Langtree Computing

Langtree Computing

Ongoing development… – August 2018

How to use this document

For optimal use this document should be opened on Microsoft PowerPoint 2010 or later. It may work on other presentation software but it does contain a large number of carefully positioned pictures and hyperlinks, so you may find it hard to navigate. Please note if viewing as a pdf the "back" button won't work, instead use the small numbers underneath "Main Menu" to re-access the year group plans.

Hyperlinks have been set up throughout this document to help you navigate the Scheme of Work.

To begin, simply select "Main Menu" in the bottom left corner or click here.

A note for non-Langtree users…

Please feel free to make use of our Scheme of Work, however, we would ask that if you share our scheme of work you keep the Langtree Logo at the top – Credit where credit is due… (Our thanks to the TES Resources Community for many of the activities!)

Any feedback can be sent to one of Langtree School's Computing teachers – A full staff email list can be found at our school website, www.langtreeschool.com.

Main Menu

Langtree Computing

Key Stage 3

Year 7 | Year 8 | Year 9

Key Stage 4

Option Year 10 | Option Year 11

Main Menu

7 8 9 10 11

National Curriculum

- Algorithms
- Tests & Revision Time
- Programs
- Holiday
- Software & Data
- Off Curriculum
- Computers
- No lessons
- Communication and the Internet

Year 7

Langtree Computing

Wk1	Wk2	Wk3	Wk4	Wk5	Wk6	Wk7	Wk8	Wk9	Wk10
CATS	Core Computing Skills						October Holiday	Core Computing Skills	
Wk11	**Wk12**	**Wk13**	**Wk14**	**Wk15**	**Wk16**	**Wk17**	**Wk18**	**Wk19**	**Wk20**
Core Computing Skills				Assessment Core Computing	Final Week Fun	Christmas Holiday		Programs 1 (Scratch)	
Wk21	**Wk22**	**Wk23**	**Wk24**	**Wk25**	**Wk26**	**Wk27**	**Wk28**	**Wk29**	**Wk30**
Programs 1 (Scratch)				February Holiday	Hardware, Binary & Networks				
Wk31	**Wk32**	**Wk33**	**Wk34**	**Wk35**	**Wk36**	**Wk37**	**Wk38**	**Wk39**	**Wk40**
Assessment Hardware Binary Networks	Easter Holiday		Algorithms & Pseudocode				Assessment Algorithms	May Holiday	Programs 2 (Python)
Wk41	**Wk42**	**Wk43**	**Wk44**	**Wk45**					
Programs 2 (Python)				Creativity Week					

Main Menu

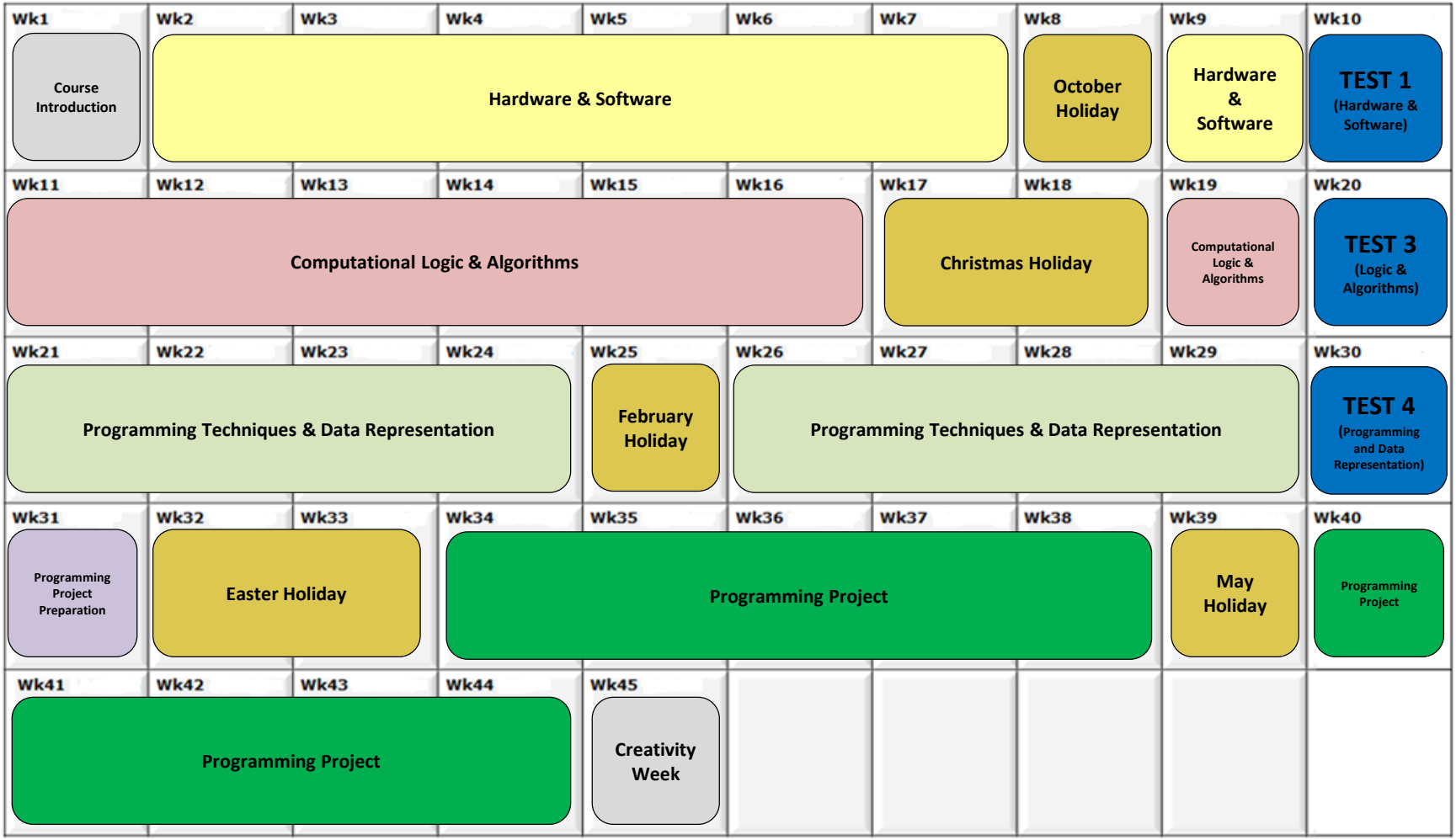
7 8 9 10 11

Year 8

Year 8

All pupils complete 4 out of 5 modules of the OCR Entry Level Certificate in Computer Science (R354)

