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

**Answer:** Booleans represent one of two values: **True or False / 1 or 0**

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

**Answer:** There are three logical operators that are used to compare values. They evaluate expressions to Boolean values, returning either True or False.

These operators are **and, or and not**

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).

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)

5. What are the six comparison operators?

**Answer:** Python has six comparison operators: less than (<), less than or equal to (<=), greater than (>), greater than or equal to (>=), equal to (==), and not equal to (!=).

6. How do you tell the difference between the equal to and assignment operators? Describe a

condition and when you would use one.

**Answer:** equal to (=) is the assignment operator used for assigning a value to a variable. (==) is the equality operator used to check if a variable is equal to another variable.

E.g. int a = 10; assigns 10 to variable a.

E.g. if (a == 10) checks if variable a is equal to 10 or not.

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’)

**Answer:**

spam = 0

if spam == 10:

print('eggs')

if spam > 5:

print('bacon')

else:

print('ham')

print('spam')

print('spam')

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:**

spam = int(input('Enter any value : '))

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 you’ll press?

**Answer**: You can stop an infinite loop with CTRL + F2

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

**Answer:** when **break** keyword encountered, it terminates the execution of the current loop and passes the control to the next loop or main body, whereas when **continue** keyword is encountered, it skips the current iteration and executes the very next iteration in the loop

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

**Answer:** There are three ways we can call range() :

1. **range(stop) takes one argument.**

When user call range(10) with one argument, user will get a series of numbers that starts at 0 and includes every whole number up to, but not including 10.

**e.g.** for i in range(10):

print(i, end=" ")

print()

**output**: 0 1 2 3 4 5 6 7 8 9

1. **range(start, stop) takes two arguments.**

When user call range(0,10) 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)

**e.g.** for i in range(0,10):

print(i, end=" ")

print()

**output**: 0 1 2 3 4 5 6 7 8 9

1. **range(start, stop, step) takes three arguments.**

When the user call range(0,10,1) 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.

**e.g.** for i in range(0,10,1):

print(i, end=" ")

print()

**output**: 0 1 2 3 4 5 6 7 8 9

**Another example (with step size 3)**

**e.g.**for i in range(0,10,3):

print(i, end=" ")

print()

**output**: 0 3 6 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.

1. **For loop**

for i in range(0,11):

print(i, end=" ")

print()

**output:** 0 1 2 3 4 5 6 7 8 9 10

1. **While loop**

i = 1

while i < 11:

print(i, end=" ")

i += 1

**output:** 0 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:** we need to use the **import** keyword along with the desired module name. When interpreter imports the module then we can use the functions inside a module by using a dot(.) operator along with the module name.

e.g

import spam

spam.bacon()