From ICT to Computing

These notes are two-fold. Some comments are to support other people running similar sessions with Boardworks and some are for people to take away and adapt for their own teaching.

Session Outline & Housekeeping

4:15 - Introduction & curriculum changes

4:30 - Workshop 1: Sorting algorithms

5:00 - Break & networking

5:10 - Boardworks Resources

5:20 - Workshop 2: Transition from Scratch to Python

6:00 - Plenary

Introduction

Mark Clarkson
@mwclarkson
mwclarkson@gmail.com
mwclarkson.co.uk

Head of Computing & ICT
CAS Master Teacher
CAS Hub Leader

The obligatory bit of self-promotion :-)

Curriculum, what curriculum?

September 2010: OCR GCSE Computing Pilot launched

January 2012: Gove announces concerns over ICT

September 2012: ICT Programme of Study withdrawn

February 2013: Draft Computing Programme of Study published

September 2014: Computing curriculum compulsory with NC

Quick recap of how we got to where we are and how things are changing

Draft Programme of Study

Pupils should be taught to:

- design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
- understand at least two key algorithms for each of sorting and searching; use logical reasoning to evaluate the performance trade-offs of using alternative algorithms to solve the same problem

Probably best not to go through these in detail (it's not a lecture), but useful for teachers to take away.

Draft Programme of Study

Pupils should be taught to:

- use two or more programming languages, one of which is textual, each used to solve a variety of computational problems; use data structures such as tables or arrays; use procedures to write modular programs; for each procedure, be able to explain how it works and how to test it

- understand simple Boolean logic (such as AND, OR and NOT) and its use in determining which parts of a program are executed; use Boolean logic and wildcards in search or database queries; appreciate how search engine results are selected & ranked

board works

Draft Programme of Study

Pupils should be taught to:

- understand the hardware and software components that make up networked computer systems, how they interact, and how they affect cost and performance; explain how networks such as the internet work; understand how computers can monitor and control physical systems

Draft Programme of Study

Pupils should be taught to:

- explain how instructions are stored and executed within a computer system
- explain how data of various types can be represented and manipulated in the form of binary digits including numbers, text, sounds and pictures, and be able to carry out some such manipulations by hand

Draft Programme of Study

Pupils should be taught to:

- undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting and analysing data and meeting the needs of known users

- create, reuse, revise and repurpose digital information and content with attention to design, intellectual property and audience.

Workshop 1 - Sorting Algorithms

30 minute workshop. Outline plan:
Starter – Kinaesthetic sorting
Main 1 – What is an algorithm
Main 2 – Sorting with scales (extension: efficiency & complexity)
Main 3 – Sorting reply slips

Session Objectives

ALL students will be able to describe one method of comparing and sorting values

MOST students will be able to describe at least two sorting algorithms using appropriate technical language

SOME students will be able to compare the efficiency of different sorting algorithms and evaluate their complexity

A competition!

Stand in order by:

Height

First name (ascending)

Sweets as prizes for the winners (first to get themselves sorted)! Groups of 6-10?

The aim here is to have delegates / students think about how they get themselves into a particular order. It gives a reason to sort as well as a kinaesthetic example. It might well be worth introducing the concept that people are more intelligent than computers because we can sort more ‘cleverly’ - considering a range of values at the same time (whereas a computer can only compare 2 values at a time).

What is an algorithm?

In mathematics and computer science, an algorithm is a step-by-step procedure for calculations. Algorithms are used for calculation, data processing, and automated reasoning. (Wikipedia)

e.g. “Come in, sit down, log on”

Sorting Algorithms

Load the **insertion sort** web page

Follow the instructions to sort the boxes into order

Insertion sort web page currently at mwclarkson.hopto.org/sort/insertion.html (subject to my Raspberry Pi remaining online).

Bubble, Insertion and Selection sorts all provide similar results in terms of efficiency.
Merge and Quick sort should be a bit more efficient - and even moreso for bigger data sets.

Prompt for comparisons of efficiency and complexity.