Langtree Computing



Main Menu

7 8 9 10 11

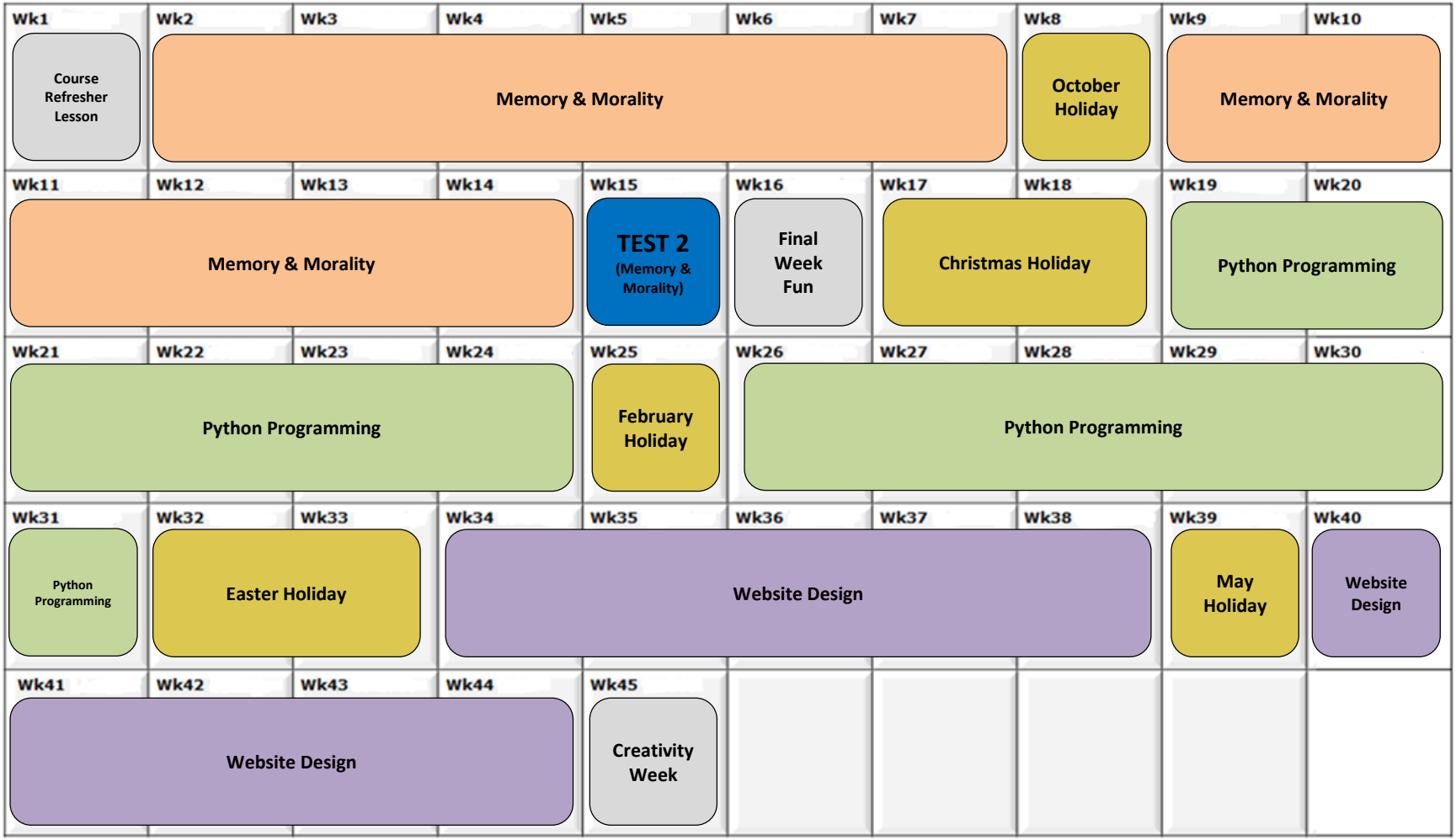
Year 7

Year 9

Year 9

In 2018/2019 the Memory and Morality unit will be replaced with 2 Scratch programming projects, as these pupils will have completed the unit in year 8.

Langtree Computing



Main Menu

7 8 9 10 11

Year 8

Year 10

Year 10 - Option

Following the J276 OCR GCSE Computer Science

Langtree Computing

Wk1	Wk2	Wk3	Wk4	Wk5	Wk6	Wk7	Wk8	Wk9	Wk10
Introduction to Course	1.1 System Architecture		1.2 Memory	1.3 Storage	1.4 Networks & Network Topologies		October Holiday	1.6 Systems Security	
Wk11	**Wk12**	**Wk13**	**Wk14**	**Wk15**	**Wk16**	**Wk17**	**Wk18**	**Wk19**	**Wk20**
1.6 Systems Security	1.7 Software	1.8 Ethical, Social and environmental concerns		REVISION – Theory so far		Christmas Holiday		2.1 Algorithms	
Wk21	**Wk22**	**Wk23**	**Wk24**	**Wk25**	**Wk26**	**Wk27**	**Wk28**	**Wk29**	**Wk30**
2.1 Algorithms				February Holiday	2.2 Programming				
Wk31	**Wk32**	**Wk33**	**Wk34**	**Wk35**	**Wk36**	**Wk37**	**Wk38**	**Wk39**	**Wk40**
2.2 Programming	Easter Holiday		2.2 Programming					May Holiday	2.2 Programming + Prep
Wk41	**Wk42**	**Wk43**	**Wk44**	**Wk45**					
2.2 Programming + Assignment Preparation				Creativity Week					

