Introduction to Programming in Python

Trainer Notes

# Resources needed

Computers with internet access  
Python 3.x  
Headphones if teachers want to recap on elements from <http://pythonschool.net>

# Practical elements

Computers– suitable for pair or individual work  
Coffee/tea for 11am and 2.30  
Lunch break at 12.45

Box for demonstrating what a variable is

# Session 1 (9:45 – 11:10)

Start off with introductions – of trainer(s) and ask all teachers to introduce themselves and say something about their school and where they are with Computing. You also want some indication about how much they know about Computing already as this will help you pair people up so that they can support each other.

## Slide 2: Objectives

It is important that the objectives of the session are clear and what will be covered. We want to evaluate the sessions and it is important that teachers are aware of what the objectives are when they review whether they learned enough from the session. Sometimes they may have some misconceptions about what the goals of the sessions actually are. This slide is shown at the end of the day as well to help guide teachers with their reflections on the session.

This session maps to the following topic areas in the TA Matrix of Range and Content at Primary and Secondary: **A1, A3 A4, D1, P1, P2, P3, P4, P5, P10, P11**

See <http://www.computingatschool.org.uk/data/uploads/CSSubjectKnowledgeRequirements.pdf> for more information

Slide 3: Agenda

## Slides 4-6 Why program? Why Python?

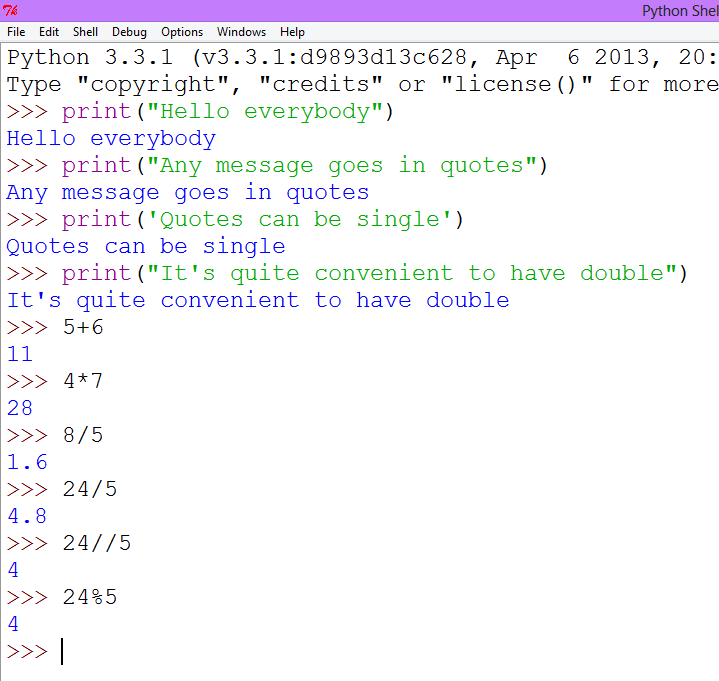
It is useful to put programming into the context of computational thinking before starting. This should be quite short as the teachers will be desperate to get going but I feel it is useful to outline that through learning to program students can develop ways of thinking that will be useful to them in all areas; also that although Python is the most popular language for learning to program at the moment in schools, the basic principles are the same for programming in any language.

## Slide 6 – Demonstration in Python

For the rest of the day you will not need a PowerPoint presentation except to display exercises that teachers will carry out.

## Demonstration – using the interpreter in Idle

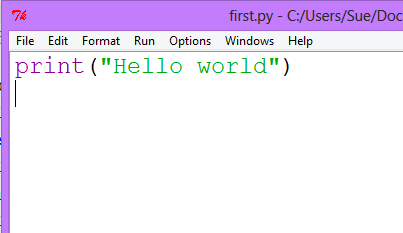
Ask teachers to open the Idle interpreter and try out some basic interpreter commands, following you (make sure you use as large a font as possible in Python)



Also look at statements like 4 = 5 , 4 < 5 , etc. to cover some of the operators and consider conditions that return True or False

## Demonstration – using the editor to save and run a program

The previous demonstration has already covered many key concepts so expect lots of questions. The next stage is to edit, save and run a file – just with one line of code.