A Practical Example

In pairs, collect two sets of reply slips

Choose an algorithm to try

Ready

Steady

Sort them in alphabetical order by first name

At this point I will hand out an algorithm card to follow. Again, the task is kinaesthetic. By following the algorithm properly students/delegates (working in paris on just ONE set) should be able to see the pros and cons of that algorithm. (Bubble, Insertion or Selection)

The exercise will then be repeated with the other set of reply slips and with a different algorithm card (Quick or Merge sort).

Students might need something more suited to their own context - but the idea is easily transferable. Football teams with points scores in the league? Eurovision songs and scores? Pretend reality TV voting?

Teachers should take away either a physical or digital copy of the reply slips along with prompt cards for algorithms

Further Resources

- Animated sorting algorithms - http://www.sorting-algorithms.com/
- AlgoRythmics - Sorting algorithms to Hungarian folk dancing - http://www.youtube.com/user/AlgoRythmics/videos

Break and Networking

Followed by Boardworks' presentation

Workshop 2 - Scratch to Python

Scratch

Drag and drop - no typing

Pre-built blocks - no syntax

If blocks / loops

Graphical - instant feedback

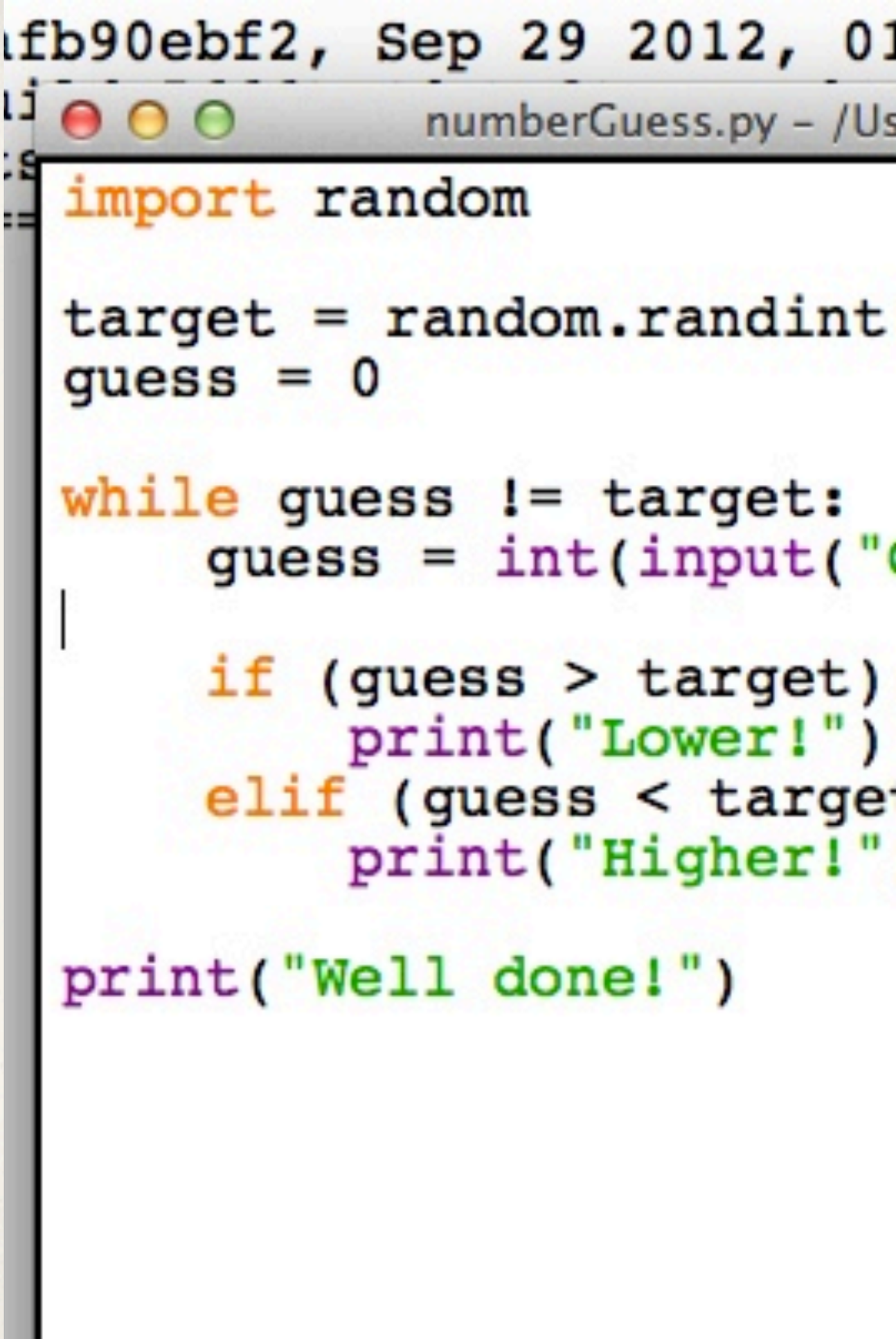
Suitable for EYFS upwards

Python

Text based programming

Syntax matters

Text based output



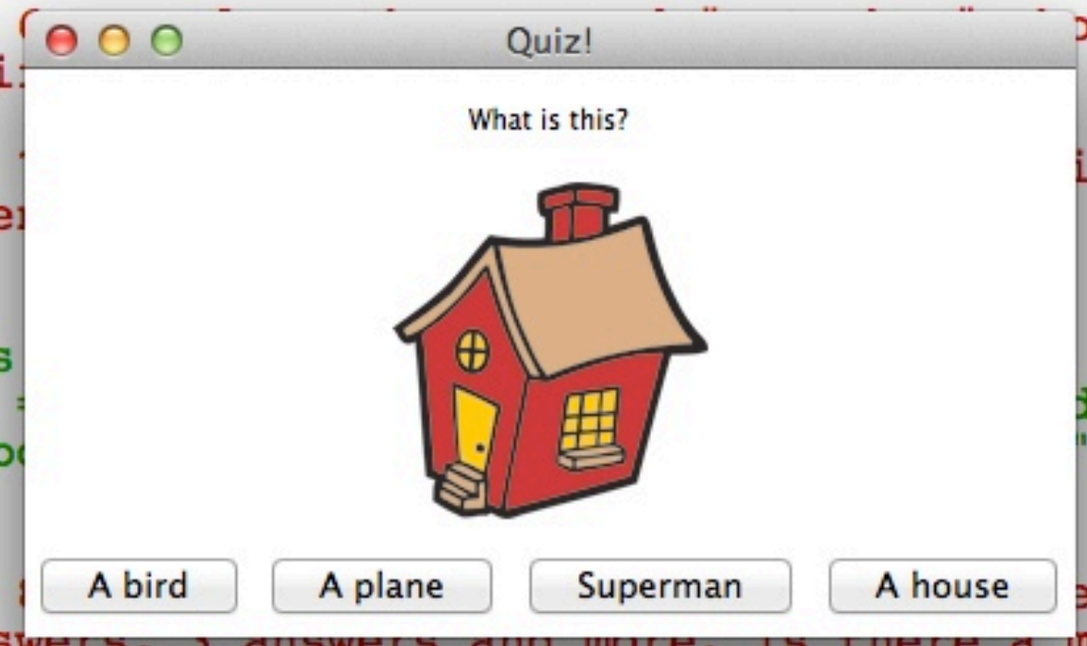
Suitable for KS3 onwards?
Can be inaccessible and offputting

EasyGUI

Text based input

Graphical output

```
# STEP 5 - Uncomment and run this code. Try to
# but DON'T try to fix it just yet

"""
number = eg.enterbox("Enter a number between

eg.msgbox("Your number doubled is " + str(numb
"""

# STEP
# See i

# STEP
# happe

"""
answers
choice
eg.msgb
"""

# STEP
# 2 answers, 3 answers and more. Is there a ma

# STEP 9 - What do you think will happen here


answers = ["A bird","A plane","Superman","A ho
choice = eg.buttonbox("What is this?","Quiz!"
if choice == "A house":
    eg.msgbox("Well done!","Quiz!")
else:
    eg.msgbox("No, you muppet. It was a house


# STEP 10 - Try changing this question. You w
# a suitable GIF image (jpg and png images wo
# same folder as your python program

# STEP 11 - Uncomment and run this line of co
```



A good compromise/bridge for students

Session Objectives

ALL students will be able to create a simple program using Python and EasyGUI

MOST students will be able to use selection (if statements) in a text-based programming language

