**ASSIGNEMENT-2**

1. THE BOOLEAN DATATYPE REPRESENTS LOGICAL VALUES AND HAS TWO POSSIBLE VALUES:

‘TRUE’ AND ‘FALSE’.

1. Logical OR

Logical NOT  
Logical AND

1. Logical AND (**and**):

| **Operand 1** | **Operand 2** | **Result** |
| --- | --- | --- |
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

1. Logical OR (**or**):

| **Operand 1** | **Operand 2** | **Result** |
| --- | --- | --- |
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

1. Logical NOT (**not**):

| **Operand** | **Result** |
| --- | --- |
| True | False |
| False | True |



4. True  
 False  
 True

False

False

False

5. The six main comparison operators are **equal to, not equal to, greater than, greater than or equal to, less than, and less than or equal to**. These operators are used in conditional expressions to determine if one block of code or another executes, thus controlling flow in a computer program. Different programming languages use different syntax to express these operators, but the meanings are the same.

### 6. Assignment Operator (=)

In mathematics and algebra, = is an **equal to** operator. In programming = is an [assignment operator](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Assignment), which means that it assigns a value to a variable.

### Equality Operator (==)

In JavaScript, the operator that compares two values is written like this: ==. It is called an [equality operator](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Equality). The equality operator is one of the many [comparison operators](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_Operators#comparison_operators) in JavaScript that are used in logical and conditional statements.

1. Spam=0

If spam==10  
 print(‘eggs’)

If spam>5:

Print(‘bacon’)

Else:

Print(‘ham’)

Print(‘spam’)

Print(‘spam)

1. spam = 1

if spam == 1:

print("Hello")

elif spam == 2:

print("Howdy")

else:

print("Greetings!")

1. Ctrl+C

## Break Statement:

A break statement is used to terminate the loop whenever a particular condition is satisfied. The statement is there just next after the loop receives control of the program. The break statement will end the innermost loop if it is contained within a nested loop that is the loop inside the other loop. It is used to end the loop that it is enclosed in, such as a do-while, while, switch, and for statement.

## Continue Statement:

The continue statement skips the remaining lines of code, for the current iteration of the loop. In this case, the loop does not end, it continues with the next iteration.

In a **for** loop, the functions **range(10)**, **range(0, 10)**, and **range(0, 10, 1)** essentially produce the same output and iterate over the same sequence of numbers. The difference lies in the way the arguments are specified.

1. **range(10)**: This form of **range** starts from 0 by default and generates a sequence of numbers up to (but not including) the specified value, which is 10 in this case. It increments by 1 by default. So, **range(10)** produces the sequence of numbers **[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]**.

**range(0, 10)**: This form of **range** explicitly specifies the starting and ending values of the sequence. It starts from 0 and generates numbers up to (but not including) the specified ending value, which is 10. The increment is 1 by default. Therefore, **range(0, 10)** also produces the sequence **[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]**.

**range(0, 10, 1)**: This form of **range** explicitly specifies the starting value, ending value, and the increment. It starts from 0, generates numbers up to (but not including) the ending value of 10, and increments by 1. The increment of 1 is explicitly mentioned in the arguments, but since it is the default value, it can be omitted without affecting the output. Therefore, **range(0, 10, 1)** also produces the same sequence **[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]**.

1. for num in range(1, 11):

print(num)

1. import spam

spam.bacon()