Main Menu

7 8 9 10 11

Year 9

Year 11

Year 11 - Option

Following the J276 OCR GCSE Computer Science

Langtree Computing

Wk1	Wk2	Wk3	Wk4	Wk5	Wk6	Wk7	Wk8	Wk9	Wk10
2.2 Programming + Assignment Preparation/Introduction			Programming Assignment				October Holiday	Programming Assignment	
Wk11	**Wk12**	**Wk13**	**Wk14**	**Wk15**	**Wk16**	**Wk17**	**Wk18**	**Wk19**	**Wk20**
Programming Assignment					Controlled Assignment Feedback	Christmas Holiday		REVISION – Theory from year 10	
Wk21	**Wk22**	**Wk23**	**Wk24**	**Wk25**	**Wk26**	**Wk27**	**Wk28**	**Wk29**	**Wk30**
2.4 Logic		2.5 Translators and facilities of languages		February Holiday	2.6 Data Representation			Exam Revision	
Wk31	**Wk32**	**Wk33**	**Wk34**	**Wk35**	**Wk36**	**Wk37**	**Wk38**	**Wk39**	**Wk40**
Exam Revision	Easter Holiday		Exam Revision/Exam					May Holiday	Students have left!
Wk41	**Wk42**	**Wk43**	**Wk44**	**Wk45**					
Students have left!									

Main Menu

7 8 9 10 11

Year 10

Core Computing Skills Year 7 (12 Lessons)

Year 7's first term of Computing at Langtree is designed to give them the skills needed to use computers for work across the curriculum.

Lesson 1	Assessment of ability
Lesson 2	Navigate Start Menu Open and write letter on Word
Lesson 3	Access and use email Access Google Docs and the VLE
Lesson 4	Understand & Navigate Through Folder Structure (sensible filenames) Saving frequently, backing up and deleting unnecessary files
Lesson 5	Powerpoint presentation (digital safety)
Lesson 6	Powerpoint presentation (digital safety)
Lesson 7	Effective searching on the internet
Lesson 8	Create logo with fireworks
Lesson 9	Create logo with fireworks
Lesson 10	Open spreadsheet select cells, format basically, SUM +, x
Lesson 11	Open spreadsheet select cells, format basically, SUM +, x
Lesson 12	Touch Typing

Year 7 Core Computing Lesson 1 – Assessment of Ability

Langtree Computing

LESSON SUMMARY

This lesson will be used for a short assessment of the students' ability.

OBJECTIVES

- Assess the students' ability after leaving Primary School.

RESOURCES

- Baseline Assessment Page

NOTES

- Pupils will need logins provided by their teacher.
- We are using the ingots 2014 baseline for all students at the start of the year.

Year 7 Core Computing
Lesson 2 – The Bare Basics

LESSON SUMMARY

By the end of this lesson all students will be able to open and use the basic features of Microsoft Word, a skill that is necessary for almost all subjects!

OBJECTIVES

- Understand how to navigate the start menu and open key programs
- Know how to change font, font size, font colour, justification and spacing in Microsoft Word.

RESOURCES

- Task List – Lesson 2
- Hey's Caravan Task Picture
- Microsoft Word Intro Video

NOTES

- No notes (*yet!*).

Year 7 Core Computing
Lesson 3 – Langtree VLE

LESSON SUMMARY

In this lesson the students learn about how to access and use their school email and VLE account.

OBJECTIVES

- Understand how to log into the school website.
- Know how to open an email in Gmail.
- Know how to compose an email in Gmail.
- Know how to create and share a collaborative document using Google Docs.

RESOURCES

- Task List (Lesson 3)
- Gmail Start Guide (video)
- Google Docs Introduction
- www.langtreeschool.com

NOTES

- **ALL** students should write their username and a password hint in their planners.

Year 7 Core Computing
Lesson 4 – Folder Structure

Langtree Computing

LESSON SUMMARY

In this lesson the students should get an overview of how files and folders are best organised on a computer.

OBJECTIVES

- Understand the importance of sensible file names
- Know the difference between a "drive", "folder" and "file".
- Be competent at navigating and searching with Windows Explorer.

RESOURCES

- Task List – Lesson 4
- "My Folder Game" Helpsheet
- Folder Structure (video)

NOTES

- No notes *(yet!)*

Year 7 Core Computing Lessons 5 & 6 – Digital Safety

LESSON SUMMARY

In these lessons the students will be given an overview of the key issues around digital safety.

OBJECTIVES

- Understand the risks associated with social networking sites
- Know the meaning of "cyber-bullying" and how to protect yourself from it
- Understand the importance of a good password
- Understand the risks associated with email attachments and downloads.
- Be able to suggest sensible ways of protecting a computer from Malicious software.

RESOURCES

- CEOP "ThinkuKnow" Videos
- Thinkuknow & eSafety Lesson/Worksheet Pack

NOTES

- *The lesson pack contains 6 lesson's worth of stuff… select what you want to use!*

Year 7 Core Computing Lessons
7 – Effective Searching

LESSON SUMMARY

In this lesson the students will consider some of the best ways to search the internet for useful information.

OBJECTIVES

- Know how to use advanced search features to filter results in internet search engines.

RESOURCES

- Internet Searching Activity
- Boolean Searching – Helpsheet

NOTES

- *No notes yet!*

Main Menu

7 8 9 10 11

Back

Year 7 Core Computing
Lessons 8 & 9 – Logo Design

Langtree Computing

LESSON SUMMARY

In these lessons the pupils will explore what makes a graphic design look professional and how to use the basic features of the Adobe Fireworks software.