SOME students will be able to use iteration (loops) in a text-based programming language

Working through easyPython.py exercise file with suitable extensions and programming challenges suitable for KS3

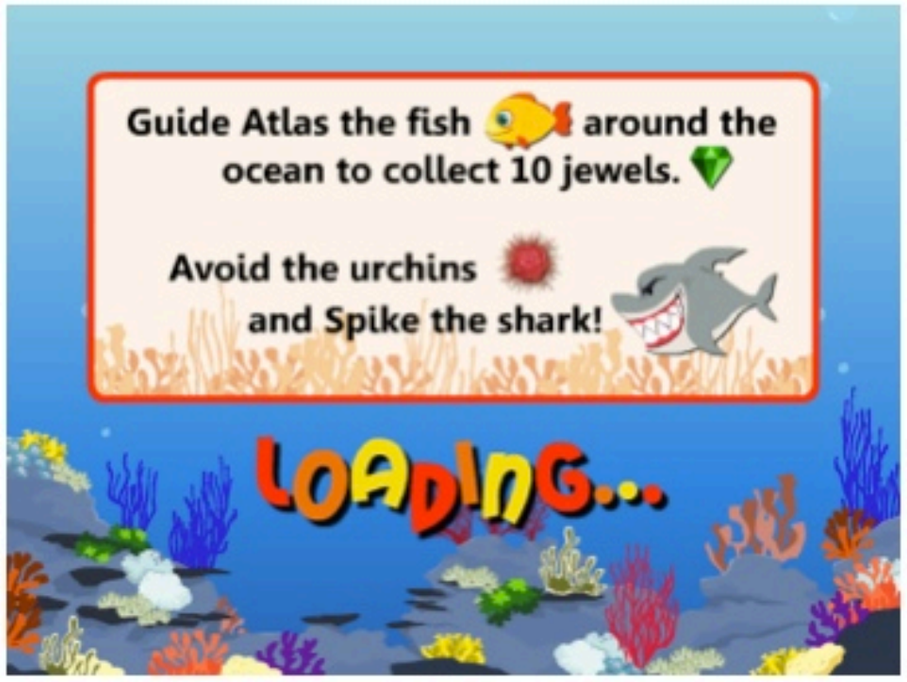
What happens next?

Creating the start-up screen

Your game also needs a sprite that forms the start-up screen and gives your user some instructions.

This sprite will be called **StartScreen**.

You can make this using the drawing tools in Scratch.



You can also make this using another drawing package. If you use another package, make sure you export the image as a JPEG before importing it into Scratch.

Boardworks Resources

Scratch Project

Variables in games

A variable changes throughout the execution of the program depending on different conditions.

For example, the score in a game changes often. It will start at 0 and then increase at different stages.

