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

Python boolean type is one of the built-in data types provided by Python, which represents one of the two values i.e. True or False. Generally, it is used to represent the truth values of the expressions. For example, 1==1 is True whereas 2<1 is False.

```
a = True
type(a)

b = False
type(b)
```

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

The logical operators and, or and not are also referred to as boolean operators. While and as well as or operator needs two operands, which may evaluate to true or false, not operator needs one operand evaluating to true or false.

Boolean 'AND' operator

Boolean and operator returns true if both operands return true.

```
a=50
b=25

a>40 and b>40
```

```
a>100 and b<50
```

```
a==0 and b==0
```

```
a>0 and b>0
```

Boolean OR operator

Boolean or operator returns true if any one operand is true

```
a=50
b=25
a>40 or b>40
```

```
a>100 or b<50
```

```
a==0 or b==0
```

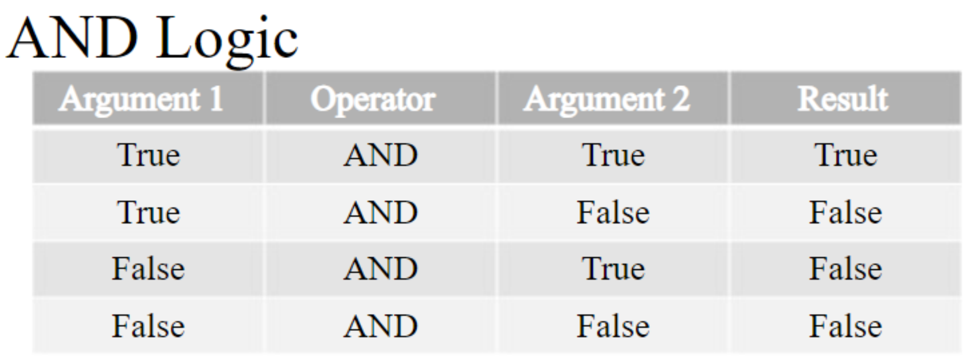
```
a>0 or b>0
```

NOT operator

The not operator returns true if its operand is a false expression and returns false if it is true.

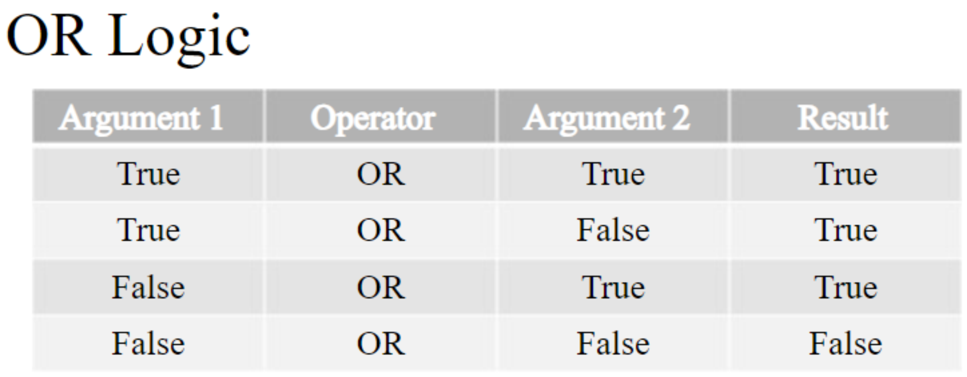
```
a=10
a>10
not(a>10)
```

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).



AND Logic

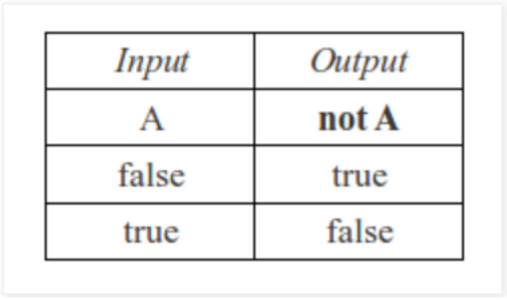
Argument 1	Operator	Argument 2	Result
True	AND	True	True
True	AND	False	False
False	AND	True	False
False	AND	False	False



