Pīkau Name: Communicating well when programming (CT P0 5)

Video Name: Comparative and Logical Operators (EMP10-7)

Presenters: Tim Bell

In computational thinking Progress Outcome 5 covers '... selection using comparative and logical operators…' Selection is primarily the 'if' statement. The 'if' statement allows a program to take two different branches based on some sort of decision or selection. The 'comparative operators' we've already come across (typically '<' or '=' or '>') can be used to compare two numbers (for example which number is the highest). They also can compare two strings or strings of characters (such as which word comes earlier in the alphabet). In more advanced contexts they can be used to compare all sorts of things such as dates (which date comes first, is this date beyond the threshold for the discounts and things like that).

The comparative operator is quite important but on it's own we can only compare two things and make a decision. That's where the 'logical operators' come in. Let's have a look at some of the logical operators that allow us to combine different comparative operators.

The most common logical operators are 'and', 'or' and 'not'. 'And' is when we want two conditions both to be met. 'Or' is when we don't mind which condition is met. 'Not' is when we want the opposite of that condition. It's going to be easier if we look at some examples.

I've put a program together here that asks a user what their age is and stores that in a variable. We are going to use some comparative operators on the 'age of user' and we will start combining them with logical operators. A simple one to start with. Let's just check with that age if the person is a preschooler. That would be if the 'age of the user' is '< 5'. I'm going to take some fairly simple definitions. If that's the case then we will say 'you are a preschooler'. Let's run this and test it. Asking 'What is your age?' Let's put in a '3'. It says 'you are a preschooler'. If I run the program again. Let's just test it with the number 5. It doesn't say anything because that's outside the range: I said it must be '< 5'. Just another quick test. We'll put in '4'. '4' counts as a preschooler. So that's a comparative operator, the '<'. The result of that of course is a Boolean result, a true or false result. If we just take that out and double click on it at the moment it's saying that that's true. That comparison is either going to be true or false. That's the nature of what we have to feed to the 'if' statement.

Let's try something a little bit different. This time I'm going to look at whether someone is old enough to vote. So we need another 'if' statement and we need to compare their age. In this

case you need to be over 18 to vote. Of course it's not being > 18. Your age can be ≥ 18. Interestingly in Scratch there is no ≥ but there is a way we can achieve that (in fact there's a couple of ways of doing it). I'm going to use it to introduce a particular operator, a 'logical' operator called 'not'. This basically turns the logic upside-down. It's very much like when people end a sentence with the word 'not'. Like "Today is really sunny, not!" It means it's the opposite of whether that statement was true or false. The way that I'm going to do it here is we'll compare the 'age of user' with 18 but I'm saying 'are they under 18?'. Now that actually means that they can't vote. Then here we have an operator for 'not' and if I feed that result into the 'not' operator then that will change 'true' to 'false' and 'false' to 'true'. Sounds useful. Let's just test it. We'll get the cat to say that if the 'age of user' is 'not < 18' or more specifically if it's not true that their age is '< 18' then 'You can vote.' A lot of this is trying to word it in English. English can be a bit loose but this is very logical the way that it works here. Let's run this. I'll say that my age is 18. It says that I can vote. We'll run it again and we'll say that my age is 4. It says that I'm a preschooler but it's not saying that I can vote. Let's put in an age of 17 because that's near the boundary, it's always good to test that. Doesn't say anything because it's not allowing me to vote. Then say my age is 20, I can vote.

We've got the 'not' operation, that's a logical operator. Another operator that we can see in here, in fact the main logical operators are these three here, 'and', 'or' and 'not'. The 'and' operator requires two conditions to be true. An example with age that I could use there is whether someone is a teenager. There's two conditions that have to be true about your age. You have to be 12 or over and you have to be 19 or younger. Actually you have to be over 12 and under 20 is probably the easier way of doing it. I'll try putting together an 'and' operation for that. The first one says that my 'age of user' has to be '> 12' if I'm a teenager but there's this other condition that my 'age of user' must be '< 20'. If both of those conditions are met then it's reasonable to say 'You are a teenager'. Let's try running that program. If I put in 13 for the age 'You are a teenager' because both of the conditions are true. The 'age of user is > 12' *and* the 'age of user < 20'. Let's just do another test. Let's suppose the age is 25. The voting triggers but not the teenage thing. I'll see if I can trigger a couple of them. If your age is 19 then you can vote and then it goes on in the sequence and says 'You are a teenager' as well. We are starting to build up this app that can tell you what you are entitled to do depending on your age.