After doing this I ask teachers to start a list of all the things that have gone wrong so far and what they would expect the students to have difficulty with. They should come up with a list including:

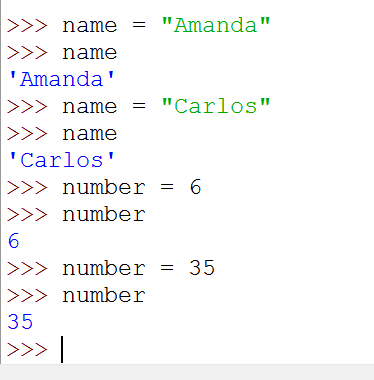
* Writing print with a capital “P”
* Forgetting the quotes
* Saving the interpreter rather than the editor window
* Forgetting to save the file as .py (lose the editor colouring)
* Using the wrong kind of brackets

I encourage teachers to add to this list the whole time they are learning to program (not forgetting to make a note of the fix) so that they feel prepared to troubleshoot when students have difficulties.

## What is a variable?

The next thing to cover is what a variable is – for this use a good-sized cardboard box as a prop. Write Name on the outside then write the name of somebody in the room on a piece of paper and put it in the box – repeat with another person’s name, emphasising that the original value has been overwritten.

Then demonstrate this in Python in the interpreter first.

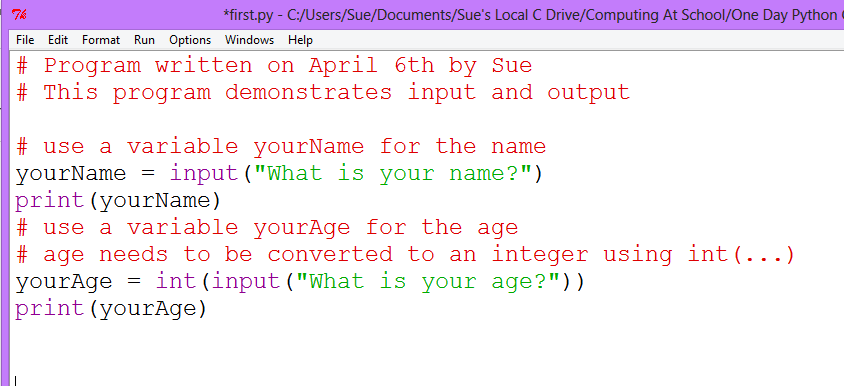


This brings up the idea of different data types, which you can discuss briefly at this point.

## Demonstration: taking input

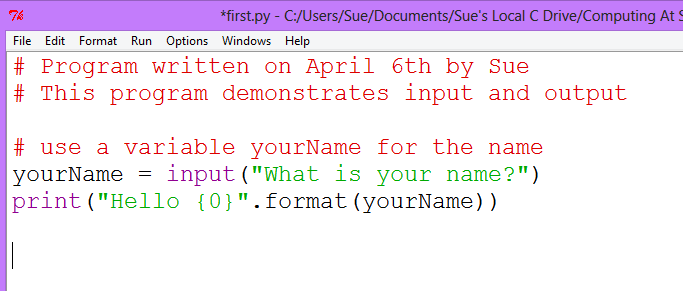
The next concept to introduce is how to take input from the keyboard.

Demonstrate as follows then ask teachers to practice different inputs and outputs. At this point I introduce the idea that each program must have comments in it. Depending on the group, you may delay introducing int(input( …)) for a while later.



## Demonstration: formatting output

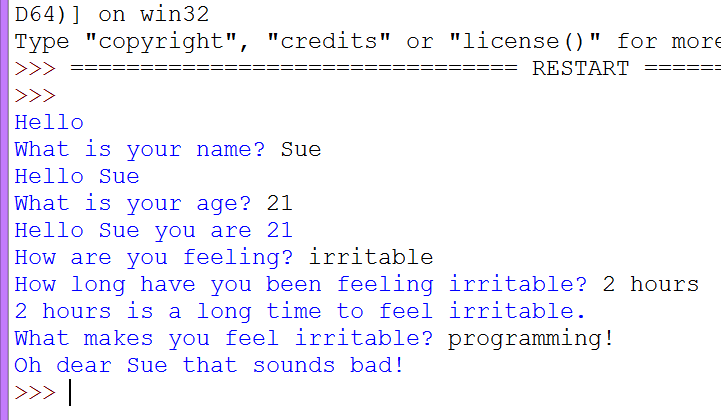
The question will invariably now arise is how you combine variables and strings in a print statement. In Python School we teach the use of the format statement as shown below.