OR Logic

Argument 1	Operator	Argument 2	Result
True	OR	True	True
True	OR	False	True
False	OR	True	True
False	OR	False	False

NOT LOGIC



Input	*Output*
A	**not A**
false	true
true	false

4. What are the values of the following expressions?

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

ii) not (5 > 4)

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

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

v) (True and True) and (True == False)

vi) (not False) or (not True)

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

```
False
```

```
not (5 > 4)
```

```
False
```

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

```
True
```

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

```
False
```

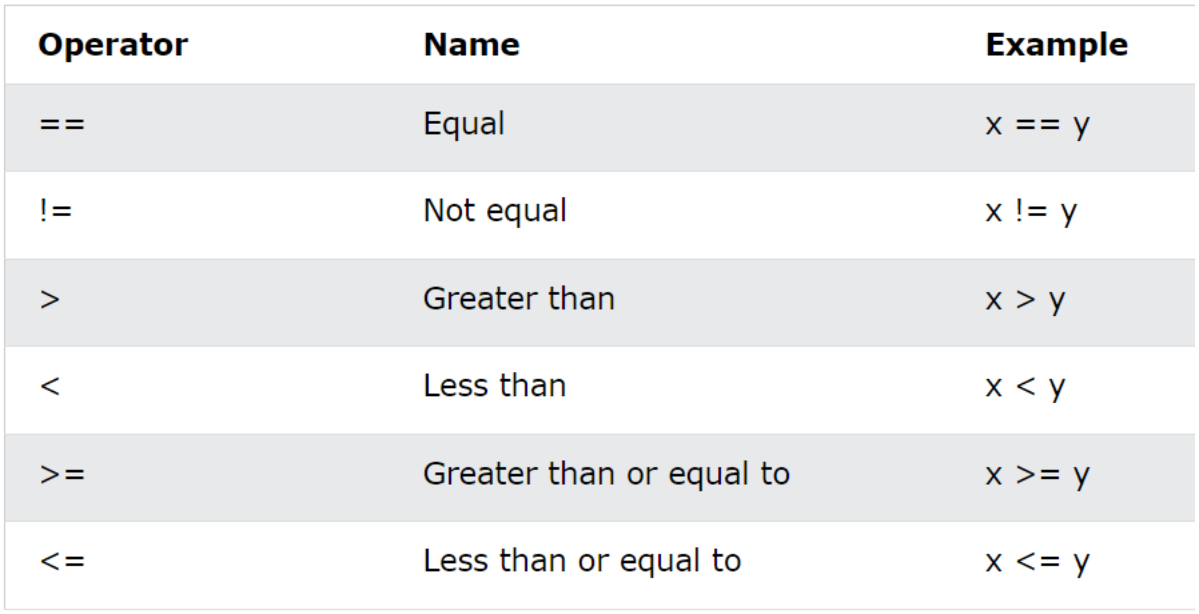
```
(True and True) and (True == False)
```

```
False
```

```
(not False) or (not True)
```

```
True
```

5. What are the six comparison operators?



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

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

= operator

The “=” is an assignment operator used to assign the value on the right to the variable on the left.

For example:

```
a = 10;
b = 20;
ch = 'y';
```

== operator

The ‘==’ operator checks whether the two given operands are equal or not. If so, it returns true. Otherwise it returns false.

For example:

```
5==5

#This will return true.
```

```
True
```

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')
```

```
spam = 0
if spam == 10:   #Block 1
  print('eggs')
if spam > 5:     #Block 2
  print('bacon')
else:
  print('ham')
print('spam')    #Block 3
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.

```
spam = int(input("Enter the value of spam"))
if spam == 1:
  print("Hello")

elif(spam == 2):
  print("Howdy")

else:
  print("Greetings!")
```

```
Enter the value of spam1
Hello
```

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

An infinite loop is a loop that runs indefinitely and it only stops with external intervention or when a break statement is found. You can stop an infinite loop with CTRL + C .

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

Break statement stops the entire process of the loop. Continue statement only stops the current iteration of the loop. Break also terminates the remaining iterations. Continue doesn't terminate the next iterations; it resumes with the successive iterations.

Break:A break statement in Python alters the flow of a loop by terminating it once a specified condition is met.

