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

Boolean datatype will return either True or False

In [4]:
```
a=10
b=20
print(bool(a>b))
```

```
False
```

In [5]:
```
bool(a)
```

Out[5]:
```
True
```

In [6]:
```
bool(b)
```

Out[6]:
```
True
```

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

AND, OR & NOT are three Boolean operators availabe in python

In [7]:
```
a=10
b=20
#comparing a and b with a boolean operator 'and'. since a==b give 0 and a!=b give 1, according to the AND truth t
#result is zero i.e, False
(a==b) and (a!=b)
```

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

In [8]:
```
c=35
d=40
```

```
#similarly a!=b is true and c!=d is true so the result is true
(a!=b) and (c!=d)
```

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

1. 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).

In [9]:
```
a==b and b==a
```

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

In [10]:
```
a==b and a!=b
```

Out[10]:
```
False
```

In [11]:
```
a!=b and a==b
```

Out[11]:
```
False
```

In [12]:
```
a!=b and b!=a
```

Out[12]:
```
True
```

AND Truth Table False False - False False Truth - False Truth False - False Truth Truth - Truth

In [13]:
```
e=35
#below is the example of OR opeator
(a==b) or (c==e)
```

Out[13]:
```
True
```

OR Truth Table False False - False False Truth - Truth Truth False - Truth Truth Truth - Truth

In [14]:
```
#here not negates the value for ex: a=10, b=20 when a<b passed through not it gives result as False
not(a<b)
```

Out[14]:
```
False
```

In [15]:
```
#actual result is true
a<b
```

Out[15]:
```
True
```

NOT truth table False - True True- False4. 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)

In [16]:
```
(5 > 4) and (3 == 5)
# True and False is False
```

Out[16]:
```
False
```

In [17]:
```
not (5 > 4)

#Negation of True is False
```

Out[17]:
```
False
```

In [18]:
```
(5 > 4) or (3 == 5)
#True or False is True
```

Out[18]:
```
True
```

In [19]:
```
not ((5 > 4) or (3 == 5))
#negation of True is False
```

Out[19]:
```
False
```

In [20]:
```
(True and True) and (True == False)
#True and False is False
```

Out[20]:
```
False
```

In [21]:
```
(not False) or (not True)
#True or False is True
```

Out[21]:
```
True
```

5. What are the six comparison operators?Six comparison operators are - <, >, <=, >=, ==, != 1. lesser than '<' 2. greater than '>' 3. lesser than equal to '<=' 4. greater than equal to '>=' 5. equal to '==' 6. not equal to '!='

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

Assignment operator is denoted by '=' where as equal to operator is denoted by '=='

In [22]:
```
#for example
a=10    # value 10 is assigned to a

print(a) # if you try to print a, the result will be 10
```

```
10
```

In [23]:
```
# where as '==' is logical operator which compares if a==10 or not then it will perform some action

a=10
if a==10:
    print(a)
```

```
10
```

In [24]:
```
#It wont give any result since the if caluse will terminate if the conditional check is false
a=20
if a==10:
    print(a)
```

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')

In [25]:
```
# First block of code which prints 'eggs' if the spam value is equals to 10,
spam = 0
if spam == 10:
    print('eggs')

    # No result since if condition is false
```

In [26]:
```
# Second block of code which returns 'bacon' if spam values is greater than 5 else returns 'ham'
if spam > 5:
    print('bacon')
else:
    print('ham')
```

```
ham
```

In [27]:
```
# third block of code is to print 'spam'

print('spam')
print('spam')
```

```
spam
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 [29]:
```
a=int(input('Enter a value'))
if a==1:
    print('spam')
elif a==2:
    print('howdy')
else:
    print('Greetings!')
```

```
Enter a value10
Greetings!
```

9.If your programme is stuck in an endless loop, what keys you'll press?CTRL+C is used to terminate the loop10. How can you tell the difference between break and continue?break- stops execution and prints the result when if/while condition is true for example:

In [31]:
```
while True:
    a=int(input('Enter a value'))
    if a%2 !=0:
        print('odd number')
        break
    print('even number')

    #one can see the out put where the loop will continue till it gets input value an 'odd number'
```

```
Enter a value10
even number
Enter a value20
even number
Enter a value30
even number
Enter a value40
even number
Enter a value11
odd number
```

In [32]:
```
#another example of using break

for i in range(5):
    if i==3:
        break
    print(i)
    # we can observe as soon as index value becomes 3 i.e, i==3 the loop exists and prints the values before brea
```

```
0
1
2
```

Continue statement is used in for loop to skip current iteration for example

In [33]:
```
for i in range(5):
    if i==3:
        continue
    print(i)
    # continue statement skips value 3 here
```

```
0
1
2
4
```

11. In a for loop, what is the difference between range(10), range(0, 10), and range(0, 10, 1)?Normally range function takes 3 variables namely stop, (start,stop),(start,stop,step) Let`s see how it works with the above give range of values

In [34]:
```
for i in range(10):
    print(i)
    # range(10) will accept the values from 0-9 excluding the given range value of 10 as below
```

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

In [35]:
```
for i in range(0,10):   # this range function almost acts like above except providing starting index number as ze
    print(i)
```

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

In [36]:
```
for i in range(0,10,1): # this also acts as prvious functions since the step value is a standard differential val
    print(i)
```

```
0
1
2
3
4
5
6
7
8
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.

In [37]:
```
for i in range(11):
    print(i)
```

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

In [38]:
```
i=0
while (i<=10):
    print(i)
    i+=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?import spam spam.bacon()

In [ ]: