Answers

1. The two values of the Boolean data type are `True` and `False`. They represent logical true and logical false, respectively. These values are written exactly as shown: `True` and `False`.

2. The three different types of Boolean operators are:
   AND: It returns true if both operands are true.
   OR: It returns true if at least one operand is true.
   NOT: It returns the opposite boolean value of the operand.

3. AND Operator (&&):

A	B	A && B
false	false	false
false	true	false
true	false	false
true	true	true

OR Operator (||):

A	B	A || B
false	false	false
false	true	true
true	false	true
true	true	true

NOT Operator (!):

A	!A
false	true
true	false

4. (5 > 4) and (3 == 5):
    - True and False
    - Result: False

    not (5 > 4):

    - not True
    - Result: False

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

    - True or False
    - Result: True

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

    - not (True or False)
    - not True
    - Result: False

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

    - True and False
    - Result: False

    (not False) or (not True):

    - True or False
    - Result: True

5. The six comparison operators in Python are:

    Equal to (==)

Not equal to (!=)
Greater than (>)
Less than (<)
Greater than or equal to (>=)
Less than or equal to (<=)

6.
In Python, the single equal sign (=) is used as the assignment operator to assign a value to a variable. It does not compare values but assigns the value on the right to the variable on the left.

For comparison, the double equal sign (==) is used to check if two values are equal. It returns True if the values are equal and False otherwise.

7. Block 1

```
if spam == 10:

    print('eggs')
```

Block 2

```
if spam > 5:

    print('bacon')
```

Block 3

```
else:

    print('ham')

    print('spam')

    print('spam')
```

8.

```
spam =  # Assign spam here

if spam == 1:
```

```
    print('Hello')
elif spam == 2:
    print('Howdy')
else:
    print('Greetings!')
```

9. press Ctrl + C (Control and C keys pressed together) to terminate the program and break out of the loop.

10. In Python, "break" and "continue" are control flow statements used within loops:

> "break": When encountered within a loop (such as "for" or "while"), the "break" statement immediately terminates the loop and transfers control to the code following the loop.
>
> "continue": When encountered within a loop, the "continue" statement skips the rest of the code inside the loop for the current iteration and proceeds to the next iteration of the loop.

11. In Python, all three forms of `range()` produce the same result in a `for` loop, generating integers from 0 to 9 inclusively. Here's a brief explanation of each:

> `range(10)`: This form of `range()` starts from 0 by default and generates integers up to, but not including, 10. It implicitly starts from 0 and increments by 1.
> `range(0, 10)`: This explicitly specifies the start and stop values for the range. It starts from 0 and stops just before 10. By default, it also increments by 1.
> `range(0, 10, 1)`: Similar to the second form, this explicitly specifies the start, stop, and step values. It starts from 0, stops just before 10, and increments by 1. However, specifying the step size explicitly as 1 doesn't change its behavior; it's the default behavior of `range()`.

All three forms are equivalent and result in the same sequence of integers: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

12. For loop

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

While loop

```
i = 1
while i <= 10:
    print(i)
    i += 1
```

13. After importing the `spam` module, you can call the `bacon()` function using the following syntax:

```
spam.bacon()
```

This syntax specifies that you're calling the `bacon()` function which is defined within the `spam` module.