OBJECTIVES

- Create a logo on Adobe Fireworks
- Apply logo to a different professional looking documents using a range of software.

RESOURCES

- Lesson Presentation
- Identifying Logos Worksheet
- Fireworks Guide
- Peer Evaluation Sheet

NOTES

- *Pinched from Westfield… adapt as needed!*

Main Menu

7 8 9 10 11

Back

Year 7 Core Computing Lessons
10 & 11 - Spreadsheets

LESSON SUMMARY

In these lessons the students are given an overview of the basic features and uses of spreadsheet software.

OBJECTIVES

- Be able to select and enter data into cells
- Know how to resize and format cell fonts
- Know how to do basic arithmetic operations with a spreadsheet
- Know how to use the SUM function on a spreadsheet

RESOURCES

- Teddy Tots Task

NOTES

- *Not all students will take two lessons over this. Extensions could include making a drop down list, using multiple sheets and conditional formatting.*

Main Menu

7 8 9 10 11

Back

Year 7 Core Computing
Lesson 12 – Touch Typing

LESSON SUMMARY

In this lesson the students should learn about how to best position their hands and fingers to type quickly.

OBJECTIVES

- Understand why quick typing is a useful skill.
- Know that using two hands to type is better than one!
- Have an idea of the best way to position hands for touch typing.

RESOURCES

- Website that teaches touch typing
- 10fastfingers.com – Assesses speed
- Youtube Video – Competition
- Youtube Video – Typist at work

NOTES

- *10 fast fingers seems to freeze when 10+ students all go onto it at the same time.*
- *There are now two exercises also on the school shared drive.*

CATS

- Cognitive Ability Tests
- Students complete online CAT tests
- These are used to help set targets across all subjects
- Usually run and prepared by Examinations Officer
- Supervised by Computing Teacher

Programs 1 – Year 7
6 Lessons

Lesson 1	Code.org – Hour of Code
Lesson 2	Introduction to Scratch (Dancing Cat)
Lesson 3	Broadcasts
Lesson 4	Snail Trail
Lesson 5	Shark Attack
Lesson 6	Free Project

Year 7 Programs 1 Lesson 1 – Code.org - Hour of Code

Langtree Computing

LESSON SUMMARY

This lesson introduces the students to some of the basic concepts of programming through the interactive "game" style tasks presented by the Code.org website.

OBJECTIVES

- Understand that programs are a sequence of instructions.
- Begin to realise the purpose of repeating instructions (and creating algorithms) to give a wanted result.

RESOURCES

- http://learn.code.org/s/1/level/2
- Extension Task – "The Artist"

NOTES

- *There is a video that introduces the tasks – Students may use headphones for this or the teacher may wish to play it on the board.*

Year 7 Programs 1 Lesson 2 – Introduction to Scratch (Dancing Cat)

Langtree Computing

LESSON SUMMARY

In this lesson the students will be introduced to the educational programming software, Scratch.

OBJECTIVES

- All students should understand how to find and log into the Scratch Website
- All students should be able to make a sprite move in scratch by using a combination of motion and event commands.

RESOURCES

- Opening & saving with scratch website helpsheet
- Scratch Website
- Getting started with Scratch (pdf guide)
- Dancing Cat Worksheet
- Scratch Science
- Scratch SoW for Primary

NOTES

- "Scratch Science" and "Scratch SoW for Primary" are not needed for this lesson, I just added them as they have some good lessons and resources!

Year 7 Programs 1 Lesson 3 – Broadcasts

LESSON SUMMARY

In this lesson students will learn how to use "broadcast" in Scratch to make one sprite respond to something that happens with another.

OBJECTIVES

- Know how to use the "broadcast" command along with "when … received" to sequence events in Scratch.

RESOURCES

- Scratch Lesson 2 Worksheet

NOTES

- *No notes yet!*

Year 7 Programs 1 Lesson 4 – Snail Trail

Langtree Computing

LESSON SUMMARY

In this lesson the students use what they've learnt with a few new skills to design their first game in Scratch.

OBJECTIVES

- Understand how to use "sensing" to detect key presses
- Begin to understand the logical structure "IF A, do this, IF B, do that…" etc

RESOURCES

- Snail Trail Worksheet

NOTES

- *No notes yet!*

Main Menu

7 8 9 10 11

Back

Year 7 Programs 1 Lesson 5 – Shark Attack

LESSON SUMMARY

In this lesson the students will make a more complex game with multiple moving sprites that interact with one another.

OBJECTIVES

- Practise and consolidate Scratch programming skills.
- Understand how to use "touching" sensor commands in Scratch.
- Recognise the importance of annotating code and be able to do this with Scratch.

RESOURCES

- Shark Attack Worksheet

NOTES

- *No notes yet!*

Year 7 Programs 1 Lesson 6 – Free Project

Langtree Computing

LESSON SUMMARY

In this lesson, if the student has finished their "Shark Attack" project, they can use all the skills they have gained in the previous lessons to create their own animation or game.

OBJECTIVES

- Consolidate, practice and demonstrate skills learnt in the previous five lessons.

RESOURCES

- *No Resources yet!*

NOTES

- *No notes yet!*

Hardware, Binary & Networks – Year 7 – 6 Lessons

Lesson 1	Hardware
Lesson 2	Hardware
Lesson 3	Binary
Lesson 4	Binary
Lesson 5	Networks
Lesson 6	Networks

Year 7 Hardware, Binary and Networks Lesson 1 – Hardware

Langtree Computing

LESSON SUMMARY

In this lesson, the students will learn about the names and purposes of each of the main components of a personal computer.

