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

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

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 ).

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)

5. What are the six comparison operators?

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

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')

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.

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

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

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

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

1. The two values of the Boolean data type are True and False, and they are written in title case (i.e., with the first letter capitalized).
2. The three different types of Boolean operators are "and", "or", and "not".
3. The truth tables for the three Boolean operators are as follows:

"and" operator:

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

"or" operator:

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

"not" operator:

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

1. The values of the following expressions are:

* (5 > 4) and (3 == 5) --> False
* not (5 > 4) --> False
* (5 > 4) or (3 == 5) --> True
* not ((5 > 4) or (3 == 5)) --> False
* (True and True) and (True == False) --> False
* (not False) or (not True) --> True

1. The six comparison operators are:

* **>** (greater than)
* **<** (less than)
* **>=** (greater than or equal to)
* **<=** (less than or equal to)
* **==** (equal to)
* **!=** (not equal to)

1. The equal to operator is "==", and it is used to compare two values for equality. The assignment operator is "=", and it is used to assign a value to a variable. You can tell the difference between them by looking at the context in which they are used. For example, if you want to check if two variables are equal, you would use the equal to operator, like this: **if x == y:**, but if you want to assign a value to a variable, you would use the assignment operator, like this: **x = 10**.
2. The three blocks in the code are:

* The first block is the initialization of the variable **spam** to the value 0.
* The second block is an if statement that checks if **spam** is equal to 10, but since it is not, the code inside the if block is not executed.
* The third block is an if-else statement that checks if **spam** is greater than 5. Since it is not, the code inside the else block is executed, which prints "ham", "spam", and "spam" to the console.

1. Here's the code that prints "Hello", "Howdy", or "Greetings!" based on the value of **spam**:

Code:

if spam == 1:

print("Hello")

elif spam == 2:

print("Howdy")

else:

print("Greetings!")

1. If your program is stuck in an endless loop, you can press the "Ctrl" and "C" keys simultaneously to interrupt the program and stop it from running.
2. The "break" statement is used to exit a loop completely, while the "continue" statement is used to skip over an iteration of the loop and go to the next one.
3. The three different ways to use **range()** in a for loop are:

* **range(10)**: This will generate a sequence of numbers from 0 to 9 (i.e., 10 numbers in total). This is the same as using **range(0, 10)**.
* **range(0, 10)**: This will generate a sequence of numbers from 0 to 9 (i.e., 10 numbers in total).
* **range(0, 10, 1)**: This will generate a sequence of numbers from 0 to 9 (i.e., 10 numbers in total), incrementing by 1 each time.

1. Here's a short program that prints the numbers 1 to 10 using a for loop:

for i in range(1,11):

print(i)

And here's an equivalent program that prints the numbers 1 to 10 using a while loop:

i=1

while(i<11):

print(i)

1. If you had a function named **bacon()** inside a module named **spam**, you could call it after importing **spam** using the following syntax:

import spam

spam.bacon()

Alternatively, you could use the **from** keyword to import the **bac**

**on()** function directly:

from spam import bacon

bacon()