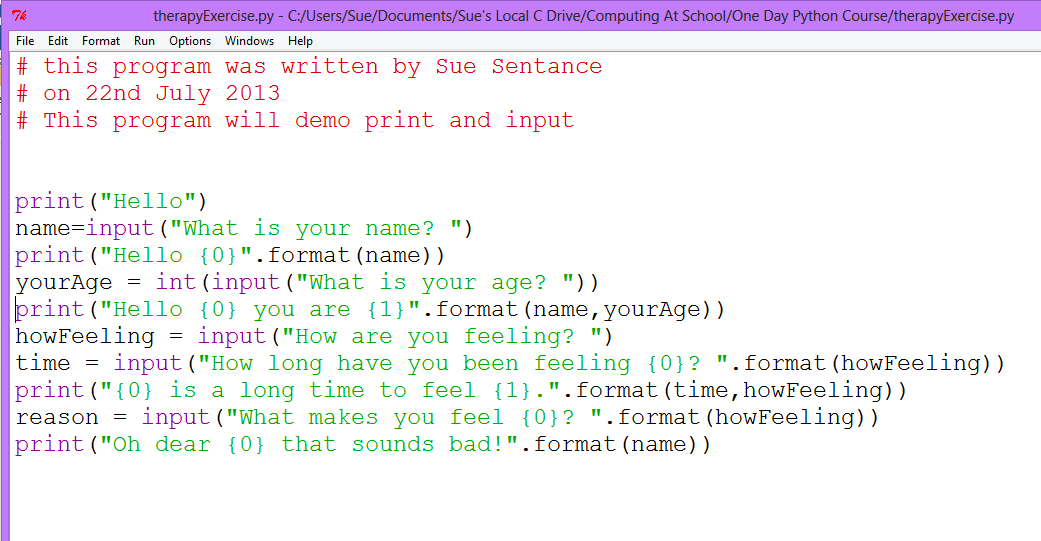


Long-term this is a better way of formatting output for teachers but using placeholders can be a bit painful for teachers at first. However if teachers start using print(“Hello”,yourName) which is not such good practice it will be harder to get rid of the habit later! So I spend some time on this.

## Exercise: Therapy program

By this point a lot has been covered so the rest of the session until coffee is spent in consolidating these skills with an exercise using input and output. An example of the exercise is demonstrated first which is a fun exercise in giving suitable output to the user when you don’t know what the input will be.





COFFEE/TEA

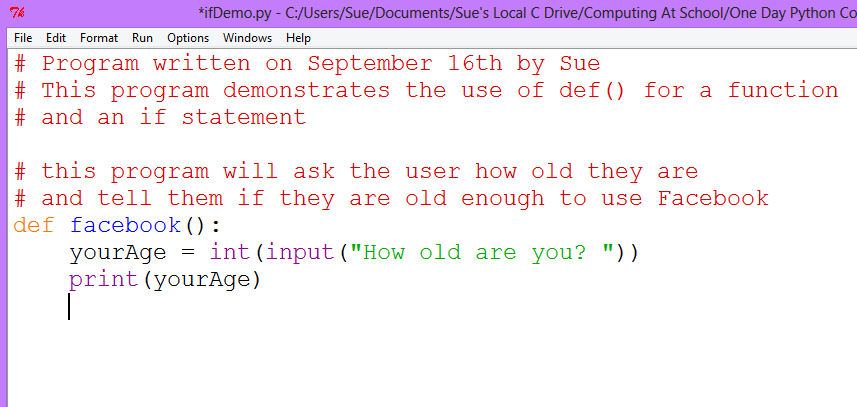
# Session 2 (11:15 – 12:45)

In this session cover the following two main areas:

* Using def() to put statements inside a function (sub-routine)
* Using the if.. elif…else structure to allow choices within the program

The main demonstration now is around a program to see if the person is old enough to use Facebook

