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

Video Name: The same concepts in Python (EMP10-6)

Presenters: Tim Bell and Joanne Roberts

Tim: So we've looked at different kinds of loops in Scratch so I thought it would be useful to look at it in a text based programming language. We are going to use Python again. With these kind of constructs they are pretty similar in most text based languages. First of all I'm going to start with a comment because good programs have a comment…

Joanne: You mean a hash?

Tim: The hash is the symbol for a comment. Yes. We are going to do the program to get someone's year of birth and just check that it's vaguely ok. So 'get the year of birth'. Actually rather than 'the' year of birth it's the user's year of birth. I'm going to put the apostrophe there.

Joanne: Just to be proper?

Tim: Just to be proper. But not just proper. That way [written as "users"] technically in English I am saying that there are many users. If I'm glancing at this program that is written I think it's getting it for all the different users but then I think no, the programmer just had bad grammar. Some people will struggle to get that right but the point is that if you get it right it's going to help down the road to have the programs working properly. So 'get the user's year of birth'. Now the variable I'm going to store that in I'm going to call 'year_of_birth'. Again it could have been called 'year' but you don't know what year it is so I'm picking a name that's reasonable but not too long that goes on forever. In most programming languages you can't have a space in the middle of a variable.

Joanne: No.

Tim: So underscores or dashes are quite common. In Python it's conventional to use underscores so I'll use those so someone who's used to Python doesn't look at it and go 'oh! Hang on!'. You know, they've got that extra hesitation 'is it a variable or not?'. Okay. 'Year_of_birth'. It's going to turn it into an integer and get it from the input. You don't want to worry too much about the details. Integer is that type of variable. It's a number.

Joanne: It's a whole number.

Tim: A whole number as opposed to a float. We don't have fractional years, or at least not here. So I'll say 'Please enter your year of birth'. It's just gone onto two lines so I'll make that a bit wider so it's a bit easier to read. I've blown up the font on here so it's a lot bigger than you'd normally have. So we've received it but now we want to keep going and check until it's valid. In Python the most common way of looping around like that is called 'while'. Then we put in the condition. So it's 'while the year_of_birth' is *wrong* which means... what's the current year? 2018 so if 'while year_of_birth > 2018:' then it's wrong. That colon is kind of the thing that says 'do the following sequence'. So if the 'year_of_birth' seems to high then we are going to ask for the year of birth again. I'm just going to copy this line. We might change the wording a wee bit. 'Seems to high year of birth:'. Again that's probably a wee bit abbreviated but I can work on that another time. In fact that's the whole thing! It's getting a new 'year_of_birth' and that 'while' loop will keep on asking that until it fails on that condition.

Joanne: So even though there's nothing actually under that second 'year_of_birth' line' it knows to keep going back to that line?

Tim: Yes. The reason it knows is that my next line is indented out near the margin here. So we can say 'print (year_of_birth)'. That's going to output or we could do something with it, work out how old they are. So everything that's indented there on line 5 (which could be a sequence of things, as it happens we only needed one thing there), that will keep on looping. But, let's test it. Run that and this time I'll make this window over here a bit bigger so we can see everything that's happening. 'Enter a year of birth'. What do you want?

Joanne: 1999.

Tim: 1999. It's just printed it out because that is valid. I'll run my program again because we need to test it with other things.

Joanne: 2018?

Tim: 2018? That's an interesting one. Will it let me do 2018? Yes it does?

Joanne: Because it is not greater than 2018 yes.

Tim: Now it's unlikely someone born in 2018 was actually typing into this program but someone might have been doing it on behalf of someone so we need to allow for that.

Joanne: What about 1808?

Tim: 1808.

Joanne: My great great grandmother is doing this.

Tim: Okay! Well that’s a valid year of birth. So that seems alright. Of course when we are testing we need to check that this ‘while’ loop is working so we better check something that’s not going to be valid. What’s a good invalid date that we will try?

Joanne: 2020.

Tim: 2020. Good round number. ‘Seems too high, what year of birth?’ This is where we want to be sure it will keep on going until we get it right so let’s put in 2100. ‘Seems too high, what year of birth?.’ And then there is the boundary one: 2019. Basically we won’t know that someone is born that year so that ‘seems too high.’ Then let’s just check the year 2000. That’s fine and now it goes ahead. So this ‘while’ loop is just going to keep going round and round. But it is a very definite sequence of things. Line 4 says ‘check that condition’ if it’s true, which you’re actually hoping it won’t be but if it is then it asks again. Then it goes back and checks the condition, asks again. So that’s the order in which things are going to happen over and over until we enter a valid year of birth.

So all of those things we were doing in Scratch apply here. Good documentation, comments to explain what’s happening, probably wouldn’t hurt to put in here ‘check that the year is valid’. And we can add other things because if it was an actual living person then if it was more than 150 years ago then you would probably query it as well. They probably made a typing mistake. Those comments: the computer completely ignores them, it just helps the person who is reading it to know what’s going on there. We’ve got variable names that hopefully mean something to the next person because we are communicating with the next programmer or the teacher and with ourselves. I know exactly what I meant by that. And that’s what it would look like in a different language.

Joanne: Fantastic.