OBJECTIVES

- Identify the components of a computer e.g. input, output and storage devices
- Identify the basic function of the common internal components of a computer e.g. motherboard, CPU, RAM, ROM, graphics cards, sound cards, hard disks
- Identify the basic functions of common peripherals e.g. camera, keyboard, microphones, monitor, mouse, scanner, speakers, printer

RESOURCES

- Hardware presentation task
- Hardware Worksheet

NOTES

- **Currently a clone of the year 9 Hardware Work!**

Year 7 Hardware, Binary and Networks Lesson 2 – Hardware

LESSON SUMMARY

In this lesson, the students will learn about the names and purposes of each of the main components of a personal computer.

OBJECTIVES

- Identify the components of a computer e.g. input, output and storage devices
- Identify the basic function of the common internal components of a computer e.g. motherboard, CPU, RAM, ROM, graphics cards, sound cards, hard disks
- Identify the basic functions of common peripherals e.g. camera, keyboard, microphones, monitor, mouse, scanner, speakers, printer

RESOURCES

- Hardware presentation task
- Hardware Worksheet

NOTES

- **Currently a clone of the year 9 Hardware Work!**

Year 7 Hardware, Binary and Networks Lesson 3 – Binary

LESSON SUMMARY

In this lesson the students are introduced to binary numbers – what they are, why they're important in computing and how they are used in images and sound.

OBJECTIVES

- Be able to convert between binary and denary numbers from 0 to 15.
- Understand why binary numbers are used in computing.

RESOURCES

- Binary Numbers in computing - Presentation
- Binary Table Worksheet
- Binary Code Worksheet
- CS Unplugged Binary Numbers Lesson Plan

NOTES

- **Currently a clone of the year 9 Binary Work!**

Year 7 Hardware, Binary and Networks Lesson 4 – Binary

LESSON SUMMARY

In this lesson the students are introduced to binary numbers – what they are, why they're important in computing and how they are used in images and sound.

OBJECTIVES

- Be able to convert between binary and denary numbers from 0 to 15.
- Understand why binary numbers are used in computing.

RESOURCES

- Binary Numbers in computing - Presentation
- Binary Table Worksheet
- Binary Code Worksheet
- CS Unplugged Binary Numbers Lesson Plan

NOTES

- **Currently a clone of the year 9 Binary Work!**

Year 7 Hardware, Binary and Networks Lesson 5 – Networks

Langtree Computing

LESSON SUMMARY

In this lesson the students complete an interactive task that introduces them to the structure of a basic workplace network.

OBJECTIVES

- Learn the names of some pieces of hardware used for managing networks.
- Have an understanding of how machines in a network are connected.

RESOURCES

- Introduction to Networks (interactive)

NOTES

- *No notes yet!*

Year 7 Hardware, Binary and Networks Lesson 6 – Networks

LESSON SUMMARY

In this lesson the students study the basics of how data is retrieved by an end user through use of a browser.

OBJECTIVES

- Have an end-to-end understanding of what happens when a user requests a web page in a browser, including:
- Browser and server exchange messages over the network
- What is in the messages [http request, and HTML]
- The structure of a web page - HTML, style sheets, hyperlinking to resources
- What the server does [fetch the file and send it back]
- What the browser does [interpret the file, fetch others, and display the lot]

RESOURCES

- Browser Story Task
- Internet key word matching exercise

NOTES

- *The browser story task assumes that the teacher has taught the students about the objectives listed on the left.*

Algorithms & Pseudocode – Year 7 – 6 Lessons

Lesson 1	Sandwich Bot
Lesson 2	Flow Charts
Lesson 3	Flow Charts
Lesson 4	Lightest & Heaviest – Sorting Algorithms
Lesson 5	Pseudocode
Lesson 6	Pseudocode

http://csunplugged.org/

Year 7 Algorithms & Pseudocode
Lesson 1 – Sandwich Bot

Langtree Computing

LESSON SUMMARY

This lesson introduces the concept of algorithms (sequences of instructions) and highlights the importance of well thought through instructions within computer programs.

OBJECTIVES

- To write an accurate algorithm (sequence of instructions) so that sandwich bot will make a jam sandwich.
- To know that all computer programs are sequences of instructions.

RESOURCES

- Lesson Notes
- Printable Resources
- Sandwich Bot Example Video

NOTES

- *No notes yet!*

Year 7 Algorithms & Pseudocode
Lesson 2 & 3 – Flow Charts

LESSON SUMMARY

In this lesson the students will look at how flow charts are used to present an algorithm.

OBJECTIVES

- Be able to follow the instructions of simple flow charts
- Recognise that flow charts contain inputs, outputs, processes and decisions.
- Know how to create simple flow charts for day to day activities.

RESOURCES

- Presentation – Algorithms and Flow Charts
- Interactive Flow Chart Tasks
- Flowcharts – Worksheet
- Flowcharts – Breakdown Challenge

NOTES

- *No notes yet!*

Year 7 Algorithms & Pseudocode Lesson 4 – Lightest and Heaviest (sorting algorithms)

LESSON SUMMARY

Computers are often used to put lists into some sort of order and in this lesson pupils are encouraged to start to explore algorithms that complete this task.

OBJECTIVES

- Recognise an algorithm as a repeatable set of instructions.
- Begin to understand that computers often do many repetitions of a simple task to achieve a complex result.

RESOURCES

- CS unplugged Sorting Algorithms page
- CS unplugged Sorting Algorithms Lesson plan

NOTES

- *No notes yet!*

Year 7 Algorithms & Pseudocode
Lesson 5 & 6 – Pseudocode

Langtree Computing

LESSON SUMMARY

In these lessons the students will be introduced to the idea of pseudocode, as well as getting further practise sequencing instructions.

OBJECTIVES

- Understand that "pseudocode" is a half way point between writing the logical steps and writing actual program code.
- Develop ability to sequence tasks into logical steps.