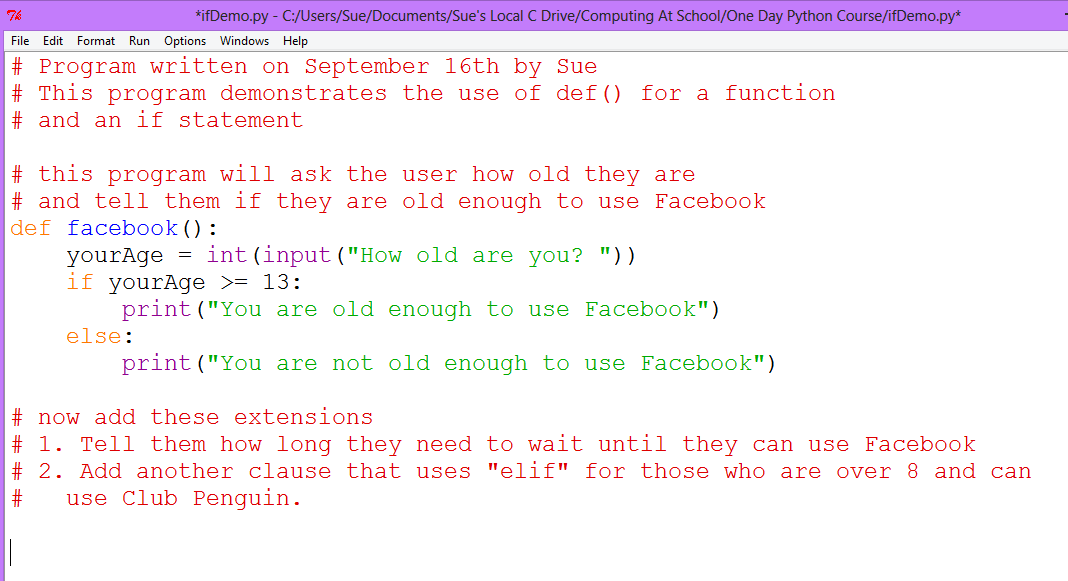
Introduce the idea of def() first which has some syntax for teachers to get used to. Write this program in side a def()



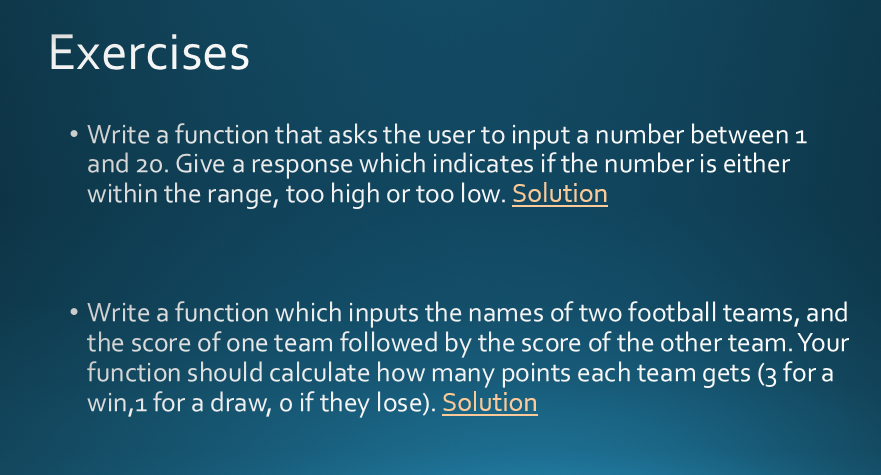
Teachers can test that this much works first because they will need to get used to the fact that they need to CALL the function in the interpreter window.

Then add the if statement. I tend to go over how if works in an Excel spreadsheet first as all ICT teachers are familiar with that – so that the concept is understood before explaining the syntax.

Teachers can copy down the initial if statement which is quite easy then give them two exercises to do as shown in the window below



Once teachers are happy with this, some more practice is needed. I give the following two exercises which are on the Python School website (with solutions).



After about 20-30 mintues on these exercises, a good task to do is the Maze Game. I generally ask teachers to do this in pairs. It introduces an approach which we recommend they use in the classroom which is to

provide a short starter program which starts them off and which they can edit. The advantages of this approach are:

* It is less daunting than starting with a blank Python program
* It gives examples of the structures that can be used
* It will contain examples of good practice such as comments

The Maze game program can be downloaded from here: <http://www.pythonschool.net/basics_task1> . On this page examples of the use of the random function are given for those teachers who are feeling confident and wish to bring in a random component to their game.

The task is to edit the Maze game so that it has more options, more consequences and perhaps introducing a new level. The reason for working in pairs is to allow the teachers to troubleshoot out loud as that helps them to articulate their thinking processes so that they are more aware of them. This task also enables those who are struggling to focus on understanding the program in front of them with minimal changes, and those who are finding it all easy so far to have some more open-ended practice. This introduces differentiation for the first time.

