1. What are the Boolean data type's two values? How do you go about writing them?

Ans:

Boolean data type is python’s built-in data types. It is used to represent truth of calue of an expression. It has two values as:

1. True

2. False

First letter of both values should be capital other are in small case.

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

Ans:

Boolean Operators:

1. and: If both expressions are true then outcome is true for this operator.

Syntax: x and y, x & y can be any expression.

2. or: Any one of expression is true then outcome is true for this operation

Syntax: x or y, x & y can be any expression.

2. not: It yield true if expression is false

Syntax: not y, y can be any expression.

e.g.

>>> x=10

>>> y=12

>>> x==10 and y==12

True

>>> x!=10 and y==12

False

>>> x!=10 or y==12

True

>>> not y==12

False

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:

Truth table for 'and':

|  |  |  |  |
| --- | --- | --- | --- |
| Expression 1 | and Operator | Expression 2 | Operator Output |
| True | and | True | True |
| True | and | False | False |
| False | and | True | False |
| False | and | False | False |

Truth table for 'or':

|  |  |  |  |
| --- | --- | --- | --- |
| Expression 1 | or Operator | Expression 2 | Operator Output |
| True | or | True | True |
| True | or | False | True |
| False | or | True | True |
| False | or | False | False |

Truth table for 'not':

|  |  |  |
| --- | --- | --- |
| not Operator | Expression | Operator Output |
| not | True | False |
| not | False | True |

4. What are the values of the following expressions?

And:

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

Output: False

Input: not (5 > 4)

Output: False

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

Output: True

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

Output: False

Input: (True and True) and (True == False)

Output: False

Input: (not False) or (not True)

Output: True

5. What are the six different types of reference operators?

Ans:

Operators in Python:

1. Arithmetic operators: used with numeric values

+ Addition x + y

- Subtraction x - y

\* Multiplication x \* y

/ Division x / y

% Modulus x % y

\*\* Exponentiation x \*\* y

// Floor division x // y

2. Assignment operators: used to assign values to varibles

Operator example same as

= x = 5 x = 5

+= x += 3 x = x + 3

-= x -= 3 x = x - 3

\*= x \*= 3 x = x \* 3

/= x /= 3 x = x / 3

%= x %= 3 x = x % 3

//= x //= 3 x = x // 3

\*\*= x \*\*= 3 x = x \*\* 3

&= x &= 3 x = x & 3

|= x |= 3 x = x | 3

^= x ^= 3 x = x ^ 3

>>= x >>= 3 x = x >> 3

<<= x <<= 3 x = x << 3

3. Comparison operators: used to compare two values

Operator

== Equal x == y

!= Not equal x != y

> Greater than x > y

< Less than x < y

>= Greater than or equal to x >= y

<= Less than or equal to x <= y

4. Logical operators: used to combine conditional statement

Operator

and Returns True if both statements are true x < 5 and x < 10

or Returns True if one of the statements is true x < 5 or x < 4

not Reverse the result, returns False if the result is true not(x < 5 and x < 10)

5. Identity operators: used to compare the objects if they are same object, with same memory location.

Operator

is Returns True if both variables are the same object x is y

is not Returns True if both variables are not the same object x is not y

6. Membership operators: used to test if sequence is present in object.

Operator

in Returns True if a sequence with the specified value is present in the object x in y

not in Returns True if a sequence with the specified value is not present in the object x not in y

7. Bitwise operators: used to compare binary numbers.

Operator

&  AND Sets each bit to 1 if both bits are 1

| OR Sets each bit to 1 if one of two bits is 1

^ XOR Sets each bit to 1 if only one of two bits is 1

~ NOT Inverts all the bits

<< Zero fill left shift Shift left by pushing zeros in from the right and let the leftmost bits fall off

>> Signed right shift Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

6. How do you tell the difference between the equal to and assignment operators?

Ans:

equal to operator: It is used to check whether the given operands are equal or not.

It is denoted as ‘==’.

e.g. 1==1 yields True

assignment operators: It is used to assign value to a variables

It is denoted as ‘=’

e.g. x=1, assign value 1 to variable x.

7. Describe a condition and when you would use one.

Ans:

Assignment operator ‘=’ should be used when we want to store a value in variable.

Equal to operator ‘==’ is used when we need to compare two values to yield a result as boolean value (True or False).

8. Recognize the following 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 # Block 1 Starting

if spam == 10: # Block 1 continues

print('eggs') #Block 2 Starting

if spam > 5: #Block 2 continues

print('bacon') #Black 3

else: #Block 2 continues

print('ham') #Block 3

print('spam') #Block 2 Ending

print('spam') #Block 1 Ending

9. Create a programme that prints. If 1 is stored in spam, prints Hello; if 2 is stored in spam, prints Howdy; and if 3 is stored in spam, prints Salutations! if there's something else in spam.

Ans:

spam=int(input("Enter number:\t"))

print("Hello") if spam ==1 else print("Howdy") if spam ==2 else print("Salutations!")

Output:

Enter number: 3

Salutations!

10.If your programme is stuck in an endless loop, what keys can you press?

Ans:

We can use ‘Ctrl + C’ keys to stop infinite loop.

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

Ans:

We will take one example where we are printing hello # (# is from 1 to 10) in while loop with break and continue statements.

Break statement:

i=0

while (i<10):

i=i+1

if i==4:

break

print("Hello", i)

Output:

Hello 1

Hello 2

Hello 3

Continue Statement:

i=0

while (i<10):

i=i+1

if i==4:

continue

print("Hello", i)

Output:

Hello 1

Hello 2

Hello 3

Hello 5

Hello 6

Hello 7

Hello 8

Hello 9

Hello 10

If we look at output of while loop it is printing hello # (# is from 1 to 10).

Due to break and continue statements some changes in output are observed.

In break loop, when i become 4, condition is true and break statement executed after that we come out of loop and next statements are skipped.

In continue loop, when i become 4, condition is true and continue statement executed after that print statement will not be executed and i will be incremented to 5 and loop will be continue till condition for while loop becomes false.

When we call break statement , it breaks execution in between and loop will stop executing.

While we use continue statement, whenever it gets executed statement wrtiien after it will be skipped in loop. Rest of loop will complete its execution.

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

Ans:

1. range(stop) takes one argument

In range(stop), we will get a series of numbers that starts with 0 and includes every whole number up to, excluding number as the stop.

Range(10): [0,1,2,3,4,5,6,7,8,9]

2. range(start, stop) takes two argument

In range(start, stop), we will get a series of whole numbers starts from start to stop-1.

Range(0,10): [0,1,2,3,4,5,6,7,8,9]

3. range(start, stop, step) takes three argument

In range(start, stop, step), we will get a series of whole numbers starts from value in start to value in stop, where as step represents difference between two consecutive numbers.

Range(0,10,1): [0,1,2,3,4,5,6,7,8,9]

13. Using a for loop, write a short programme that prints the numbers 1 to 10 Then, using a while loop, create an identical programme that prints the numbers 1 to 10.

Ans:

Program using 'for' loop:

for i in range(1,11):

print(i)

It print numbers from 1 to 10.

Program using for 'while' loop:

i=1

while i<11:

print(i)

i=i+1

It print numbers from 1 to 10.

14. If you had a bacon() function within a spam module, how would you call it after importing spam?

Ans:

There are several ways to call function:

1. import spam // import module spam.

spam.bacon() // call bacon() function from module spam.

2. from spam import bacon // import function bacon from module spam

bacon() // call function by its name only