Continue: The continue statement in Python is used to skip the remaining code inside a loop for the current iteration only.

Break:

```
for num in range(0,10):
    if num == 5:
        break
    print(f'Iteration: {num}')
```

```
Iteration: 0
Iteration: 1
Iteration: 2
Iteration: 3
Iteration: 4
```

Continue:

```
for num in range(0,10):
    if num == 5:
        continue
    print(f'Iteration: {num}')
```

```
Iteration: 0
Iteration: 1
Iteration: 2
Iteration: 3
Iteration: 4
Iteration: 6
Iteration: 7
Iteration: 8
Iteration: 9
```

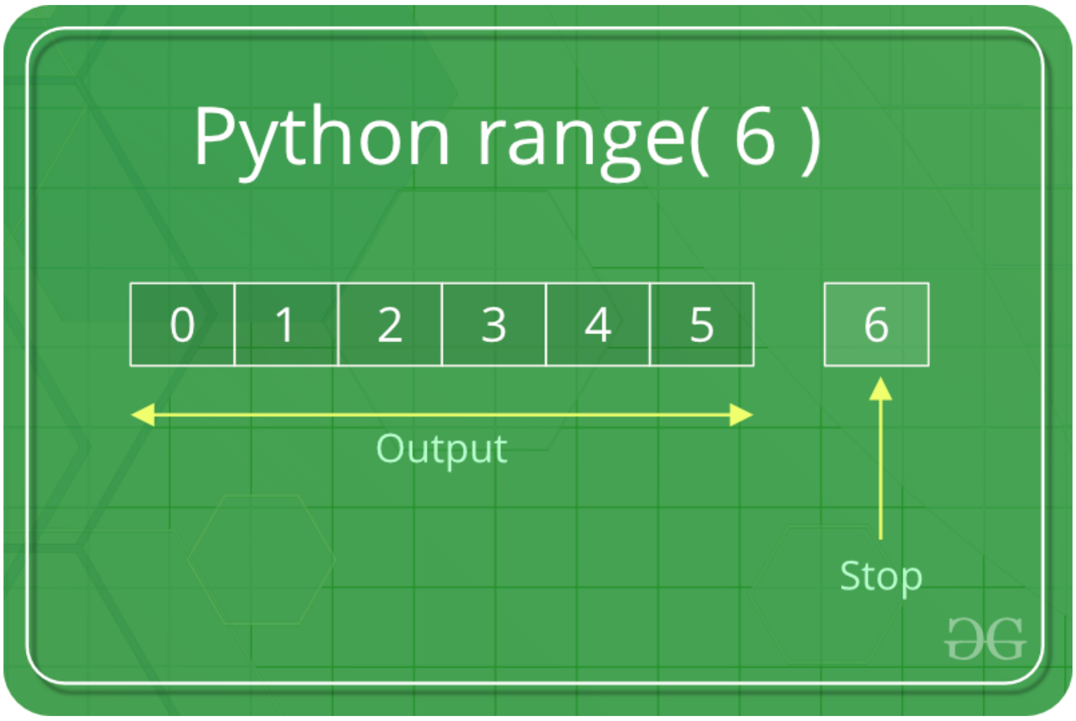
11. In a for loop, what is the difference between range(10), range(0, 10), and range(0, 10, 1)?

In simple terms, range() allows the user to generate a series of numbers within a given range. Depending on how many arguments the user is passing to the function, the user can decide where that series of numbers will begin and end, as well as how big the difference will be between one number and the next. Python range() function takes can be initialized in 3 ways.

```
range (stop) takes one argument.
range (start, stop) takes two arguments.
range (start, stop, step) takes three arguments.
```

i) **range (stop)**

When the user call range() with one argument, the user will get a series of numbers that starts at 0 and includes every whole number up to, but not including, the number that the user has provided as the stop.



