Assignment\_2

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

Answer:

Boolean data type has only two values. It can either be **True** or **False.** It holds the value of **1** and **0** respectively.

It can be used when only two conditions are possible. For instance, to check if Fan is ON or OFF. Here ON can be given the value as **True.**

To write the Boolean data type in Python, only the first letter of the word should be uppercase (**True, False).** Other types of inputs are considered as invalid (true, tRUE, TRUE, etc.)

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

Answer:

There are three types of Boolean operators viz.

1. **and**
2. **or**
3. **not**

These operators work just like the ones in logic gates.

When **and** operator is used, both the variables should be **True**. Or else the result will be **False**.

When **or** operator is used, at least one of the two operators should be **True.** Or else the result will be **False**.

When **not** operator is used, the value in the variables is inverted (**True** becomes **False** and vice-versa).

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 evaluates).

Answer:

For **and** operator:

|  |  |  |
| --- | --- | --- |
| **P** | **q** | **P and q** |
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

For **or** operator:

|  |  |  |
| --- | --- | --- |
| **P** | **q** | **P or q** |
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

For **not** operator:

|  |  |
| --- | --- |
| **P** | **not P** |
| 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)

Answer:

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

True **and** False

**False**

not (5 > 4)

**not** True

**False**

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

True **or** False

**True**

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

not (True **or** False)

not True

**False**

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

True and False

**False**

(not False) or (not True)

True or False

True

5. What are the six comparison operators?

The comparison operators compare two values and output the result in the form of Boolean.

The six operators are:

1. equals to
   1. Represented by ‘==’.
   2. Compares if two values are equal.
   3. E.g., (5 == 3) is **False**
2. not equals to
   1. Represented by ‘!=’.
   2. Compares if two values are not equal.
   3. E.g., (5 != 3) is **True**
3. greater than
   1. Represented by ‘>’.
   2. Compares if first value is greater than the second one.
   3. E.g., (5 > 3) is **True**
4. less than
   1. Represented by ‘<’.
   2. Compares if first value is less than the second one.
   3. E.g., (5 < 3) is **False**
5. greater than or equal to
   1. Represented by ‘>=’.
   2. Compares if first value is greater than or equal to the second one.
   3. E.g., (5 >= 3) is **True**
6. less than or equal to
   1. Represented by ‘<=’.
   2. Compares if first value is less than or equal to the second one.
   3. E.g., (5 <= 3) is **False**

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

Answer:

The comparison operator ‘equals’ is written represented by double equal symbol (==).

The assignment operator is represented by single equal symbol (=).

Consider an instance of code given below:

**op = ‘/’**

**if op == ‘\*’:**

**print(‘The op is \*’)**

Here the first line of the code is an assignment operator as ‘/’ is assigned to op. but on the second line, comparison operator is used. Over there it checks if the value in op is ‘\*’ or not.

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')

Answer:

The above code can be written as:

spam = 0

if spam == 10:

print('eggs')

if spam > 5:

print('bacon')

else:

print('ham')

print('spam')

print('spam')

So the three blocks of code used are **if** block(where spam should be equal to 10), another **if** block(where spam should be greater than 5), and **else** block.

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.

Answer:



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

Answer:

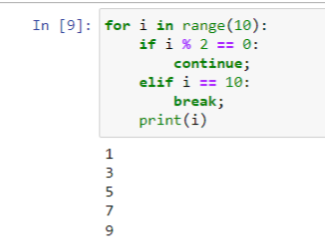
In Jupyter notebook, simply open Kernel tab in Menu, and click ‘Interrupt’ button. Another way to interrupt the kernel is to press **Ctrl + C**.

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

Answer:

**break** is used when the condition is met, and the loop is no longer need to traverse through upcoming values.

**continue** is used when the particular instance of the loop needs to be eliminated.



Here when i is divisible by 2, the loop is eliminated. And when I is equal to 10, the loop ends at that moment.

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

Answer:

A range function consists of 3 constraints. They are start, stop and step.

The range function syntax looks like:

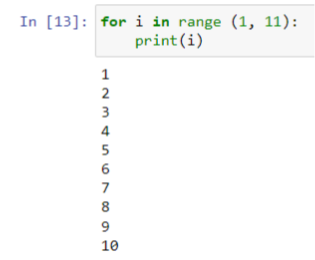
**range(start=0, stop, step=1)**

The start has a default value of 0 and the step has a default value of 1. So there is no difference between range(10), range(0,10), range(0,10,1).

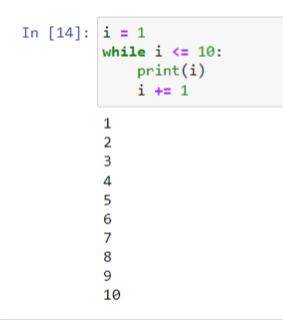
12. Write a short program that prints the numbers 1 to 10 using a for loop. Then write an equivalent program that prints the numbers 1 to 10 using a while loop.

Answer:

Using for loop:



Using while loop:



13. If you had a function named bacon() inside a module named spam, how would you call it after importing spam?

Answer:

import spam

spam.bacon()

OR

import spam as sp

sp.bacon()