# Assignment 2

### **Question 1:-**

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

#### **Ans:-**

There are two values of the Boolean data type:-

1. True
2. False

Using capital T and F, with the rest of the word in lowercase.

Use the Boolean Data Type (Visual Basic) to contain two-state values such as **true/false, yes/no, or on/off.** The default value of Boolean is False.

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. Generally, it is used to represent the truth values of the expressions. For example, 1 == 1 is True whereas 2 < 1 is False.

Or you can simply assign a True or False value or even an expression that ultimately evaluates one of these values. You can check the type of the variable by using the built-in type function in Python.

### **Question 2:-**

**What are the three different types of Boolean operators?**

#### **Ans:-**

There are three different types of Boolean operators

1. AND:- **AND** will narrow your search results to include only relevant results that contain your required keywords.
2. OR:- **OR** will expand your search results so all results must contain at least one, if not more, of your defined keywords or phrases.
3. NOT:- **NOT** limits your search by excluding defined keywords and/or phrases from your results.

### **Question 3:-**

**Make a list of each Boolean operator’s truth table (i.e. every possible combination of Boolean values for the operator and what it evaluates).**

#### **Ans:-**

Truth tables summarize how we combine two logical conditions based on **AND, OR, and NOT.**

Computer programs are constantly making decisions based on the current “STATE” of the data held by the program. Combining multiple conditions to form one True/False value is the domain of Logic.

The primary way to combine two boolean expressions into one is through the use of AND or OR. In most programming languages, **AND** is written using double ampersands: **&&**. **OR** is written using double pipes: **||**. **Not** is written using: **~.**

| Condition 1 | Condition 2 | AND | OR | NOT ( 1 / 2 ) |
| --- | --- | --- | --- | --- |
| False | False | False | False | True / True |
| False | True | False | True | True / False |
| True | False | False | True | False / True |
| True | True | True | True | False / False |

OR

| Condition 1 | Operator | Condition 2 | Result |
| --- | --- | --- | --- |
| True | And | True | True |
| True | And | False | False |
| False | And | True | False |
| False | And | False | False |
| True | Or | True | True |
| True | Or | False | True |
| False | Or | True | True |
| False | Or | False | False |
| True | Not | … | False |
| False | Not | … | True |
| … | Not | Ture | False |
| … | Not | False | True |

### 

### 

### **Question 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 :-**

( 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**

### 

### **Question 5:-**

What are the six comparison operators?

#### **Ans:-**

Python comparison operators are also known as **relational operators**.

The comparison operators return **True** or **False** by evaluating the expression.

There are many types of Python comparison operators.

1. **Less than (<):-**

The first comparison operator we will see is the **less-than** operator. It’s denoted by **“<”** and it is used to check if the left value is less than the right value or not.

**For example :-**

8<10

**Output :-**

True

1. **Greater than (>):-**

Python **greater than** symbol. It’s denoted by the **“>”** symbol and it checks whether the value on the left side is greater than the right side.

**For example :-**

0.5 > 0

**Output :-**

True

1. **Less than or equal to (<=):-**

The less than or equal to operator, denoted by **“<=”** returns True when the **left side** operand is either **less than or equal to** the **right side** operand.

**For example :-**

5 <= 10

**Output :-**

True

1. **Greater than or equal to (>=):-**

The greater than or equal to the operator is just like **Less than or equal to**. The only difference is that it checks that the **left side** value should be **greater than or equal to** the **right side** value.

**For example :-**

14 >= 10

**Output :-**

True

1. **Equal to (==):-**

The equal to the operator will return True when both the values on either side of the operator are **equal.** You compare **integers, float,** and also **strings.**

**For example :-**

23 == 23

**Output :-**

True

1. **Not equal to (!=):-**

The **not equal to the** operator **(!=)** is **opposite** to the **equal to the** operator. It returns true when the values on either side are **unequal** to each other.

**For example:-**

26 != 24

**Output:-**

True

### **Question 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 condition is an expression used in a flow control statement that evaluates to a Boolean value.

**Equal to** and **Assignment operators are not the same, “=”** is an **Assignment operator** it is used to assign the value of a variable or expression, while

**“==”** is an **Equal to Operator** and is a relation operator used for comparison (to compare values of both left and right side operands).

### **Question 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 are everything inside the if statement and the lines

1. print(‘bacon’) and print(‘ham’).
2. print(‘eggs’) if spam > 5 : print(‘bacon’)
3. else : print(‘ham’) print(‘spam’).

### **Question 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:-**

if spam == 1:

print(“Hello”)

elif spam ==2:

print(“Howdy”)

else :

print(“Greetings!”)

**spam = 5**

**if spam == 1 :**

**print("Hello")**

**if spam ==2 :**

**print("Howdy")**

**else :**

**print("Greetings!")**

**Output:-**

**Greetings!**

### **Question 9:-**

If your program is stuck in an endless loop, what keys will you press?

#### **Ans:-**

Press **Ctrl-C** to stop a program stuck in an infinite loop.

### **Question 10:-**

How can you tell the difference between break and continue?

#### **Ans:-**

| Basis for comparison | break | continue |
| --- | --- | --- |
| Task | It eliminates the execution of the remaining iteration of the loop. | It will terminate only the current iteration of the loop. |
| Control after break/continue | “break” will resume control of the program to the end of the loop enclosing that “break”. | The “continue” will resume the control of the program to the next iteration of that loop enclosing “continue”. |
| causes | It terminates the loop. | It causes the early execution of the next iteration. |
| continuation | The “break” stop the continuation of the loop. | The “continue” does not stop the continuation of the loop and it stops the current. |
| other | It used with the “switch”, and “label”. | Cannot be executed with the switch and the labels. |

### 

### 

### **Question 11:-**

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

#### **Ans:-**

They all do the same thing. The range(10) call ranges from 0 up to (but not including) 10, range(0,10) explicitly tells the loop to start at 0, and range(0,10,1) explicitly tells the loop to increase the variable by 1 on each iteration.

### 

### 

### 

### **Question 12:-**

Write a short program that prints the numbers 1 to 10 using for a loop. Then write an equivalent program that prints the numbers 1 to 10 using a while loop.

#### **Ans:-**

**Program to prints the numbers 1 to 10 using a for loop:-**

for i in range(1,11) :

print(i)

**Output:-**

1

2

3

4

5

6

7

8

9

10

**Program to prints the numbers 1 to 10 using a for loop :-**

**i = 1**

**while(i <= 10) :**

**print(i)**

**i += 1**

**Output:-**

1

2

3

4

5

6

7

8

9

10

### 

### 

### **Question 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()** .