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

The two values of the Boolean data type are: True and False These values are written as True and False, respectively. T and F should be written in uppercase as puthon is case sensitive language.

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

The three different types of Boolean operators are:

AND operator: denoted by and. It returns True if both operands are true, otherwise, it returns False. OR operator: denoted by or. It returns True if at least one of the operands is true, otherwise, it returns False. NOT operator: denoted by not. It returns the opposite of the operand's logical value. If the operand is True, not will return False, and if the operand is False, not will return 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 ).

the truth tables for each Boolean operator:

AND operator (and):
value 1      value 2     Result
False        False       False
False        True        False
True         False       False
True         True        True

OR operator (or):
Value 1      Value 2     Result
False        False       False
False        True        True
True         False       True
True         True        True

NOT operator (not):
Value        Result
False        True
True         False
These truth tables represent the evaluation of each Boolean operator for every
possible combination of Boolean values.
```

In [3]:

```
# Boolean AND operator
Value1 = False
Value2 = False
result = Value1 and Value2
print(result)  # Output: False

Value1 = False
Value2 = True
result = Value1 and Value2
print(result)  # Output: False

Value1 = True
Value2 = False
result = Value1 and Value2
print(result)  # Output: False

Value1 = True
Value2 = True
result = Value1 and Value2
print(result)  # Output: True

# Boolean OR operator
Value1 = False
Value2 = False
result = Value1 or Value2
print(result)  # Output: False

Value1 = False
Value2 = True
result = Value1 or Value2
print(result)  # Output: True

Value1 = True
Value2 = False
result = Value1 or Value2
print(result)  # Output: True

Value1 = True
Value2 = True
result = Value1 or Value2
print(result)  # Output: True

# Boolean NOT operator
Value = False
result = not Value
print(result)  # Output: True

Value = True
result = not Value
print(result)  # Output: False
```

```
False
False
False
True
False
True
True
True
True
False
```

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

(5 > 4) and (3 == 5)__________________# Output - False
not (5 > 4)___________________________***# Output - False***
(5> 4) or (3 == 5)_____________________*# Output - True*
*not ((5 > 4) or (3 == 5))*_____________# Output - False
(True and True) and (True == False)___***# Output - False***
(not False) or (not True)_____________**# Output - True**

In [1]:

```
a = (5 > 4) and (3 == 5)
print(a)
```

```
False
```

In [2]:

```
b = not (5 > 4)
print(b)
```

```
False
```

In [4]:

```
c = (5> 4) or (3 == 5)
print(c)
```

```
True
```

In [5]:

```
d = not ((5 > 4) or (3 == 5))
print(d)
```

```
False
```

In [6]:

```
e = (True and True) and (True == False)
print(e)
```

```
False
```

In [7]:

```
f = (not False) or (not True)
print(f)
```

```
True
```

# 5. What are the six comparison operators?

```
The six comparison operators in Python are:
Equal to (==)
Not equal to (!=)
Greater than (>)
Less than (<)
Greater than or equal to (>=)
Less than or equal to (<=)
These operators are used to compare values and return a Boolean result (True or
False) based on the comparison.
```

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

The equal to operator (==) is used to compare two values for equality, while the assignment operator (=) is used to assign a value to a variable.

In [8]:

```
p = 12
if p == 12:
    print("p is equal to 12")
```

```
p is equal to 12
```

In this case, the equal to operator (==) is used to check if the value of p is equal to 12. If it is, the condition is evaluated as True, and the statement inside the if block is executed.

In [9]:

```
q = 12
print(q)
```

```
12
```

Here, the value 12 is assigned to the variable q. The assignment operator assigns the value on the right-hand side to the variable on the left-hand side. The equal to operator (==) is used for comparison, while the assignment operator (=) is used for assigning values to variables.

7. Identify the three blocks in this code:

In [10]:

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

```
ham
spam
spam
```

Block1: This block is an if statement that checks if the value of spam is equal to 10. If it is, the code inside the block (in this case, the print('eggs') statement) will be executed.

In [11]:

```
if spam == 10:
 print('eggs')
```

Block2: This block is another if statement that checks if the value of spam is greater than 5. If it is, the code inside the block (the print('bacon') statement) will be executed.

In [ ]:

```
if spam > 5:
 print('bacon')
```

Block3: This block is an else statement that is associated with the second if statement. If the condition in the second if statement is not met (i.e., if spam is not greater than 5), the code inside this block (the print('ham') statement) will be executed. The subsequent print statements ('spam' and 'spam') are also part of this block and will be executed regardless of the condition.

In [ ]:

```
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.

In [12]:

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

    ('Greetings!')
```

```
Hello
```

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

If my programme is stuck in an endless loop, then I will press 'Ctrl + C' keys on keyboard to send an interrupt signal to the program. This will terminate the execution of the program and break out of the endless loop.

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

Break and continue are used within loops to alter the flow of execution.

Break is used to terminate the current loop and exit its execution immediately. When encountered, the loop is exited, and the program continues with the next statement after the loop.

continue is used to skip the remaining code in the current iteration of the loop and move to the next iteration. When encountered, the current iteration is terminated, and the loop jumps to the next iteration without executing the remaining code within the loop for that iteration.

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

In a for loop, range(10), range(0, 10), and range(0, 10, 1) are different ways of specifying the range of values to iterate over. However, they all produce the same result.

range(10) generates a sequence of numbers starting from 0 (default start value) up to, but not including, 10 (specified end value). By default, it increments by 1 for each iteration.

range(0, 10) explicitly sets the start value as 0 and the end value as 10. It generates a sequence of numbers starting from 0 up to, but not including, 10. It also increments by 1 for each iteration, which is the default behavior.

range(0, 10, 1) specifies the start value as 0, the end value as 10, and the step value as 1. It generates a sequence of numbers starting from 0 up to, but not including, 10, incrementing by 1 for each iteration.

In summary, all three forms of range() produce the same sequence of numbers from 0 to 9, incrementing by 1. The difference lies in the way the start, end, and step values are explicitly specified or implicitly assumed.

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.

In [2]:

```python
for count in range(11):
    print(count)
```

```
0
1
2
3
4
5
6
7
8
9
10
```

In [4]:

```
count = 0
while count < 11:
    print(count)
    count += 1
```

```
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?

spam is the module name, and bacon() is the function name. By prefixing the function name with the module name and a dot (spam.bacon()), you can access and call the bacon() function from the imported module.

In [ ]:

```
import spam

spam.bacon()
```

In [ ]: