Conditions and Branching

In this video, you will learn about conditions and branching.

**Comparison operations**

Equal (equality) ==

Identity = is

Not equal !=

Greater than >

Less than <

Greater than or equal to >=

Less than or equal to <=

**Numbers & Floats:**

Comparison operations compares some value or operand. Then based on some condition, they produce a Boolean.

**equality operator**

Let's say we assign a value of a to six. We can use the equality operator denoted with two equal signs to determine if two values are equal. In this case, if seven is equal to six. In this case, as six is not equal to seven, the result is false.

Test is 6

a==6

a==7

1=2 this results in a “SyntaxError:can’t assign to a literal”

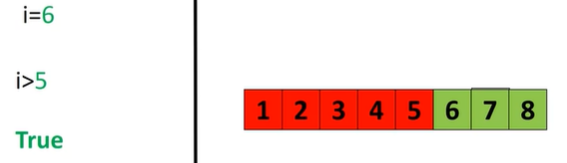
If we performed an equality test for the value six, the two values would be equal. As a result, we would get a true.

Consider the following equality comparison operator:

i > 5

If the value of the left operand, in this case, the variable i is greater than the value of the right operand, in this case five, the condition becomes true or else we get a false.

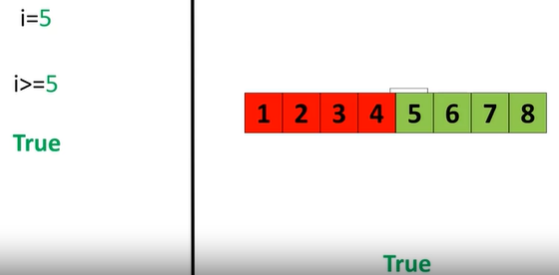
Let's display some values for i on the left. Let's see the value is greater than five in green and the rest in red. If we set i equal to six, we see that six is larger than five and as a result, we get a true.



We can also apply the same operations to floats. If we modify the operator as follows, if the left operand i is greater than or equal to the value of the right operand, in this case five, then the condition becomes true. In this case, we include the value of five in the number line and the color changes to green accordingly. If we set the value of i equal to five, the operand will produce a true.

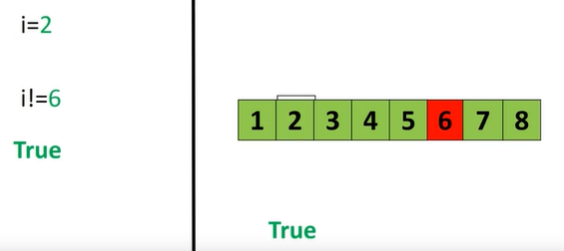
i=5

i>5



If we set the value of i to two, we would get a false because two is less than five. We can change the inequality if the value of the left operand, in this case, i is less than the value of the right operand, in this case, six. Then condition becomes true.

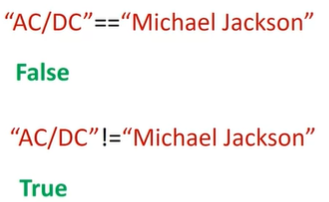
Again, we can represent this with a colored number line. The areas where the inequality is true are marked in green and red where the inequality is false. If the value for i is set to two, the result is a true. As two is less than six. The inequality test uses an explanation mark preceding the equal sign.



If two operands are not equal, then the condition becomes true. We can use a number line. When the condition is true, the corresponding numbers are marked in green and red for where the condition is false. If we set i equal to two, the operator is true as two is not equal to six.

**Strings:**

We compare strings as well. Comparing ACDC and Michael Jackson using the equality test, we get a false, as the strings are not the same. Using the inequality test, we get a true, as the strings are different. See the Labs for more examples.



**Branching**

**IF Statement**

Branching allows us to run different statements for a different input. It's helpful to think of an if statement as a locked room. If this statement is true, you can enter the room and your program can run some predefined task. If the statement is false, your program will skip the task.

For example, consider the blue rectangle representing an ACDC concert. If the individual is 18 or older, they can enter the ACDC concert. If they are under the age of 18, they cannot enter the concert. Individual proceeds to the concert their age is 17, therefore, they are not granted access to the concert and they must move on. If the individual is 19, the condition is true. They can enter the concert then they can move on.

This is the **syntax** of the if statement from our previous example. We have the if statement. We have the expression that can be true or false. The brackets are not necessary.

age=17

if(age>18);

print(“you can enter”)

print(“move on”)

We have a colon. Within an indent, we have the expression that is run if the condition is true. The statements after the if statement will run regardless if the condition is true or false. For the case where the age is 17, we set the value of the variable age to 17. We check the if statement, the statement is false. Therefore the program will not execute the statement to print, "you will enter".

