In [ ]:
```
#1.What are the two values of the Boolean data type? How do you write them?
ANS:Python boolean type is one of the built-in data types provided by Python, which re
    i.e. True or False. Generally, it is used to represent the truth values of the exp
    The boolean value can be of two types only i.e. either True or False. The output
    is a boolean data type.
    We can evaluate values and variables using the Python bool() function. This method
    to a Boolean value i.e., True or False, using the standard truth testing procedure
```

In [ ]:
```
#2. What are the three different types of Boolean operators?
ANS:There are three basic Boolean search commands: AND, OR and NOT.

AND searches find all of the search terms.  For example, searching on dengue AND malar
that contain all three search terms.  Very limited results.

OR searches find one term or the other.  Searching on dengue OR malaria OR zika returr
three search terms.  Returns a large number of results.

NOT eliminates items that contain the specified term.  Searching on malaria NOT zika r
but will specifically NOT return items that contain the word zika.  This is a way to f
```

In [ ]:
```
#3.Make a list of each Boolean operators truth tables?
ANS:TRUTH TABLE OF NOT OPERATOR:
     A           not A
    True         False
    False        True

    TRUTH TABLE OF AND OPERATOR:

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

    TRUTH TABLE OF OR OPERATOR:

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

In [9]:
```
#4. What are the values of the following expressions?
(5 &  4) and (3 == 5)
```

Out[9]:
```
False
```

In [3]:
```
not (5 & 4)
```

Out[3]:
```
False
```

In [5]:
```
(5 & 4) or (3 == 5)
```

Out[5]:
```
4
```

In [6]:
```
not ((5 & 4) or (3 == 5))
```

Out[6]:
```
False
```

In [7]:
```
(True and True) and (True == False)
```

Out[7]:
```
False
```

In [8]:
```
(not False) or (not True)
```

Out[8]:
```
True
```

In [ ]:
```
#5. What are the six comparison operators?
ANS: COMPARISION OPERATORS:

Operator                Name                          Example
==                     Equal                         x == y
!=                     Not equal                     x != y
>                      Greater than                  x > y
<                      Less than                     x < y
>=                     Greater than or equal to      x >= y
<=                     Less than or equal to         x <= y
```

In [ ]:
```
#6. How do you tell the difference between the equal to and assignment operators?Descr
ANS:The “=” is an assignment operator is used to assign the value on the right to the
    The '==' operator checks whether the two given operands are equal or not.
```

In [13]:
```
x = 5       #this is the assignment operator  ,here we are assigning 5 to x

print(x)
```

```
5
```

In [14]:
```
x = 5          #this is the equal to operator, here we are checking whether x and y ar
y = 3

print(x == y)
```

```
False
```

In [22]:
```
#7. Identify the three blocks in this code
#ANS:
spam = 0
if spam == 10:
    print(eggs)    #block1
if spam == 5:
    print(bacon)   #block2
else:
    print(spam)     #block3
    print(spam)
```

```
0
0
```

In [28]:
```
#8. Write code that prints Hello if 1 is stored in spam, prints Howdy if 2 is stored i
#if anything else is stored in spam.
#ANS:
spam=0
```

```
spam=int(input())
if spam==1:
    print("hello")
if spam==2:
    print("Howdy")
if spam>2 and spam<1:
    print("Greetings!")
```

```
2
Howdy
```

In [ ]:
```
#9.If your programme is stuck in an endless loop, what keys you'll press?
ANS:An infinite loop is a loop that runs indefinitely and it only stops with external
    is found. You can stop an infinite loop with CTRL + C .
```

In [ ]:
```
#10. How can you tell the difference between break and continue?
ANS:BREAK:
        *leaves the loop
        *skips remaining execution of complete loop
        *useful, if condition always evaluates to be True
    CONTINUE:
        *jumps for next iteration
        *skips execution of remaining statement(s) inside the loop for current iterati
        *useful, if wants to skip execution of some statement(s) inside the loop for a
    simply,The break statement is primarily used as an exit statement, allowing you to
    Conversely, the continue statement aids in moving from the current loop iteration
```

In [ ]:
```
#11. In a for loop, what is the difference between range(10), range(0, 10), and range(
ANS:
    RANGE(10):
        Python range() function generates the immutable sequence of numbers starting f
        integer. The range() is a built-in function that returns a range object that c
        which we can iterate using a for loop.
    RANGE(0,10):
         The range() function generates a sequence of integer numbers as per the argum
          to use the range() function in Python.
        For example, range(0, 10). Here, start=0 and stop = 10. It will generate integ
        -1. i.e., [0, 1, 2, 3, 4, 5,6,7,8,9]
    RANGE(0,10,1):
        This indicates range(start, stop[, step])
        start: (Lower limit) It is the starting position of the sequence. The default
            For example, range(0, 10). Here, start=0 and stop = 10
        stop: (Upper limit) generate numbers up to this number, i.e., An integer numbe
            stop (upper limit). The range() never includes the stop number in its resu
        step: Specify the increment value. Each next number in the sequence is generat
            preceding number. The default value is 1 if not specified. It is nothing b
            the result.
```

In [1]:
```
#12. Write a short program that prints the numbers 1 to 10 using a for loop. Then writ
# numbers 1 to 10 using a while loop.


for i in range(1, 11):    #for loop
    print(i)
```

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

In [2]:
```
i = 1
while(i<=10):                    #while loop
    print(i)
    i += 1
```

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

In [ ]:
```
#13. If you had a function named bacon() inside a module named spam, how would you cal
ANS:This function can be called with spam. bacon()
```