Assignment 2

Q1: What are the two values of the Boolean data type? How do you write them?

Answer: -

Python Boolean type is one of the built-in data types provided by Python, which represents one of the two values i.e., True or False.

INPUT: a = 10

b = 20

a == b

OUTPUT: False

It is written by using capital T and F, with the rest of the word in lowercase.

Q2: What are the three different types of Boolean operators?

Answer: -

Boolean Operations are simple arithmetic of True and False values. These values can be manipulated by the use of Boolean operators which include **AND, Or, and NOT.**

Q3: Make a list of each Boolean operator truth tables (i.e. every possible combination of Boolean values for the operator and what it evaluate).

Answer: -

Boolean OR Operator

A	B	A or B
True	True	True
True	False	True
False	True	True
False	False	False

Boolean And Operator

A	B	A and B
True	True	True
True	False	False
False	True	False
False	False	False

Boolean Not Operator

A	Not A
True	False
False	True

Q4. 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)

Answer: -

```
(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
```

Q5. What are the six comparison operators?

Answer: -

> ***Comparison operators*** — operators that compare values and return true or false. The operators include: >, <, >=, <=, ===, ! ==.

Description of each operator: -

- **Less than (<)** — returns true if the value on the left is less than the value on the right, otherwise it returns false.
- **Greater than (>) —** returns true if the value on the left is greater than the value on the right, otherwise it returns false.
- **Less than or equal to (<=) —** returns true if the value on the left is less than or equal to the value on the right, otherwise it returns false.
- **Greater than or equal to (>=) —** returns true if the value on the left is greater than or equal to the value on the right, otherwise it returns false.
- **Equal to (===) —** returns true if the value on the left is equal to the value on the right, otherwise it returns false.
- **Not equal to (! ==) —** returns true if the value on the left is not equal to the value on the right, otherwise it returns false.

Q6. How do you tell the difference between the equal to and assignment operators? Describe a condition and when you would use one.

Answer: -

= Operator: - Assignment Operator

The **“=”** is an assignment operator is used to assign the value on the right to the variable on the left.

For example:

a = 10;

b = 20;

Ch = 'y';

Constant term cannot be placed on left hand side.
Example: 1=x; is invalid.

== Operator: - Comparison Operator

The ‘==’ operator checks whether the two given operands are equal or not. If so, it returns true. Otherwise, it returns false.

For example:

5==5

This will return true.

Constant term can be placed in the left-hand side.
Example: 1==1 is valid and returns 1.

Q7. 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’)
```

Answer: -

```
spam = 0
if spam == 10:
    print('eggs')
if spam > 5:
    print ('bacon')
else:
    print ('ham')
    print('spam')
```

```
ham
spam
```

Q 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.

Answer: -

```
In [13]: spam = int(input())

if spam == 1:
    print('Hello')
elif spam == 2:
    print('Howdy')
else:
    print('Greetings!')
```

```
3
Greetings!
```

Q9: If your programme is stuck in an endless loop, what keys you'll press?

Answer: - Ctrl + C.

Q10. How can you tell the difference between break and continue?

Answer: -

Break Statement	Continue Statement
The Break statement is used to exit from the loop constructs.	The continue statement is not used to exit from the loop constructs.
The break statement is usually used with the switch statement, and it can also use it within the while loop, do-while loop, or the for-loop.	The continue statement is not used with the switch statement, but it can be used within the while loop, do-while loop, or for-loop.
When a break statement is encountered then the control is exited from the loop construct immediately.	When the continue statement is encountered then the control automatically passed from the beginning of the loop statement.
Syntax: break;	**Syntax:** continue;

Q11. In a for loop, what is the difference between range (10), range (0, 10), and range (0, 10, 1)?

Answer: -

There are three ways you can call range ():

- range (**stop**) takes one argument.
- range (**start, stop**) takes two arguments.
- range (**start, stop, step**) takes three arguments.

range (stop): -When user call range () with one argument, user will get a series of numbers that starts at 0 and includes every whole number up to, but not including, the number that user has provided as the stop.

Example: - **range (10)**

Output: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

range (start, stop): - When user call **range ()** with two arguments, user get to decide not only where the series of numbers stops but also where it starts, so user don't have to start at 0 all the time. User can use range () to generate a series of numbers from X to Y using a range (X, Y).

Example: **range (0, 10)**

Output: - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

range (start, stop, step): - When the user call range () with three arguments, the user can choose not only where the series of numbers will start and stop but also how big the difference will be between one number and the next. If the user doesn't provide a step, then range () will automatically behave as if the step is 1.

Example: - **range (0, 10, 1)**

Output: - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Q12. 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.

Answer: -

```
In [4]: for i in range (1,11):
            print(i)

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

```
In [25]: i = 1
         while(i<=10):
             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?

Answer: -

This function can be called with spam.bacon().

**********END**********