In this case, it will just print "move on". For the case where the age is 19, we set the value of the variable age to 19. We check the if statement. The statement is true. Therefore, the program will execute the statement to print "you will enter". Then it will just print "move on".

In the video, if **age=18** what would be the result

Ans: you have to be greater than 18 so you would not be able to enter, and move on would be printed.

**ELSE Statement**

The else statement will run a different block of code if the same condition is false. Let's use the ACDC concert analogy again. If the user is 17, they cannot go to the ACDC concert but they can go to the Meat Loaf concert represented by the purple square. If the individual is 19, the condition is true, they can enter the ACDC concert then they can move on as before.

The **syntax** of the else statement is similar.

[If statement, without indentation (will raise an error):]

age=18

if(age>18):

print("you can enter")

else:

print("go to see Meat Loaf")

print("move on")

We simply append the statement else. We then add the expression we would like to execute with an indent. For the case where the age is 17, we set the value of the variable age to 17. We check the if statement, the statement is false. Therefore, we progress to the else statement.

We run the statement in the indent. This corresponds to the individual attending the Meat Loaf concert. The program will then continue running.

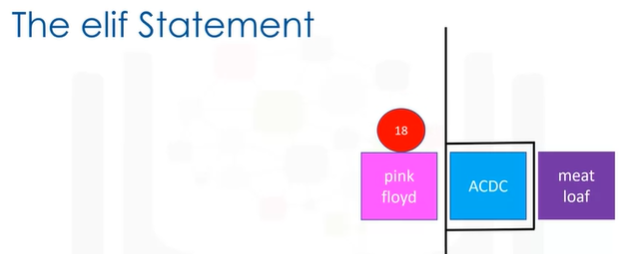
For the case where the age is 19, we set the value of the variable age to 19. We check the if statement, the statement is true. Therefore, the program will execute the statement to print "you will enter".

The program skips the expressions in the else statement and continues to run the rest of the expressions.

**ELIF Statement**

The elif statement, **short for else if**, allows us to check additional conditions if the preceding condition is false.

If the condition is true, the alternate expressions will be run. Consider the concert example, if the individual is 18, they will go to the Pink Floyd concert instead of attending the ACDC or Meat Loaf concerts. The person of 18 years of age enters the area as they are not over 19 years of age. They cannot see ACDC but as their 18 years, they attend Pink Floyd. After seeing Pink Floyd, they move on.



The syntax of the elif if statement is similar. We simply add the statement elif with the condition. We then add the expression we would like to execute if the statement is true with an indent. Let's illustrate the code on the left.

age=18

if(age>18):

print("you can enter")

elif(age==18):

print("go to see Pink Floyd")

else:

print("go to see Meat Loaf")

print("move on")

An 18 year old enters. They are not older than 18 years of age. Therefore, the condition is false. So the condition of the elif statement is checked. The condition is true. So then we would print "go see Pink Floyd". Then we would move on as before.

If the variable age was 17, the statement "go see Meat Loaf" would print. Similarly, if the age was greater than 18, the statement "you can enter" would print. Check the Labs for more examples.

**Logical Operators**

Now let's take a look at logic operators.

Logic operations take Boolean values and produce different Boolean values.

**Not Operator:**

The first operation is the not operator.

If the input is true, the result is a false.

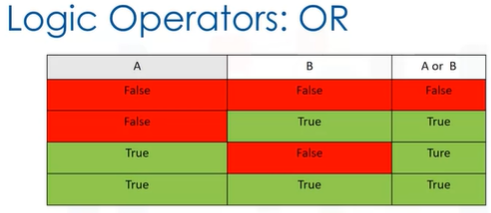
If the input is false, the result is a true.

Let A and B represent Boolean variables.

**OR operator:**

The **OR operator** takes in the two values and produces a new Boolean value.

We can use this table to represent the different values.



The first column represents the possible values of A. The second column represents the possible values of B. The final column represents the result of applying the OR operation.

**We see the OR operator only produces a false if all the Boolean values are false.**

The following lines of code will print out:

album\_year = 1990

if(album\_year<1980) or (album\_year>1989):

print("The Album was made in the 70's or 90's")

else:

print("The Album was made in the 1980's")

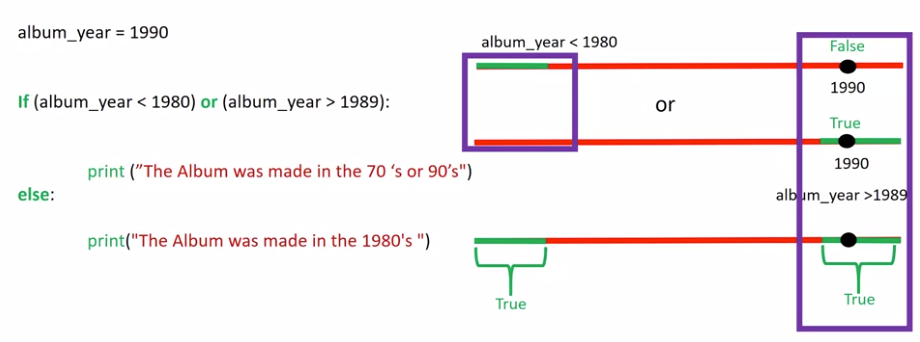
"This album was made in the 70s' or 90's", if the variable album year does not fall in the 80s.