In summary, this exercise includes three pedagogical features:

* Differentiation
* Collaborative working (Working in pairs)
* Scaffolding (a starter program)

Highlight the pedagogical strategies that you are using as a trainer and ask teachers if they think these would be appropriate for their classrooms.

Ask teachers to maintain their list of the errors they have made and the fixes. Ask them what they think the students’ common mistakes would be.

LUNCH BREAK

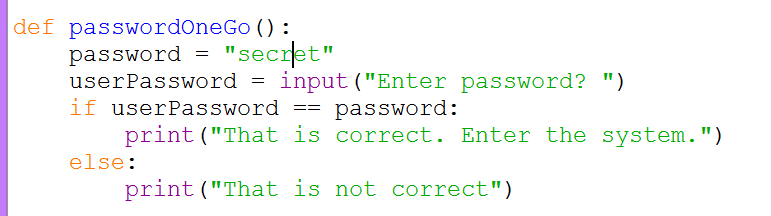
# Session 3 (1:30 – 2:30)

In this session the while loop is introduced to teachers. However it is as important to consolidate the morning knowledge as well as introducing loops.

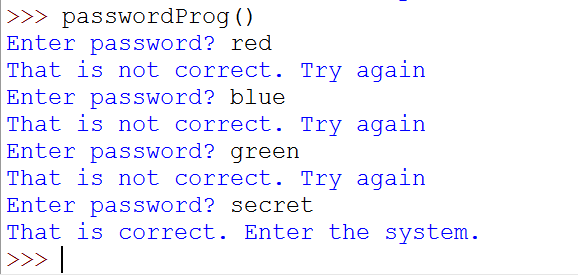
## Exercise: Password

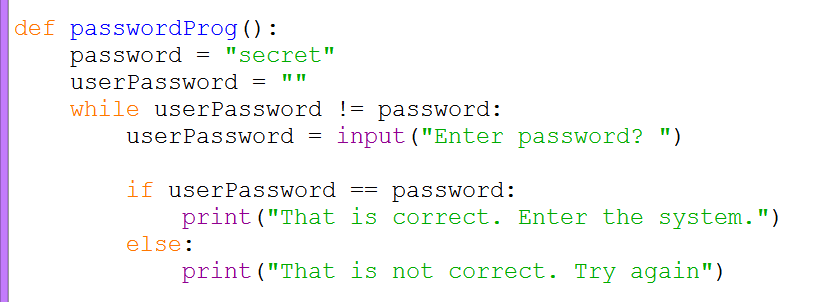
Start with a recap of what was covered in the morning, perhaps writing on the board all the concepts that had been covered.

Then develop the password program together. Start by asking teachers to write a simple password program that gives only one attempt to get the password correct. They should end up with something like this.



Then develop the program together with contributions from teachers that will allow repeated guessing. Demonstrate it working first then build up the code.

