Sonic Pi

Lesson 2

The Art of Programming: Debugging & Iteration

Designed by
Sam Aaron & Carrie Anne Philbin

Introduction

In this lesson we will be given an opportunity to experience the art of programming - from debugging to careful syntax placement to the joy of a working program. Also, in order to to construct some meaningful and interesting musical structures we need to learn some meaningful and interesting programming structures. Today we are going to learn how to create a baseline.

Learning Objectives	**4.2 Programs**
	Programs contain syntax.
	Incorrect syntax creates errors which need to be debugged
	Debugging is a problem solving activity
	Programs can contain interesting and meaningful structures.
	An interesting and useful structure is repetition or iteration.
	Syntax can be used to represent both actions to be performed and also punctuate the structure of the program.

Teaching Progression

For the majority of the lesson, it is suggested that work is carried out in pairs. Each pair should have access to the standard equipment described below. In addition, is suggested that you have your own setup connected with a speaker for the demonstration sections.

Sonic Pi Equipment

A Raspberry Pi with the Sonic Pi software installed per pair;
A keyboard and mouse connected to the RPi per pair;
A monitor connected to the RPi per pair;
A headphone splitter connected to the RPi audio jack per pair;
A pair of headphones connected to the splitter per pupil.

Equipment

• Computational cards with iteration statements.

Lesson Summary

• A short exercise to recap the notion of statements and control flow within a program
• A debugging exercise
• Starting the Sonic Pi application
• Iteration as a repeating structure.
• Syntax punctuating structure.

Starter

If any of the pupils have interesting statements that they created as part of their homework, they are invited to demonstrate them by getting a number of other pupils to act them out. This could be done in groups or with the class as a whole. As a backup the *computer program* cards could be used similar to the previous lesson.

Main/development

1. With just one group acting, and with the rest of the class watching, introduce the two *error* cards into the deck of *computer program* cards, shuffle them up and repeat the exercise described in the previous lesson. Observe what the pupil with the foreign language *error* card does and then have a short discussion with the class about what perhaps should happen. Then point out that there's another *error* card, but with a subtle error in the spelling. Explain that to the computer they are both as unintelligible as each other. The computer isn't clever enough to read through the subtle error.
2. The pupils are then separated into pairs and are given their own set of Sonic Pi equipment. Pupils follow the instructions to connect the various parts of the Sonic Pi together and once the RPi is booted, they start the Sonic Pi program by clicking on the menu item under *start -> programming -> sonic-pi*
3. Next, the pupils are then given their first introduction to debugging. They are asked to type in the following one line program which is incorrect: `pllay 42` They are then asked to observe what happens when they run the program. At this point it's important to emphasise that this is a typical situation for real programmers. It means that they have done something that the computer doesn't understand. Which, because the computer is not very clever, is very easy to do. They are then invited to correct (debug) the program by removing the extra `l`. They are then asked to observe the output of the program.
4. A selection of pupils are asked to form a line with the computational cards and to act out the program as carried out in the previous lesson. The class is then asked how we can repeat this program twice. One of the answers might be to duplicate the line - act this out, forming a line twice as long as the original with the original sequence of statements duplicated twice. Now, ask the class to act out this new longer program. Once this has been completed ask how we might repeat the original program 10 times, or 100 times. What about 1000 times? Clearly we'd run out of people!
5. Introduce the iteration cards: `1.times`, `do` and `end`. Wrap the original program with the `do` and `end` cards and prefix everything with the `1.times` card. Explain that this is just a long-winded way of doing the same thing as the original program. The new cards are extra syntax which should be thought of as the same as punctuation - i.e capital letters, commas and full stops. Like punctuation this syntax helps the computer structure the program. However, with this extra syntax, we have actually formed a circle out of our code. - the syntax defines the start and the end points of the loop - the `do` and `end.` In this case, we only loop round the circle once, but if we replace the first card with `3.times` we loop round our program 3 times. If you have enough space, try forming a circle and enact the loop. Explain that this loop structure is called iteration.
6. Pupils are asked to enter the following 'baseline' code into the Sonic Pi application:

```
  play 42
  sleep 0.5
  play 45
  sleep 0.85
  play 54
  sleep 1
  play 54
  sleep 0.7
  play 45
  sleep 0.2
  play 49
  sleep 1
```

7. They are then asked to repeat this line 5 times. They do this by writing:

```
5.times do
  play 42
  sleep 0.5
  ... rest of code ...
  sleep 1
end
```

8. The pupils are then invited to play around with this modifying it as they feel fit for the remainder of the time. For the adventurous pupils, you can suggest that they nest the iterations - i.e. have an iteration inside another.

Plenary

Each group is invited to demonstrate the program they have written by executing it and playing the resulting music. This can be achieved by passing the speaker round to each group and playing the music through the speaker rather than the headphones.

Homework

Students should write their own musical programs on paper to be discussed in the next lesson.

All students are able to...	Understand that computers don't know what to do do with an error. Write a simple program. Iterate some code a number of times

Most students are able to...	Understand that subtle errors in language are just as unintelligible to the computer as a foreign language is to someone that doesn't speak it. Understand the consequences of their program before they run it and therefore design a musical program that's interesting to them. Understand that the do and end keywords are structural syntax rather than actions such as play.
Some students are able to...	Read an error message and pick out the cause of the error. Understand that iterations can be nested within each other. Use the advanced commands in the pupil notes.