Let's see what happens when we set the album year to 1990. The colored number line is green when the condition is true and red when the condition is false. In this case, the condition is true.

Examining the second condition, we see that 1990 is greater than 1989. So the condition is also true.

We can verify by examining the corresponding second number line. In the final number line, the green region indicates where the area is true. This region corresponds to where at least one statement is true. We see that 1990 falls in the area.

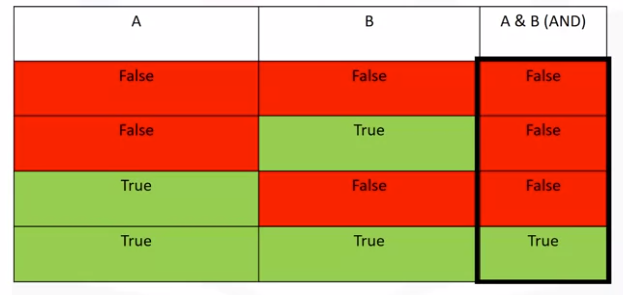
Therefore, we execute the statement.



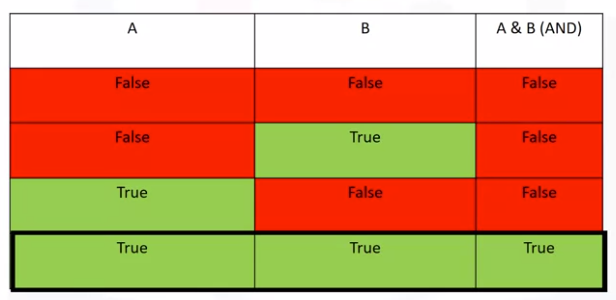
**AND operator**

Let A and B represent Boolean variables. The **AND operator** takes in the two values and produces a new Boolean value.

We can use this table to represent the different values. The first column represents the possible values of A. The second column represents the possible values of B. The final column represents the result of applying the AND operation.



**We see the OR operator only produces a true if all the Boolean values are true.**



The following lines of code will print out "This album was made in the 80's" if the variable album year is between 1980 and 1989.

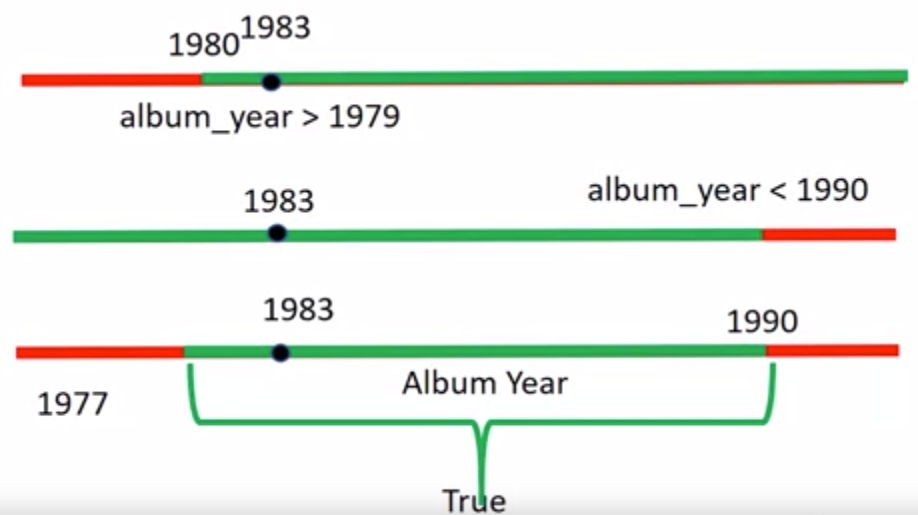
album\_year=1983

if(album\_year>1979) and (album\_year<1990):

print("The Album was made in the 1980's")

Let's see what happens when we set the album year to 1983. As before, we can use the colored number line to examine where the condition is true. In this case, 1983 is larger than 1980, so the condition is true.

Examining the second condition, we see that 1990 is greater than 1983. So this condition is also true.



We can verify by examining the corresponding second number line. In the final number line, the green region indicates where the area is true.

Similarly, this region corresponds to where both statements are true. We see that 1983 falls in the area. Therefore, we execute the statement. Branching allows us to run different statements for different inputs.

**And, Or, Not operators** can be summarised in these tables