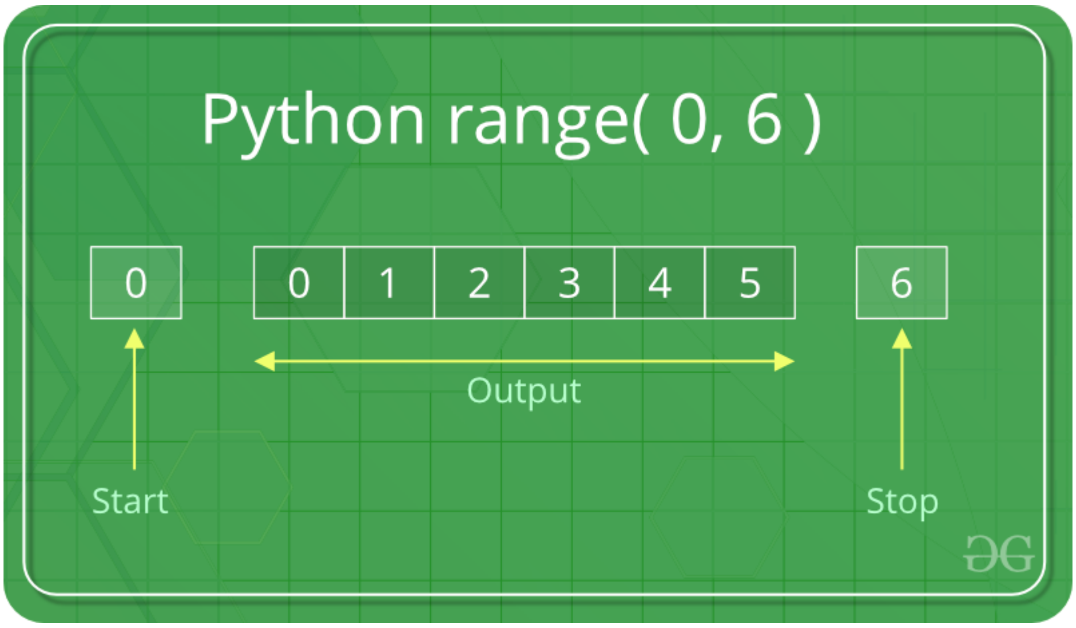
```
# printing first 10
# whole number
for i in range(10):
    print(i, end=" ")
print()
```

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

Above code print values from 0 to 9.

ii)range (start, stop)

When the user call range() with two arguments, the user gets to decide not only where the series of numbers stops but also where it starts, so the user doesn't have to start at 0 all the time. Users can use range() to generate a series of numbers from X to Y using range(X, Y).

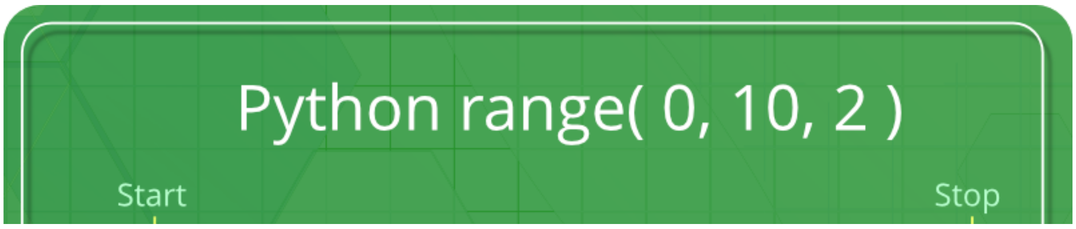


```
# printing a natural
# number from 0 to 10
for i in range(0, 10):
    print(i, end=" ")
```

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

iii) range (start, stop, step)

When the user call range() with three arguments, the user can choose not only where the series of numbers will start and stop, but also how big the difference will be between one number and the next. If the user doesn't provide a step, then range() will automatically behave as if the step is 1. In this example, we are printing even numbers between 0 and 10, so we choose our starting point from 0(start = 0) and stop the series at 10(stop = 10). For printing an even number the difference between one number and the next must be 2 (step = 2) after providing a step we get the following output (0, 2, 4, 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.

```
for i in range(1, 11):
    print(i, end=" ")
```

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

```
i=1
while i < 11:
  print(i,end="  ")
  i=i+1
```

```
    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.beacon()

Above statement will invoke the fuction beacon from the module spam

Colab paid products - Cancel contracts here

0s completed at 5:32 PM