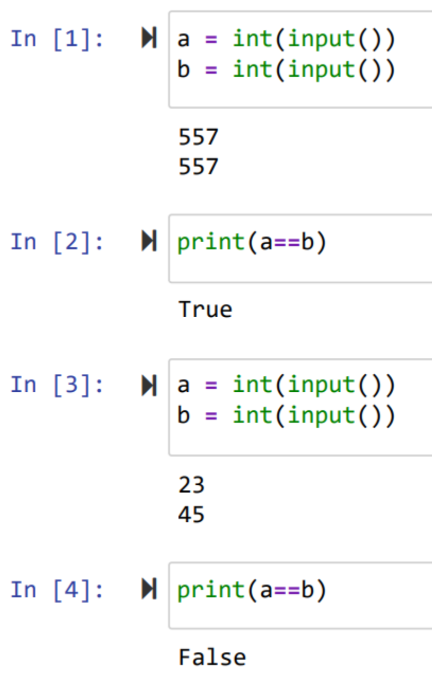
1.What are the two values of the Boolean data type? How do you write them?

Ans. The two types of values in boolean data type are:

a)True

b)False



```
In [1]:  a = int(input())
         b = int(input())

         557
         557

In [2]:  print(a==b)

         True

In [3]:  a = int(input())
         b = int(input())

         23
         45

In [4]:  print(a==b)

         False
```

In Python, these values are written as True and False respectively, with the first letter capitalized. It's important to note that these are case-sensitive, so true or false (with lowercase letters) will be treated as variable names, not boolean values.

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

Ans.

AND Operator (and):

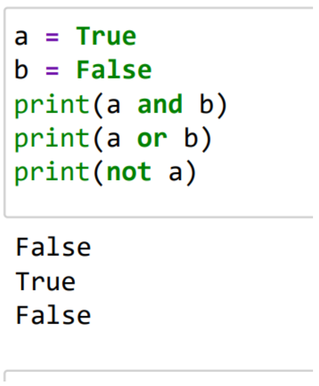
The and operator returns True if both operands are True, otherwise it returns False.

OR Operator (or):

The or operator returns True if either of the operands is True, otherwise it returns False.

NOT Operator (not):

The not operator is a unary operator that returns the opposite of the operand's boolean value.

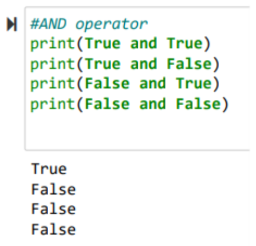


```python
a = True
b = False
print(a and b)
print(a or b)
print(not a)
```

```
False
True
False
```

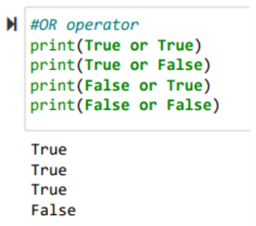
3. Make a list of each Boolean operator's truth tables (i.e. every possible combination of Boolean values for the operator and what it evaluate).

<u>Ans.</u>



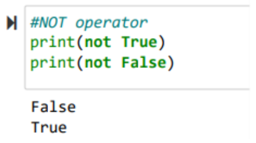
```python
#AND operator
print(True and True)
print(True and False)
print(False and True)
print(False and False)
```

```
True
False
False
False
```



```python
#OR operator
print(True or True)
print(True or False)
print(False or True)
print(False or False)
```

```
True
True
True
False
```



```python
#NOT operator
print(not True)
print(not False)
```

```
False
True
```

4. What are the values of the following expressions?

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

<u>output:</u>

True

b) not (5 > 4)

<u>output:</u>

False

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

output:

True

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

output:

False

e) (True and True) and (True == False)

output:

False

f) (not False) or (not True)

output:

True

5. What are the six comparison operators?

Ans.

a)Equal to (==)

b)Not equal to (!=)

c)Greater than (>)

d)Less than (<)

e)Greater than or equal to (>=)

f)Less than or 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.

Equal to (==) is used as a comparison operator, used to compare if two values are equal or not. But in case of a **assignment operator(=)**, it is used to assign a value to a variable.

Example1:

```
a = 500

b = 500

print(a==b)
```

<u>Output:</u>

```
True
```

<u>Example2:</u>

```
a = 500

print(a)
```

<u>Output:</u>

```
500
```

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')
```

<u>Ans.</u>

<u>BLOCK1:</u>

```
if spam == 10:

print('eggs')
```

<u>BLOCK2:</u>

```
if spam > 5:

print('bacon')
```

BLOCK3:

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.

Ans.

CODE:

```
spam = int(input())

if spam == 1:

    print("Hello")

elif spam == 2:

    print("Howdy")

else:

    print("Greetings!")
```

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

Ans.

You can stop an infinite loop with CTRL + C .

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

Ans.

break statement:

When encountered in a loop (such as for or while), the break statement immediately terminates the loop and transfers control to the next statement after the loop.

Example:

Code

```
for i in range(1, 6):

    if i == 4:

        break

    print(i)
```

output

1

2

3

continue statement:

When encountered in a loop, the continue statement skips the remaining code in the loop for the current iteration and moves to the next iteration.

Example:

Code

```
for i in range(1, 6):

    if i == 3:

        continue

    print(i)
```

output

1

2

4

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

Ans.

All 3 range(10), range(0, 10) and range(0,10,1) will be giving the same output.

But here,

In **range(10),** you specified only the stop and the start value is assumed to be 0, and the step size is assumed to be 1.

In **range(0, 10),** you explicitly specify both the start and stop values of the range, while the step size is assumed to be 1.

In **range(0, 10, 1),** you explicitly specify both the start and stop values of the range, and alsothe step size is specified to be 1.

CODE:

```
for i in range(10):

  print(i)
```

Outptut:

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

CODE:

```
for i in range(0, 10):

  print(i)
```

Outptut:

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

CODE:

```
for i in range(0, 10, 1):

  print(i)
```

Outptut:

```
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.

Ans.

program that prints the numbers 1 to 10 using a for loop:

```
for i in range(1, 11):

  print(i)
```

program that prints the numbers 1 to 10 using a while loop:

```
i = 1

while(i<= 10):

  print(i)

  i += 1
```

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

Ans.

Code:

```
import spam

spam.bacon()
```