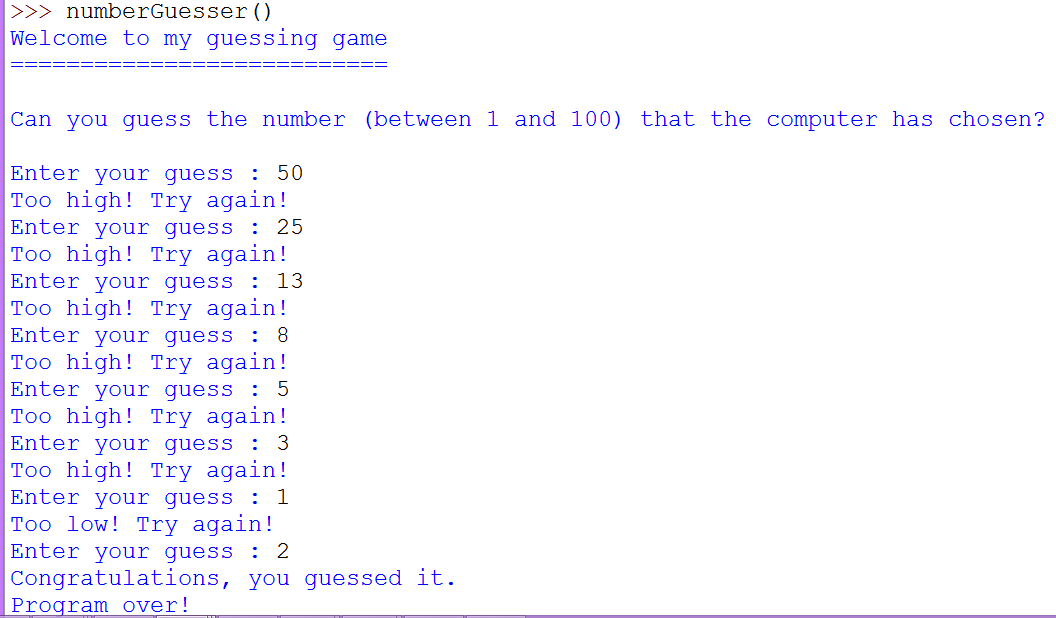




For some extra practice before the guessing game, another nice program to try is “Are you bored yet?”

## Exercise: Guessing game (Slide 16)

Demonstrate the guessing game working.



Then ask teachers to work in pairs again to work out how to program the game.. This is a good opportunity to discuss planning on paper and **developing an algorithm** for the game. Encourage thinking before plunging in.

If teachers struggle with syntax, they could start by building on the exercise that they did in the morning looking at whether a number input was bigger than 20

Some teachers may find this difficult. If so, work through on the board with this group.

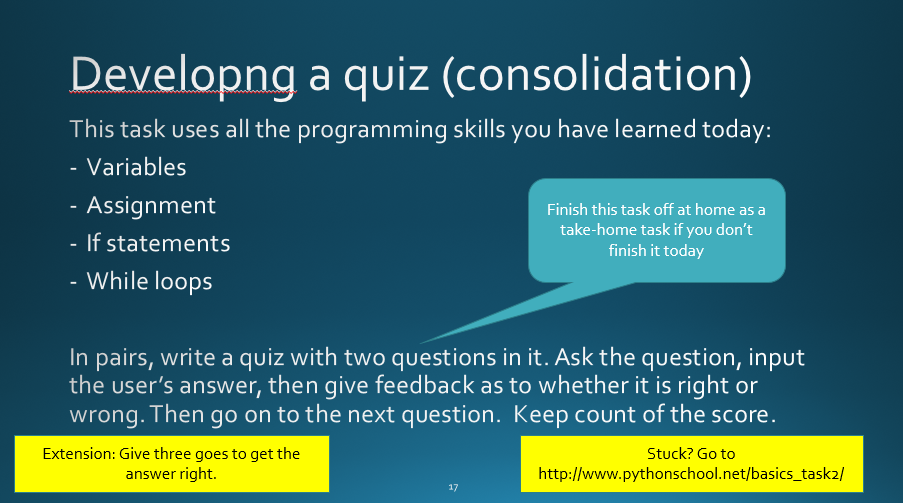
COFFEE/TEA

# Session 4 (2:45 – 4:00)

Again, recap what the difficult concepts are and what teachers could recommend to each other for troubleshooting tips.

## Exercise: Develop a quiz (Slide 17)

The final exercise is to develop a quiz. There is a starter program for this on the Python School website so you can use your judgement whether you use this or not. One option is to give the teachers the starter program and ask them to add another question and a score. If teachers are confident, some of them may want to develop the quiz from scratch.



This is a good exercise in pairs, but if this is not appropriate for your group, some may want to work independently.

On the slide an extension exercise is given but there are lots of ways this could be extended.

Slide 18 – recap learning objectives (5 minutes)

Allow a good fifteen minutes at the end of the session to explore what the teachers have learned and take feedback. Go over the original objectives of the session to help teachers reflect on what they have learned and start to identify next steps. Emphasise a) that this is the beginning of learning to program and should be followed up by a longer course over a period of time, like a set of evening classes, and b) that the new skills will need bedding in with lots of practice at home and trying out in the classroom. Daily practice required! Do show teachers that there are lots of exercises with answers to practice on the Python School website which they should work through.

Slide 19 – complete feedback forms (5 minutes)

The feedback forms are stored here: <http://surveymonkey.com/s/NoEFeedback> . Teachers should fill this in in order to receive their certificate for the course. The NOE will share the feedback with you as soon as it is collected.

Slide 20 – final reflections (5 minutes)

Ask teachers to share with each other what they will do as a result of this course. Encourage them to stay in touch with each other and provide a means for them to share their email addresses. Actually making a commitment to do something is the first step – otherwise the value of the training can be quickly lost.

**Finish at 4pm**