**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**
  + **False**

These values are written with capital letters: True and False.

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

The three main Boolean operators are:

1. **and**: Returns True only if both operands are True.
2. **or**: Returns True if at least one of the operands is True.
3. **not**: Reverses the Boolean value; it returns True if the operand is False, and False if the operand is True.

**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 to).**

**AND (and)**:

| **A** | **B** | **A and B** |
| --- | --- | --- |
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

**OR (or)**:

| **A** | **B** | **A or B** |
| --- | --- | --- |
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

**NOT (not)**:

| **A** | **not A** |
| --- | --- |
| True | False |
| False | True |

**4. What are the values of the following expressions?**

Let's evaluate the Boolean expressions:

* (5 > 4) and (3 == 5)  
  (5 > 4) is True, and (3 == 5) is False.  
  So, True and False evaluates to **False**.
* not (5 > 4)  
  (5 > 4) is True, so not True evaluates to **False**.
* (5 > 4) or (3 == 5)  
  (5 > 4) is True, and (3 == 5) is False.  
  So, True or False evaluates to **True**.
* not ((5 > 4) or (3 == 5))  
  (5 > 4) is True, so the expression inside the parentheses evaluates to True.  
  Therefore, not True evaluates to **False**.
* (True and True) and (True == False)  
  (True and True) is True, and (True == False) is False.  
  So, True and False evaluates to **False**.
* (not False) or (not True)  
  not False is True, and not True is False.  
  So, True or False evaluates to **True**.

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

The six comparison operators in Python are:

1. **==**: Equal to
2. **!=**: Not equal to
3. **>**: Greater than
4. **<**: Less than
5. **>=**: Greater than or equal to
6. **<=**: Less than or 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.**

* **Equal to (==)**: Used in **comparisons** to check if two values are equal.  
  Example: if x == 10: — checks if x is equal to 10.
* **Assignment (=)**: Used to assign a value to a variable.  
  Example: x = 10 — assigns the value 10 to the variable x.

**Condition**: Use == when comparing two values for equality, and use = when you want to assign a value to a variable.

**7. Identify the three blocks in this code:**

python

Copy

spam = 0

if spam == 10:

print('eggs')

if spam > 5:

print('bacon')

else:

print('ham')

print('spam')

print('spam')

* **Block 1**: if spam == 10: and the code inside it (if this condition is true, 'eggs' will be printed).
* **Block 2**: if spam > 5: and the code inside it ('bacon' will be printed if this condition is true).
* **Block 3**: The else block, where 'ham' is printed if the condition spam > 5 is false.

After that, the last two print('spam') are executed because they are outside of the conditional blocks.

**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.**

python

Copy

spam = 1

if spam == 1:

print("Hello")

elif spam == 2:

print("Howdy")

else:

print("Greetings!")

**9. If your program is stuck in an endless loop, what keys would you press?**

To stop a program that is stuck in an endless loop, you would typically press **Ctrl + C**. This sends a signal to interrupt the program.

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

* **break**: Exits the loop entirely, and no further iterations are executed. Example:

python

Copy

for i in range(5):

if i == 3:

break # Exits the loop when i is 3

print(i)

* **continue**: Skips the current iteration of the loop and continues with the next iteration. Example:

python

Copy

for i in range(5):

if i == 3:

continue # Skips the iteration when i is 3

print(i)

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

* **range(10)**: Generates numbers from 0 to 9 (default start value is 0, and the step value is 1).
* **range(0, 10)**: Also generates numbers from 0 to 9. The start is explicitly specified as 0, and the step is 1 (the default).
* **range(0, 10, 1)**: Again generates numbers from 0 to 9, with the same start (0), end (10), and step (1).

In essence, all three are equivalent in this case.

**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 a for loop:**

python

Copy

for i in range(1, 11):

print(i)

**Using a while loop:**

python

Copy

i = 1

while i <= 10:

print(i)

i += 1

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

python

Copy

import spam

spam.bacon() # Calling the bacon() function from the spam module