RESOURCES

- Khan Academy Pseudocode Demo
- Pseudocode Task 1
- Pseudocode Task 2

NOTES

- *No notes yet!*

Programs 2 – Year 7
6 Lessons

Lesson 1	Python Booklet – Opening & Saving a Python File, Turtle Introduction
Lesson 2	More Turtle – Trying out extra commands
Lesson 3	Loops in Python
Lesson 4	Sabotage! – Learning about Syntax Errors
Lesson 5	print(), input() and variables
Lesson 6	Free Project

Year 7 Programs 2 Lesson 1 – Opening Python & Turtle()

Langtree Computing

LESSON SUMMARY

In this lesson the students will learn how to open the Python GUI and will be introduced to text based programming through the "Turtle()" pre-installed package.

OBJECTIVES

- Know how to open IDLE, the python GUI.
- Be able to write a simple program to draw a line on Python using Turtle() graphics.

RESOURCES

- Python Lessons Booklet

NOTES

- *No notes yet!*

Year 7 Programs 2 Lesson 2 – More Turtle

Langtree Computing

LESSON SUMMARY

In this lesson the students will reinforce and extend ideas from the lesson before, getting used to the python interface and Turtle() package.

OBJECTIVES

- Understand and be able to describe the purpose of at least 8 different commands within Turtle().

RESOURCES

- Python Lessons Booklet

NOTES

- *No notes yet!*

Main Menu

7 8 9 10 11

Back

Year 7 Programs 2 Lesson 3 – Loops in Python

Langtree Computing

LESSON SUMMARY

In this lesson the students will be re-introduced to the concept of loops in programming. They will then use "for" loops within Python to create more interesting Turtle() graphics.

OBJECTIVES

- Understand that a loop allows a sequence of instructions to be repeated.
- Be able to write a "for" loop for a given number of iterations within Python.

RESOURCES

- Python Lessons Booklet

NOTES

- *No notes yet!*

Main Menu

7 8 9 10 11

Back

Year 7 Programs 2 Lesson 4 – Syntax Errors

Langtree Computing

LESSON SUMMARY

In this lesson the students will learn about how sensitive programs are to "syntax errors", errors in typing, capitalisation and spacing.

OBJECTIVES

- Understand that computer programs have to be typed and structed perfectly.
- Learn the systems that can be used to quickly identify and fix syntax errors within programs.

RESOURCES

- Python Lessons Booklet

NOTES

- *No notes yet!*

Year 7 Programs 2 Lesson 5 – print(), input() and variables

Langtree Computing

LESSON SUMMARY

In this lesson the students will learn a little about basic text input and output within python.

OBJECTIVES

- Understand the use of the “print()” and “input()” functions within Python.
- Know how to store information entered with “input()” using a variable.

RESOURCES

- Python Lessons Booklet

NOTES

- *No notes yet!*

Year 7 Programs 2 Lesson 6 – Free Project

Langtree Computing

LESSON SUMMARY

In this lesson, if the student has finished their other python lesson exercises, they can use all the skills they have gained in the previous lessons to create their own program.

OBJECTIVES

- Consolidate, practice and demonstrate skills learnt in the previous five lessons.

RESOURCES

- Python Lessons Booklet

NOTES

- *No notes yet!*

Assessment – Year 7 Core Computing

- The students will be given an assessment that tests them on the content studied in the above unit.

Assessment – Year 7
Hardware, Binary & Networks

- The students will be given an assessment that tests them on the content studied in the above unit.

Assessment – Year 7 Programs

- The students will be given an assessment that tests them on the content studied in the above unit.

Main Menu

7 8 9 10 11

Back

Year 9 – Course Introduction

- Year 9 complete the OCR Entry Level Computing Science course.

Main Menu

7 8 9 10 11

Back

Entry Level Computer Science
Computational Logic & Algorithms
6 Lessons

Lesson 1	Binary Numbers
Lesson 2	Binary Addition
Lesson 3	Logic Gates & Boolean Logic
Lesson 4	Computational Thinking & Flowcharts 1
Lesson 5	Computational Thinking & Flowcharts 2
Lesson 6	Boolean & Arithmetic Operators

Entry Level Computer Science
Binary Numbers

LESSON SUMMARY

In this lessons students should revisit the idea of binary numbers that has been introduced in year 7.

OBJECTIVES

- understand how numbers are represented in binary
- be able to convert between binary and decimal from 0 to 15
- understand the purpose of data in computer systems being represented in binary form

RESOURCES

- James May On Binary
- Binary Numbers in computing - Presentation
- Binary Table Worksheet
- Binary Code Worksheet
- CS Unplugged Binary Numbers Lesson Plan

NOTES

- *Links closely to next lesson (binary addition)*
- *Also links to lessons 2 and 3 in next session – representing data in binary*

Main Menu

7 8 9 10 11

Back

Entry Level Computer Science
Binary Addition

LESSON SUMMARY

In this lesson pupils will be taught explicit methods for adding binary numbers.

OBJECTIVES

- be able to carry out simple operations on binary numbers using binary addition (4 bit)

RESOURCES

- Khan Academy Video
- Binary Addition Basic Worksheet
- Binary Addition Short Textbook Extract
- Binary Addition MEP Textbook – Multiple Exercises, Guides & Activities

NOTES

- *Pupils who get this quickly could be extended with larger binary values, hexadecimal numbers or other arithmetic problems.*
- *Links to Logic Gates (next lesson).*

Main Menu

7 8 9 10 11

Back

Entry Level Computer Science
Logic Gates & Boolean Logic

LESSON SUMMARY

In this lesson pupils learn the basics about the AND, OR and NOT logic gates.

OBJECTIVES

- understand simple Boolean logic and some of its uses in programming
- be able to create the basic truth tables for the output of the logic gates: → AND → OR → NOT

RESOURCES

- Boolean Logic Explanation
- Basic Truth Table Worksheet
- Logic Gate Circuit exercises
- Cambridge Textbook Extract
- Interactive Logic Gate Program
- Logic Gate Challenge Diagram

NOTES

- *No notes yet!*

Entry Level Computer Science
Computational Thinking & Flowcharts 1

Langtree Computing

LESSON SUMMARY

In this pair of lessons, pupils will consider how everyday and computer specific problems can be broken down into small steps and structured algorithms.