The 'or' operation is where either of the conditions can be true. For example maybe we are giving a discount if either someone is very young or very old. We'll put another one in here this time using the 'or' operation. This time we'll say that if you are under… whoops. This is one tricky thing about blocks actually. After a while when you get too many of these they get a bit hard to use which is why people prefer text based languages but we'll get there. We are going to say if your 'age of user < 5' 'or' if your 'age of user > 65' then you can get a discount. Test out my program. We'll put in the age of 2. It's registered as a preschooler *and* it says that I get a discount. If we try out the age of 67 I can vote and also I get a discount. Where as if I try something in between like 62 I can vote but no discount. So we are using these combinations of logical operations. We can actually combine many of them.

Here's another example where I need to use a combination of logical operators to achieve a decision. What I'm doing here is I'm writing a program. Maybe it's going to become an app to tell me the best days for line fishing. This is based on the day of the lunar month. Let's just assume that the user can tell us what day of the month it is. Normally you try and get that from the system's clock or something like that so your app can pop up and say 'today's a good day for line fishing.' There are some maramataka that say the best days for fishing are the first six days of the month or the last seven days or Day 20. So we've got this logical condition or many different values that we want to fit. Here we've got a list of days. I've put them in a comment so I can see them while I'm working on things.

Now I need to put in an 'if' statement to figure out if it's going to be a good day for line fishing. Let's just put in the 'say' already. Say 'It's a good day for line fishing.'. The condition here is kind of interesting because for the day to be number 1 - 6 there's many ways we can do that logically. One simple way I could do it is basically find out if the 'lunar day' (notice I've named it 'lunar day' just to be clear to anyone else reading this what sort of day it is) is '< 7'. That is checking really if it is between 1 and 6. Another condition of course will be if the day is '= 20'. So we will just do the 'lunar day = 20'. I'm just putting together the components of my logic. I'm not putting them into the 'if' statement yet. Then finally it's if it's at the end of the month, if the day is after day 23 so 24 onwards (we'll put 'lunar day > 23) then that will be accepted.

Now here's where there is a bit of a twist between English and pure logic because the English says the best days are days 1-6, Day 20 *and* days 24 [and greater] but in pure logic if we say 'is it Day 1 *and* Day 2 *and* Day 20' and so on it will never be Day 1 and Day 20 at the same time. That cannot happen. What's meant here is an 'or'. If it's Day 1 *or* if it's Day 2 *or* if it's Day 20 *or* if it's '> 23'. That's where I need to use my logical operator which is the 'or' operator that will accept if either of these conditions is true. When I say 'either' it's got two possibilities. I've got *three* conditions that I want. This is where we are getting a little beyond the logic that students might need but it's good to know we can actually put an 'or' within an 'or' and that will accept any of them. So we will put the 'lunar day = 20' in there and the 'lunar day > 23' there. What this is saying is that (the left hand side here) 'lunar day < 7' if that is true we'll just say it is a good day for fishing. *Or* if that doesn't work let's see if the right hand side is true which is one of these two things can be true and so on. We'll put this up into my 'if' statement. I'm now getting quite a complicated logical decision going for what was a relatively simple rule. Let's run it and see if the cat is good at giving advice about fishing. 'Which day of the lunar month is it?' Let's start off with Day 3. 'It's a good day for line fishing'. Try again with something that's out of range. According to the rules up there Day 7 isn't. If I type in a '7' no result. Let's check out the Day 20 is working okay. 'Good day for line fishing.' Day 24 should be included but let's just check that Day 23 doesn't trigger it so try that. No, no result. Day 24 of the lunar month let's say that. 'It's a good day for line fishing.' So there is a rule that obeys this tradition that those are good days for line fishing. It's embedded in a logical operator.

I've got a very manual system here where you type in the month. That kind of logic would be buried inside an app. Maybe it's an app that automatically starts up each day and figures out the

lunar day and puts it through that logic and sends through an alert to tell you that it's time to go line fishing.

So that's comparative and logical operators. The comparative operators allow us to compare values but by combining them with logical operators we can come up with quite complex conditions. The kind of conditions that happen in real life when several requirements have to be met *and* you have unusual conditions that you are looking for in the data *and* you are making decisions about how your program's going to proceed based on the data.