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

Answer: The two values of the Boolean data type are "true" and "false". In python programming, these values are typically represented using the keywords "True" and "False", respectively. In many programming languages, the Boolean data type is used to represent logical values and is often used in conditional statements, loops, and other constructs where the program needs to make decisions based on whether a particular condition is true or false.

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

Answer:

The three different types of Boolean operators are ‘AND’, ‘OR’ & ‘NOT’.

‘AND’ operator is represented by the keyword "and" in Python. It returns "True" if and only if both operands are true, and returns "False" in other cases.

‘OR’ operator is represented by the keyword "or" in Python. It returns "True" if either or both of the operands are true, and returns "False" only if both operands are false.

‘NOT’ operator is represented by the keyword "not" in Python. It returns the opposite of the operand's value. If the operand is true, the NOT operator returns False, and if the operand is False, the NOT operator returns True.

3. Make a list of each Boolean operator truth tables (i.e. every possible combination of Boolean values for the operator and what it evaluate).

Answer:

AND operator truth table:

Operand 1	Operand 2	Result
True	True	True
True	False	False
False	True	False
False	False	False

OR operator truth table:

Operand 1	Operand 2	Result
True	True	True
True	False	True
False	True	True

False False False

NOT operator truth table:

Operand	**Result**
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:

i. (5 > 4) and (3 == 5) : False

ii. not (5 > 4) : False

iii. (5 > 4) or (3 == 5) : True

iv. (True and True) and (True == False) : False

v. (not False) or (not True) : True

5. What are the six comparison operators?

Answers:

The six comparison operators are:

1. Greater than (>): Returns True if the left operand is greater than the right operand, and False otherwise. For example, 5 > 3 would evaluate to True.
2. Less than (<): Returns True if the left operand is less than the right operand, and False otherwise. For example, 3 < 5 would evaluate to True.

3. Greater than or equal to (>=): Returns True if the left operand is greater than or equal to the right operand, and False otherwise. For example, 5 >= 5 would evaluate to True.
4. Less than or equal to (<=): Returns True if the left operand is less than or equal to the right operand, and False otherwise. For example, 3 <= 5 would evaluate to True.
5. Equal to (==): Returns True if the left operand is equal to the right operand, and False otherwise. For example, 5 == 5 would evaluate to True.
6. Not equal to (!=): Returns True if the left operand is not equal to the right operand, and False otherwise. For example, 5 != 3 would evaluate to True.

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

Answer:

In programming, the equal to operator (==) is used to compare two values for equality and return a Boolean value indicating whether they are equal or not. On the other hand, the assignment operator (=) is used to assign a value to a variable.

The difference between the two operators is that the assignment operator (=) assigns a value to a variable, while the equal to operator (==) checks whether two values are equal. Here's an example to illustrate the difference:

x = 5 # assign the value 5 to the variable x y = 7 # assign the value 7 to the variable y

if x == y: # compare the values of x and y for equality print("x and y are equal") else: print("x and y are not equal")

In this example, the assignment operator (=) is used to assign the values 5 and 7 to the variables x and y, respectively. Then, the equal to operator (==) is used to compare the values of x and y to check if they are equal.

If x and y have the same value, the program will print "x and y are equal". Otherwise, it will print "x and y are not equal".

In summary, the assignment operator (=) is used to assign a value to a variable, while the equal to operator (==) is used to compare two values for equality.

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:

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:

1. The first block is not indented and contains only one line of code.
2. The second block is indented and contains two conditional statements. This block contains two if statements. The first if statement checks if the value of spam is equal to 10, and if it is, it prints the string 'eggs'. The second if statement checks if the value of spam is greater than 5, and if it is, it prints the string 'bacon'. If the value of spam is not greater than 5, the else statement is executed, and it prints the string 'ham'.
3. The third block is not indented and contains three lines of code: These lines of code print the string 'spam' twice.

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

Answer: Ctrl + C

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

Answer:

In Python, break and continue are two keywords used in loops to alter their normal behavior.

break is used to exit a loop prematurely when a certain condition is met. When the break statement is executed inside a loop, it immediately terminates the loop, and control is transferred to the next statement after the loop. The following example demonstrates the use of the break statement:

```
for i in range(10):
    if i == 5:
        break
    print(i)
```

In this example, the for loop iterates through the numbers 0 to 9, and when the value of i becomes equal to 5, the break statement is executed, causing the loop to terminate prematurely. Therefore, the output of this program will be:

```
0
1
2
3
4
```

On the other hand, continue is used to skip the current iteration of a loop and move on to the next one. When the continue statement is executed inside a loop, the current iteration is immediately terminated, and control is transferred to the next iteration. The following example demonstrates the use of the continue statement:

```
for i in range(10):
    if i % 2 == 0:
        continue
    print(i)
```

In this example, the for loop iterates through the numbers 0 to 9, and when the value of i is even (i.e., divisible by 2), the continue statement is executed, causing the current iteration to be skipped. Therefore, the output of this program will be:

```
1
3
5
7
9
```

In summary, break is used to prematurely terminate a loop, while continue is used to skip the current iteration and move on to the next one.

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

Answer:

In a for loop, range(10), range(0, 10), and range(0, 10, 1) all produce the same result because they generate a sequence of integers from 0 up to (but not including) 10. However, there are some differences in how they work:

range(10): This creates a sequence of integers from 0 to 9. The single argument provided to range represents the number of integers in the sequence. By default, range starts at 0 and increments by 1. Therefore, range(10) is equivalent to range(0, 10, 1)

range(0, 10): This creates a sequence of integers from 0 to 9. Here, the two arguments provided to range represent the start and stop values of the sequence. By default, range increments by 1, so this is equivalent to range(0, 10, 1).

range(0, 10, 1): This creates a sequence of integers from 0 to 9, just like the previous two examples. However, here the three arguments provided to range represent the start value, the stop value, and the step size. In this case, the step size is explicitly set to 1. This is equivalent to range(10) and range(0, 10).

In general, the range function is used to generate a sequence of integers for use in a for loop. The arguments provided to range allow you to customize the starting point, the stopping point, and the step size of the sequence.

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:

```
for i in range(1,11):
        print(i)
while i<11:
        print(i)
```

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

importing spam?

Answer:

spam.bacon()