OBJECTIVES

- identify the success criteria of a problem
- create a basic plan to solve a problem
- sequence instructions in a logical way
- identify potential difficulties
- identify ways to check that a solution works
- produce algorithms using flow charts
- use and be familiar with the flow chart shapes for:
  - o Start / Stop
  - o Process
  - o Input / Output
  - o Decision
  - o Flow Lines

RESOURCES

- Presentation – Algorithms and Flow Charts
- Interactive Flow Chart Tasks
- Flowcharts – Worksheet
- Flowcharts – Breakdown Challenge
- Flowcharts – presentation in more detail

NOTES

- *No notes yet!*

Entry Level Computer Science
Computational Thinking & Flowcharts 2

LESSON SUMMARY

In this pair of lessons, pupils will consider how everyday and computer specific problems can be broken down into small steps and structured algorithms.

OBJECTIVES

- identify the success criteria of a problem
- create a basic plan to solve a problem
- sequence instructions in a logical way
- identify potential difficulties
- identify ways to check that a solution works
- produce algorithms using flow charts
- use and be familiar with the flow chart shapes for:
  - o Start / Stop
  - o Process
  - o Input / Output
  - o Decision
  - o Flow Lines

RESOURCES

- Presentation – Algorithms and Flow Charts
- Interactive Flow Chart Tasks
- Flowcharts – Worksheet
- Flowcharts – Breakdown Challenge
- Flowcharts – presentation in more detail

NOTES

- *No notes yet!*

Entry Level Computer Science
Boolean and Arithmetic Operators

Langtree Computing

LESSON SUMMARY

In this lesson pupils will learn the way that most programming languages write Boolean and Arithmetic operators.

OBJECTIVES

- understand and evaluate the following Boolean operators:
  - o equal to (a == b)
  - o not equal to (a != b)
  - o less than (a < b)
  - o greater than (a > b)
- understand and be able to use the following mathematical symbols:
  - o + (add)
  - o – (subtract)
  - o * (multiply)
  - o / (divide)
- use common arithmetic operators within a program
- use common Boolean logic operators within a program

RESOURCES

- Code.org Lesson Plans (Look at lessons 8 & 9. There are some excellent worksheets with these lessons)
- BBC Bitesize, Boolean Logic

NOTES

- *No notes yet!*

Langtree School
PUTTING LEARNING FIRST

Entry Level Computer Science
Programming Techniques and Data Representation
7 Lessons

Langtree Computing

Lesson 1	Units of Memory
Lesson 2	Representing Data in Binary
Lesson 3	Representing Data in Binary & Compression of Data
Lesson 4	Input, Output and Storage of Data
Lesson 5	Variables
Lesson 6	Sequence, Iteration and Selection
Lesson 7	Comments in Programming

Entry Level Computer Science
Units of Memory

LESSON SUMMARY

In this lesson pupils will learn about the units that are used to measure quantities of computer memory.

OBJECTIVES

- understand that computer memory or storage are measured using different units:
  - o bit
  - o nibble
  - o byte
  - o kilobyte
  - o megabyte
  - o gigabyte

RESOURCES

- Code.org Lesson Link
- Code.org Worksheet
- Poster on units of storage

NOTES

- *No notes yet!*

Entry Level Computer Science
Representing Data in Binary

LESSON SUMMARY

This is the first of two lessons which looks at how everyday data (text, sounds and pictures) can be stored as a binary number.

OBJECTIVES

- understand how data can be represented digitally, in the form of binary digits for:
  - o text
  - o sounds
  - o pictures

RESOURCES

- Binary Numbers in computing – Presentation
- BBC Bitesize – Encoding Images
- BBC Bitesize – Encoding Audio & Video
- Code.org Binary Images (with worksheets)

NOTES

- *Different quality versions of an audio file are on the school shared drive (and may later be added to the resources)*
- *Some school licenced textbooks have good worksheet activities*

Entry Level Computer Science
Representing Data in Binary & Compression of Data

LESSON SUMMARY

This is the second of two lessons which looks at how everyday data (text, sounds and pictures) can be stored as a binary number. This lesson also introduces data compression.

OBJECTIVES

- understand how data can be represented digitally, in the form of binary digits for:
  - text
  - sounds
  - pictures
- understand the purpose of data compression in terms of:
  - transmission of data
  - storage

RESOURCES

- Binary Numbers in computing – Presentation
- BBC Bitesize – Encoding Images
- BBC Bitesize – Encoding Audio & Video
- Code.org Binary Images (with worksheets)
- CSunplugged on text compression (part 1)
- CSunplugged on text compression (part 2)

NOTES

- *Different quality versions of an audio file are on the school shared drive (and may later be added to the resources)*
- *Some school licenced textbooks have good worksheet activities*
- *The BBC Bitesize site has good videos on compression in the later sections*

Entry Level Computer Science
Input, Output and Storage of Data

LESSON SUMMARY

In this lesson pupils are introduced to variables and the basic data types used within programs.

OBJECTIVES

- explain and show how input may be captured and assigned to a variable for use/storage within a program.
- explain and show how to output text or movement on screen
- be able to use a range of data types including:
  - integers
  - real numbers
  - text
  - Boolean
  - lists / arrays (one dimensional) or equivalent

RESOURCES

- BBC Bitesize – data types and structures
- CanYouCompute brief lesson plan (with worksheet)

NOTES

- *It is recommended to teach this through teaching programming as opposed to the theory in isolation.*
- *This ties into, and in many ways has the same objective, as the following lesson*

Entry Level Computer Science
Variables

LESSON SUMMARY

In this lesson pupils will develop their understanding of variables within programs.

