Assignment-2

1. **What are the two values of the Boolean data type? How do you write them?**
   Ans.
   In programming, we often need to know if an expression is True or False. When an expression is evaluated in Python, we only get two answers i.e. True or False.
   Ex. print(10 > 9) o/p: True
   print(10 == 9) o/p: False

2. **What are the three different types of Boolean operators?**
   Ans.
   - The logical operators 'and', 'or' and 'not' are referred to as Boolean operators.
   - While and as well as or operator needs two operands, which may evaluate to true or false, not operator needs one operand evaluating to true or false.

     Ex. on and

     a = 50
     b = 25

     1. a > 40 and b > 50
        o/p: False

     2. a > 0 and b > 0
        o/p: True

     Ex. on or

     a = 50
     b = 25

     1. a > 100 or b > 20
        o/p: True

     2. a == 0 or b == 0
        o/p: False

Ex. on not

1. a = 100
   a > 100
   o/p: False

   not(a > 100)
   o/p: 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 evaluate).**
   Ans.
   and Table:

a	b	a and b
False	False	False
False	True	False
True	False	False
True	True	True

or Table:

a	b	a or b
False	False	False
False	True	True
True	False	True
True	True	True

not Table:

a	not a
False	True
True	False

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)

Ans.

```
In [2]: (5 > 4) and (3 == 5)

Out[2]: False

In [3]: not(5 > 4)

Out[3]: False

In [4]: (5 > 4) or (3 == 5)

Out[4]: True

In [5]: not ((5 > 4) or (3 == 5))

Out[5]: False

In [6]: (True and True) and (True == False)

Out[6]: False

In [7]: (not False) or (not True)

Out[7]: True
```

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

Operator	Description	Syntax
>	Greater than: True if left operand is greater than right	a > b
<	Less than: True if left operand is less than right	a < b
==	Equal to: True if both operands are equal	a == b
!=	Not equal to: True if both operands are not equal	a != b
>=	Greater than or equal to: True if left operand is greater than or equal to right	a >= b
<=	True if left operand is less than or equal to right	a <= b

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

   a = 20

   Here, equal to (=) sign acts as an assignment operator. Which stores the value in the memory named as ‘a’. a is a variable.

x = 25

y = 25

So, x == y, in this x and y have equal values which forms an equality. Therefore, = is used for assigning a value to the variable, and == is used for representing equality between two values or operands.

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')
```

Ans.

The three blocks in the code are:

- if spam == 10:
- if spam > 5
- else

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.**
Ans.

```
In [5]: spam = int(input())
        if spam == 1:
            print('Hello')
        elif spam == 2:
            print('Howdy')
        else:
            print('Greetings!')

        2
        Howdy
```

9. **If your program is stuck in an endless loop, what keys you'll press?**
   Ans.
   If the program gets stuck while executing the following code:

```
i = 5
# Run this loop while i is less than 15
while i < 15:
    # Print a message
    print("Hello, World!")
```

   Now, as the value of the variable **i** is never updated (it's always 5). Therefore, the condition **i < 15** is always **True** and the loop never stops.
   So, To stop the infinite execution of print statement we can manually press **Ctrl + C**.

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

break	**continue**
It eliminates the execution of remaining iteration of loop	It will terminate only the current iteration of loop.
'break' will resume control of program to the end of loop enclosing that break.	The 'continue' will resume the control of the program to next iteration of that loop enclosing continue.
It early terminates the loop.	It causes the early execution of the next iteration.
The break stop the continuation of the loop.	The continue does not stop the continuation of loop and it stops the current.

11. **In a for loop, what is the difference between range(10), range(0, 10), and range(0, 10, 1)?**
    Ans.
    In for loop range() method is used for iteration on iterables like list, set, dictionary, etc. Inhere,
    range(10): Iteration will initiate from 0^{th} index location, which is by default whether starting index location is mentioned or not.
    range(0, 10): Iteration will intiate at 0^{th} index location to 9^{th} index location total forming a iteration count of ten.
    range(0, 10, 1): 0 is the start location, 10 is the stop location, and step i.e. 1 is the integer value which determines the increment between each integer in the sequence.

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.**
Ans.

```
In [15]: #Program to print integers from 1 to 10 using for loop

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

```
1
2
3
4
5
6
7
8
9
10
```

```
In [16]: #Program to print integers from 1 to 10 using while loop

i = 1

while i < 11:
    print(i)
    i += 1
```

```
1
2
3
4
5
6
7
8
9
10
```

13. **If you had a function named bacon() inside a module named spam, how would you call it after importing spam?**
Ans.
This function can be called with spam.bacon().