Assignment No-2

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

Solution: The two values of the Boolean data type are "true" and "false". In programming languages, these values are written as "true" and "false," respectively, without quotes.

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

Solution: The three different types of Boolean operators are:

1. AND operator: This operator is represented by the symbol "&&". It returns true if both operands are true. Otherwise, it returns false.

2. OR operator: This operator is represented by the symbol "||". It returns true if at least one of the operands is true. Otherwise, it returns false.

3. NOT operator: This operator is represented by the symbol "!". It reverses the logical state of its operand. If the operand is true, the NOT operator returns false; if the operand is false, the NOT operator returns 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).

Here are the truth tables for the three basic Boolean operators: AND, OR, and NOT.

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 (also known as the negation operator):

A NOT A

True False

False True

These truth tables show every possible combination of Boolean values for the given operator and the resulting evaluation of the expression.

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)

Solution: The values of the expressions are as follows:

(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

5. What are the six comparison operators?

Solution: The six comparison operators in many programming languages are:

a) Equality operator (==): Checks if two values are equal.
b) Inequality operator (!=): Checks if two values are not equal.
c) Greater than operator (>): Checks if the value on the left is more significant than on the right.
d) Less than operator (<): Check if the value on the left is less than on the right.
e) Greater than or equal to the operator (>=): Check if the value on the left is greater than or equal to the value on the right.
f) Less than or equal to the operator (<=): Check if the value on the left is less than or equal to the value on the right.

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

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

For example, let's consider a condition where we must check if a number equals 5.

To check the equality, we use the equal to the operator:

number = 5

if number == 5:

print("The number is equal to 5")

In this case, the condition `number == 5` returns `True` if the value of the `number` variable is equal to 5. If the condition is proper, the statement inside the if block will be executed.

On the other hand, the assignment operator (=) is used to assign a value to a variable:

number = 5

result = number * 2

In this example, we use the assignment operator to assign the result of multiplying `number` by 2 to the variable `result`. The value of `result` would be 10 after this assignment.

To summarize, the equal to the operator (==) is used for comparison, while the assignment operator (=) is used for assigning values to variables.

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')

Solution: The three blocks in this code are:

a) The first block is within the if statement `if spam == 10:` where the print statement `print('eggs')` is indented under it.

b) The second block is within the if statement `if spam > 5:` where the print statement `print('bacon')` is indented under it.
c) The third block is within the else statement where the print statements `print('ham')`, `print('spam')`, and `print('spam')` are all indented under it.

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.

Solution: Here is the code that prints the desired outputs based on the value stored in the variable Spam :

```
spam = 3

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 do you press?

Solution: If a program is stuck in an endless loop, you can press the designated key combination to terminate or force the program to stop. The exact key combination can vary depending on the operating system and the development environment you are using. Here are some commonly used key combinations:

a) Windows: Ctrl + C or Ctrl + Break
b) macOS: Command +. (period)
c) Linux: Ctrl + C

These key combinations send a signal to the program to stop execution and terminate the loop. Forcefully terminating a program may cause any unsaved progress to be lost, so it should be used as a last resort.

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

Solution: The "break" and "continue" statements are used in loops to control the flow of execution.

1. "Break":

- When the "break" statement is encountered within a loop (for loop, while loop, or do-while loop), it immediately terminates the loop and comes out of it.

a) It is typically used to stop the execution of a loop prematurely before its normal termination condition is met.
b) After the break statement is executed, the program continues to execute the following statement after the loop.
c) The break statement can only be used in nested loops to break out of the innermost loop.

Example:

```
for i in range(1, 6):
    if i == 3:
        break
    print(i, end=" ")
# Output: 1 2
```

2. "Continue":

a) When the "continue" statement is encountered within a loop, it skips the remaining code inside the loop for the current iteration and moves on to the next iteration.
b) is typically used to skip certain iterations based on a specific condition without terminating the entire loop.
c) After the continue statement is executed, the program goes back to the beginning of the loop to check the condition again.
d) The continue statement can also be used in nested loops to skip the remaining iterations of the innermost loop.

Example:

```
for i in range(1, 6):
    if i == 3:
        continue
    print(i, end=" ")
# Output: 1 2 4 5
```

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

Solution: In a for loop, the difference between range(10), range(0, 10), and range(0, 10, 1) is as follows:

a) range(10): This creates a sequence of numbers from 0 to 9 (inclusive). It starts from 0 by default and increments by 1 each time.

b) range(0, 10): This also creates a sequence of numbers from 0 to 9 (inclusive). However, the first parameter explicitly specifies the starting point, which is 0. The ending point is not provided, so it ends at 10-1=9. The increment is still 1 by default.
c) range(0, 10, 1): This is the same as range(0, 10). It explicitly specifies the starting point (0), the ending point (10-1=9), and the increment (1). However, since the increment is the default value, it can be omitted, resulting in a range(0, 10).

In practice, all three variations would produce the same result in a for loop and iterate 10 times, from 0 to 9.

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.

Solution: Here is the program that prints the numbers 1 to 10 using a for loop:

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

Here is the equivalent program that prints the numbers 1 to 10 using a while loop:

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

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

Solution: You can call the bacon() function from the spam module using the syntax `spam.bacon()`.