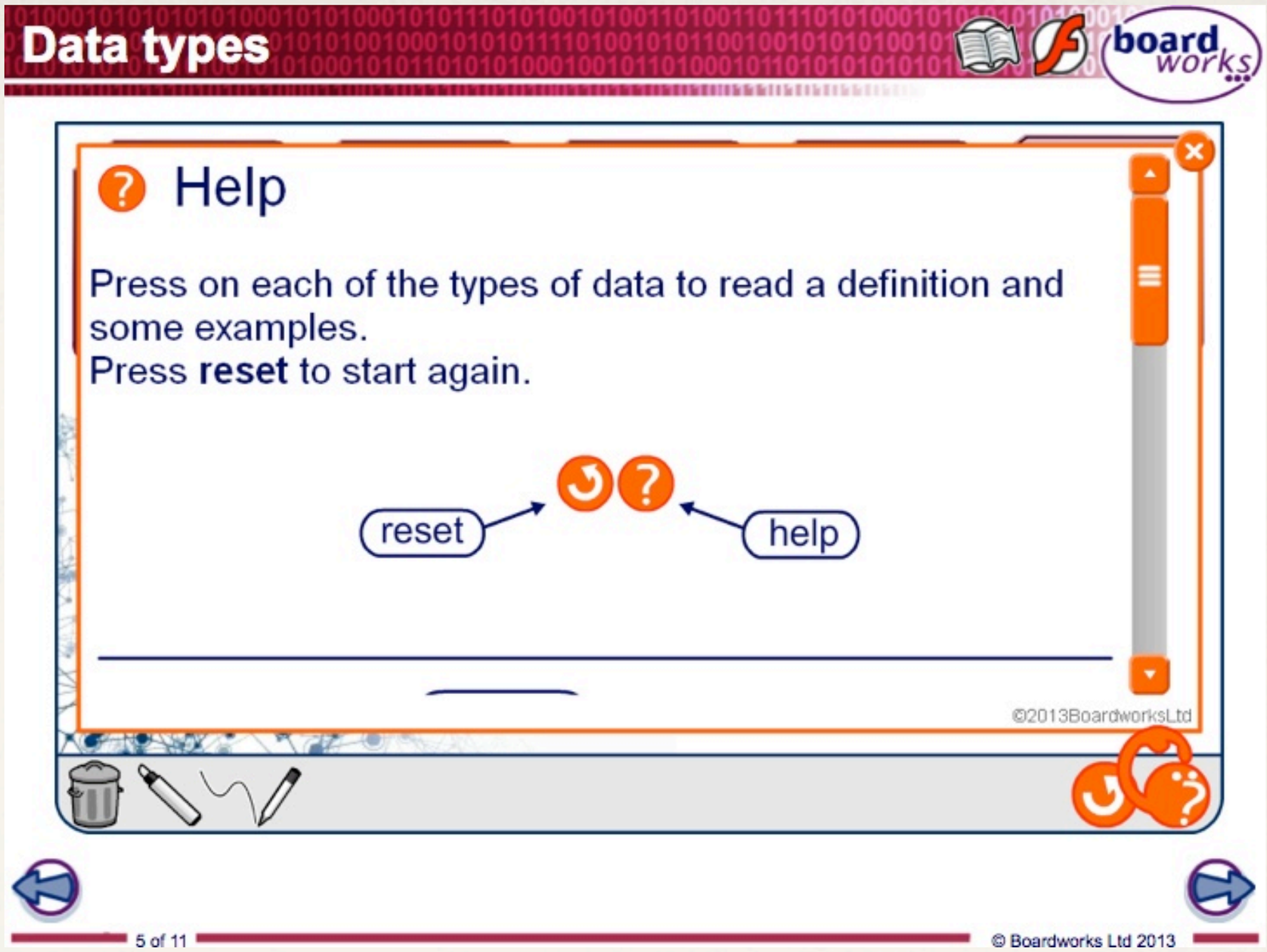
Here is a variable called 'score'. At the beginning of the program it holds the value 0.

```
score = 0
```

If you gain 10 points for every target you hit, the variable will change by 10.

Boardworks Resources

Programming Basics



Boardworks Resources

Handling Data in Algorithms

Network hardware: network cards

In order for a computer to function as part of a network, it has to have a special piece of equipment installed to give it this capability.

This is called a **Network Interface Card** (**NIC**). It has a port for a network cable to be connected.

NICs are still very common, but increasingly people are using wireless network cards or wireless USB adaptors.

Boardworks Resources

Networks

Shameless Plug

3rd July - CAS Hub
teessidehub.eventbrite.co.uk

6th & 7th July - Teesside Uni
goo.gl/GzmD8

10th July - Newcastle Uni
goo.gl/a9aXw

CAS Hub Meeting – teessidehub.eventbrite.co.uk
Teesside Uni details – goo.gl/GzmD8
Newcastle Uni details – goo.gl/a9aXw
Computing summer school – computingsummerschool.eventbrite.co.uk
XML/RSS feed –http://goo.gl/3SiL4

Shameless Plug

23rd July - 9th August
Computing Summer School
Newcastle

computingsummerschool.
eventbrite.co.uk

Plus much more to follow:
goo.gl/3SiL4

CAS Hub Meeting – teessidehub.eventbrite.co.uk
Teesside Uni details – goo.gl/GzmD8
Newcastle Uni details – goo.gl/a9aXw
Computing summer school – computingsummerschool.eventbrite.co.uk
XML/RSS feed –http://goo.gl/3SiL4