of i is 100")

else:

print("Value of i is not 100")

The output of above code would be: Value of i is 100

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')

The 3 blocks in the above code are:

1. if spam == 10:

print('eggs')

1. if spam > 5:

print('bacon')

1. 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.

spam = int(input("Please enter a value: "))

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?

Ctrl + C

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

The difference between break and continue statements can be illustrated by the following codes:

*#Use of break statement:*

for i in range(20):

if i%2 == 0:

print(i)

if i==6:

break

The output of above code would be:

0

2

4

6

In the above code, when the condition i==6 becomes True, then the break statement gets executed which breaks the flow of code & the iteration also ends at i=6. Hence we only see values of i being printed till 6.

*#Use of continue statement:*

for i in range(20):

if i%2 == 0:

if i==6:

continue

print(i)

The output of above code would be:

0

2

4

8

10

12

14

16

18

In the above code, when the condition i==6 becomes True, then the continue statement gets executed which sends the flow of code back to the top of the code & hence the print statement for value of i=6 does not get executed. Hence, in the output, we do not see the value of 6 being printed.

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

There is no difference between range(10), range(0, 10), and range(0, 10, 1). This can be shown through the following codes:

1. for i in range(10):

print(i)

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

print(i)

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

print(i)

All the above 3 pieces of code would give us the same 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.

*#Using for loop:*

for i in range(1,11):

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?

import spam

bacon()