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

Ans : Two values of Boolean data type are

True

False

These are written with first letter capitalised and without any quotes and are built in keywords

These values are used to represent truth or falsity of the condition or expression

Example : >> a = True

>>b = Flase

>>print (a) >>>> True

>>print(b) >>>>>False

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

Ans : the three different types of Boolean operators

1. and
2. or
3. not

example for and example for or example for not

a = 3 a = 3 a =3

b = 5 b = 5 b = 5

a<b and b>a a>b or b<a not(a<b)

>>>True >>>False >>>False

a>b and b<a a<b or b>a not(a>b)

>>>False >>>True >>>True

**3**. Make a list of each Boolean operator&#39;s truth tables (i.e. every possible combination of Boolean values for the operator and what it evaluate ).

Ans: if we have the two operands **True** and **False**, the **and** operator would evaluate to **False** because only one of the operands is **True**. However, if we have both operands as **True**, then the **and** operator would evaluate to **True** because both operands are **True**

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

if we have the two operands **True** and **False**, the **or** operator would evaluate to **True** because at least one of the operands is **True**. However, if both operands are **False**, then the **or** operator would evaluate to **False** because neither operand is **True**

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

In each of the truth tables above, the operands represent Boolean values that can either be **True** or **False**. The operator combines the two operands and returns a Boolean result, which is also either **True** or **False**. The truth tables show every possible combination of Boolean values for the operator and what it evaluates to.

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

**4**. What are the values of the following expressions?

(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

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

Ans : 1)Greater than : >

2)Smaller than:<

3)equal to:==

4)Greater than or equal to:(>=)

5)Smaller than or equal to:(<=)

6)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.

Ans: The equal to operator (==) and assignment operator (=) are two different approach in programming

Equal to operator is used to compare the two values to check if they are equal or not

It return the Boolean value or True or False

Example : 5==5 would return True

5==6 would return False

Assignment operators assign value to the variable , it takes value on right hand side and and stores it in the left hand side

Example : x = 5

example : x = 5

y = 6

if x==y :

print(“both x and y value are equal”)

Else :

print(“they are not equal”)

**7**. Identify the three blocks in this code:

spam = 0

if spam == 10:

print(‘eggs’)

if spam > 5:

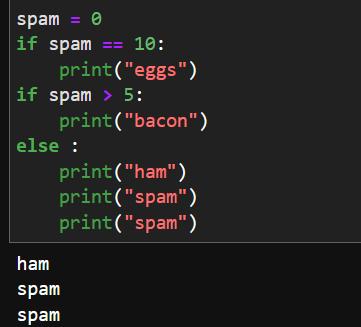
print(‘bacon’)

else:

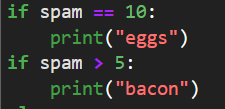
print(‘ham’)

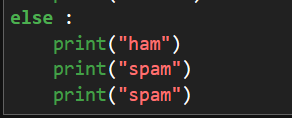
print(‘spam’)

print(‘spam’)



 This is first block

 2nd block

 3rd block

So three block of code are:

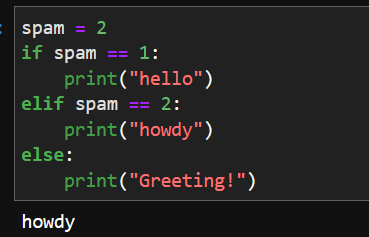
1.initalization of spam variable

2.First if statement and if-else statement block

3.else statement block

**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.**

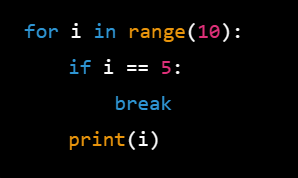
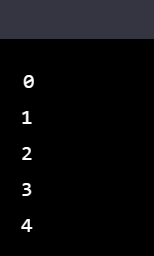
Ans : 

**9.If your programme is stuck in an endless loop, what keys you’ll press?**

Ans:Click on the stop button or Terminate button or menu option to stop excuting program

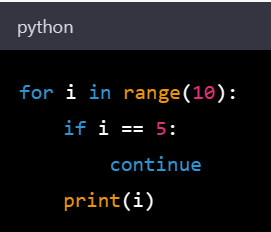
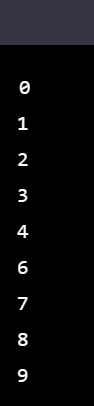
**10. How can you tell the difference between break and continue?**

Ans: **break** statement: When executed inside a loop, it immediately terminates the loop and transfers the control to the statement immediately following the loop. It is used to exit a loop prematurely when a certain condition is met

In the above example loop has to run i=0 to i = 9 but break execute and terminate the loop prematurely after i = 4

**continue** statement: When executed inside a loop, it immediately skips the rest of the statements in the current iteration of the loop and goes to the next iteration. It is used to skip certain iterations of a loop when a certain condition is met

In above example when loop is i = 5 its skips and continue end the loop

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

Ans: range(10):

It prints the values from 0 to 9

range(0,10)

it prints the value from 0 to 9 but here we can modify the starting range and ending range

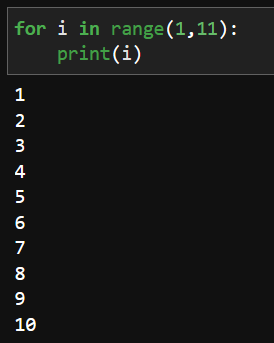
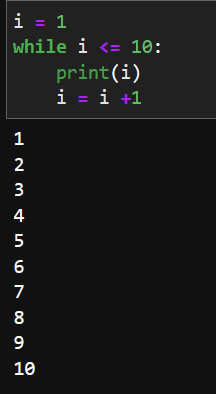
for example range(5,10) prints >>>5,6,7,8,9

range (0,10,1)

it prints the value from 0 to 9 with the step size of 1

example : range(0,10,2) >>>> 0,2,4,6,8

**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:  

**13. If you had a function named bacon() inside a module named spam, how would you call it after**

**importing spam?**

Ans: If you have a function named **bacon()** inside a module named **spam**, you can call it after importing the **spam** module using the dot notation:

Import spam

Spam.bacon()