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

The two values of the Boolean data type are True and False.

True represents a logical truth, while False represents a logical falsehood.

They are written in all caps, and they are not enclosed in quotation marks.

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

Below are the three different types of Boolean operators:

* AND
* OR
* NOT

The **AND** operator takes two Boolean values and returns True if both values are True. For example, the expression True and False evaluates to False because only one of the values is True.

The **OR** operator takes two Boolean values and returns True if at least one of the values is True. For example, the expression True or False evaluates to True because one of the values is True.

The **NOT** operator takes a Boolean value and returns the opposite value. For example, the expression not True evaluates to False because the opposite of True is 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 evaluates).**

**AND**: The AND operator returns True if both values are True. For example, the expression True and True evaluates to True because both values are True. The AND operator returns False if either value is False. For example, the expression True and False evaluates to False because one of the values is False.

**OR:** The OR operator returns True if at least one value is True. For example, the expression True or False evaluates to True because one of the values is True.The OR operator returns False if both values are False. For example, the expression False or False evaluates to False because both values are False.

**NOT:** The NOT operator returns the opposite of the value. For example, the expression not True evaluates to False because the opposite of True is False. The NOT operator returns the opposite of the value. For example, the expression not False evaluates to True because the opposite of False is 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)**

Here is a breakdown of how each expression is evaluated:

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

The first expression evaluates to True because 5 is greater than 4. The second expression evaluates to False because 3 is not equal to 5. The overall expression evaluates to False because the AND operator only returns True if both expressions are True.

**not (5 > 4)**

The expression 5 > 4 evaluates to True. The NOT operator negates the value, so the overall expression evaluates to False.

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

The first expression evaluates to True because 5 is greater than 4. The second expression evaluates to False because 3 is not equal to 5. The overall expression evaluates to True because the OR operator returns True if at least one of the expressions is True.

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

The expression (5 > 4) or (3 == 5) evaluates to True because the OR operator returns True if at least one of the expressions is True. The NOT operator negates the value, so the overall expression evaluates to False.

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

The first expression evaluates to True because both values are True. The second expression evaluates too False because True is not equal to False. The overall expression evaluates to False because the AND operator only returns True if both expressions are True.

**(not False) or (not True)**

The expression not False evaluates to True because False is the opposite of True. The expression not True evaluates to False because True is the opposite of False. The overall expression evaluates to True because the OR operator returns True if at least one of the expressions is True.

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

* Equal to = = Returns True if the two values are equal.
* Not equal to! = Returns True if the two values are not equal.
* Greater than > Returns True if the first value is greater than the second value.
* Greater than or equal to >= Returns True if the first value is greater than or equal to the second value.
* Less than < Returns True if the first value is less than the second value.
* Less than or equal to <= Returns True if the first value is less than or equal to the second value.

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

The equal to operator == is used to compare two values and return a Boolean value. The assignment operator = is used to assign a value to a variable.

Equal to == Returns True if the two values are equal.

Assignment = Assigns a value to a variable.

For example, the expression 5 == 4 evaluates to False because 5 is not equal to 4. The expression a = 5 assigns the value 5 to the variable a.

Below is a condition that we could use:

if a == 5:

print("a is equal to 5")

This condition would print the message "a is equal to 5" if the value of the variable a is equal to 5.

**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')**

**Block 1:**

if spam == 10:

print('eggs')

This block is an if statement that checks if the value of the variable spam is equal to 10. If it is, the message "eggs" is printed.

**Block 2:**

if spam > 5:

print('bacon')

This block is an if statement that checks if the value of the variable spam is greater than 5. If it is, the message "bacon" is printed.

**Block 3:**

else:

print('ham')

This block is an else statement that is executed if the if statement in block 2 evaluates to False. In this case, the message "ham" is printed.

**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.**

Below is the 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 = 1

if spam == 1:

print("Hello")

elif spam == 2:

print("Howdy")

else:

print("Greetings!")

This code first assigns the value 1 to the variable spam. Then, it uses an if statement to check if the value of spam is equal to 1. If it is, the message "Hello" is printed. If the value of spam is not equal to 1, the if statement is skipped. Next, the code uses an elif statement to check if the value of spam is equal to 2. If it is, the message "Howdy" is printed. If the value of spam is not equal to 2, the elif statement is skipped. Finally, the else statement is executed if the if statement and elif statement are both skipped. The else statement prints the message "Greetings!".

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

If my program is stuck in an endless loop, I will press Ctrl+C to interrupt the program and stop the loop. The Ctrl+C keyboard shortcut is a standard way to interrupt a program in most operating systems. When we press Ctrl+C, the operating system sends a signal to the program that tells it to stop what it is doing and return control to the user. If we are using a Python interpreter in a terminal window, we will see a message that says "KeyboardInterrupt" when you press Ctrl+C. This message indicates that the program has been interrupted. Once we have interrupted the program, we can then examine the code to see why it was stuck in an endless loop. Once we have identified the problem, we can fix the code and try running the program again.

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

The break and continue statements are both used in Python to control the flow of a program. However, they have different purposes. The break statement is used to terminate the current loop. For example, the following code will print the numbers from 1 to 10, but it will only print the even numbers because the break statement is used to terminate the loop when the value of i is equal to 6:

for i in range(1, 11):

if i % 2 == 0:

print(i)

else:

break

The continue statement is used to skip the rest of the current iteration of the loop and go to the next iteration. For example, the following code will print the numbers from 1 to 10, but it will skip the number 5:

for i in range(1, 11):

if i == 5:

continue

print(i)

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

The range() function in Python is used to generate a sequence of numbers. The three forms of the range() function are:

range(10)

range(0, 10)

range(0, 10, 1)

The first form of the range() function generates a sequence of numbers from 0 to 9, inclusive. The second form of the range() function generates a sequence of numbers from 0 to 9, inclusive, but it also includes the start value, 0. The third form of the range() function generates a sequence of numbers from 0 to 9, inclusive, and it also includes the step value, 1. Below are the points that summarizes the difference between the three forms of the range() function:

range(10): Generates a sequence of numbers from 0 to 9, inclusive.

range(0, 10): Generates a sequence of numbers from 0 to 9, inclusive, and it also includes the start value, 0.

range(0, 10, 1): Generates a sequence of numbers from 0 to 9, inclusive, and it also includes the step value, 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.**

Below are the two programs:

**# For loop**

for i in range(1, 11):

print(i)

**# While loop**

i = 1

while i <= 10:

print(i)

i += 1

The for loop is a more concise way to write the program, but the while loop is more explicit about the logic of the program. The for loop uses the range() function to generate a sequence of numbers from 1 to 10. The while loop starts with the value 1 and then increments the value of i by 1 until i is greater than or equal to 10. Both programs print the numbers 1 to 10.

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

Below is how we would call the function bacon() inside a module named spam after importing spam:

import spam

spam.bacon() The import statement imports the module spam into the current namespace. This means that the functions and variables defined in the spam module are now available in the current namespace. The spam.bacon() syntax calls the function bacon() from the spam module. The . operator is used to access the function bacon() from the spam module. For example, if the module spam defines the following function:

def bacon():

print("Bacon!")

Then the following code would print the message "Bacon!"

import spam

spam.bacon()