OBJECTIVES

- explain what a variable is used for (i.e. storing data within a program)
- perform basic mathematical or logical calculations on variables

RESOURCES

- BBC Bitesize – data types and structures
- CanYouCompute brief lesson plan (with worksheet)

NOTES

- *It is recommended to teach this through teaching programming as opposed to the theory in isolation.*
- *This ties into, and in many ways has the same objective, as the previous lesson*

Entry Level Computer Science
Sequence, Iteration and Selection

LESSON SUMMARY

sum

OBJECTIVES

- understand that instructions are executed in the sequence they are written
- write programs with instructions in the correct order
- be able to identify errors in the order of a sequenced set of steps
- explain and identify how programs can be made to execute code based on a choice (true or false) e.g. IF statements
- understand what is meant by a loop
- use a loop in a program to execute statements multiple times (WHILE loop and FOR loop)

RESOURCES

- *re*

NOTES

- *These objectives cannot all be fully covered in one lesson. This lesson should be focussed on identifying these programming constructs and discussing when they are used. Actually using them in depth will be taught as part of the programming project.*

Entry Level Computer Science
Comments in Programming

LESSON SUMMARY

In this lesson pupils will explore why and when to use comments in code.

OBJECTIVES

- explain why comments in code are useful
- show examples of commenting in code

RESOURCES

- Why Comment? (blog post)
- BBC Bitesize Comments page
- Understanding Code Challenge (exercises could be used as examples of poor code)

NOTES

- *A good exercise could be showing pupils ambiguous code and then asking "what's the problem with this code?"*
- *As an exercise, ask pupils to write well commented code, explain good practice and then get them to give feedback to one another.*

Entry Level Computer Science
Hardware and Software
5 Lessons

Lesson 1	Computer Hardware
Lesson 2	Input, Output, Storage and Processing
Lesson 3	System Software
Lesson 4	The Operating System
Lesson 5	Application Software

Entry Level Computer Science
Computer Hardware

LESSON SUMMARY

In this lesson pupils will revisit hardware and learn the parts that make up a computer.

OBJECTIVES

- identify the basic function of the common internal components of a computer:
  - motherboard - CPU - RAM
  - BIOS - hard disks
- identify the basic functions of common peripherals:
  - camera - keyboard - microphones
  - monitor - mouse - scanner
  - headphones - speakers - printer

RESOURCES

- Hardware presentation task
- Hardware Worksheet
- Textbook, Worksheets and Activities

NOTES

- *Storage devices are visited in more detail in the first lesson of the next section – It will be useful to make students aware that they will be learning more about hard drives and RAM.*

Entry Level Computer Science
Input, Output, Storage and Processing

LESSON SUMMARY

In this lesson pupils will be reminded of how we can classify many hardware devices as input, output or storage.

OBJECTIVES

- classify the components of a computer with respect to:
  - o input
  - o output
  - o storage

RESOURCES

- Hardware presentation task
- Hardware Worksheet
- Input and Output devices Worksheet

NOTES

- *Storage devices are visited in more detail in the first lesson of the next section – It will be useful to make students aware that they will be learning more about hard drives and RAM.*

Entry Level Computer Science
System Software

LESSON SUMMARY

In this lesson pupils will be introduced to some of the system utility software programs that are used on most computers.

OBJECTIVES

- state the purpose of different system utilities:
  - computer security (antivirus, anti-malware, anti-spyware and firewalls),
  - disk management (formatting, file transfer, and defragmentation), and back up
  - system maintenance (system information and diagnosis, system clean-up tools, automatic updating)

RESOURCES

- Data Security Worksheet
- Video on types of software for revision
- Good Video on Utility Software (*requires paid subscription for full access – first part still useful*)
- Worksheet on Utility Software

NOTES

- *No notes yet!*

Entry Level Computer Science
The Operating System

LESSON SUMMARY

In this lesson, pupils learn about Operating systems (such as Windows, Linux and IOS). They learn about their purpose and some key features.

OBJECTIVES

- identify a range of operating systems, including Open Source and Proprietary
- state why operating systems are needed
- state the basic functions of an operating system:
  - management of software
  - management of hardware (through device drivers)
  - management of CPU and memory

RESOURCES

- Lesson Tasks
- TeachICT operating systems
- Video on Operating Systems
- Operating Systems Key Words Worksheet
- Operating systems tasks list

NOTES

- *No notes yet!*

Entry Level Computer Science
Application Software

LESSON SUMMARY

In this lesson pupils will learn about the different ways in which we often categorise application software, and about how it is used.

OBJECTIVES

- Identify a range of common application software packages and understand their uses, such as:
  - Image Processing
  - Word Processing
  - Spreadsheet
  - Web Browsers
  - Presentation
  - Database
  - Integrated Development Environment (IDE)
- identify examples of application software and system software

RESOURCES

- Matching Exercise
- Some good lecture notes on Application Software
- Spreadsheet to fill in

NOTES

- *No notes yet!*

Entry Level Computer Science
Memory and Morality
5 Lessons

Lesson 1	Primary & Secondary Storage
Lesson 2	Environmental and Legal Issues
Lesson 3	Computer Science Legislation
Lesson 4	Moral Issues
Lesson 5	Open Source and Proprietary Software

Entry Level Computer Science
Primary and Secondary Storage

LESSON SUMMARY

In this lesson pupils will learn about how computer memory is classified as either primary or secondary storage, and what this means.

OBJECTIVES

- describe the purpose of RAM
- describe the purpose of Cache
- describe the purpose of ROM
- explain the purpose of secondary storage
- give examples of common types of secondary storage devices and key characteristics:
  - Magnetic (Hard Disk Drive, Tape Drive)
  - Optical (CD ROM, DVD)
  - Flash Memory (Solid State Drive, SD Card and USB Pen Drive)
- identify appropriate use of secondary storage devices with respect to:
  - capacities
  - speed
  - portability
  - cost

RESOURCES

- BBC bitesize on memory (includes video)
- Storage Devices Table to Fill In

NOTES

- *This is a lot for one lesson, but some content should have been covered during the hardware lessons earlier.*

Entry Level Computer Science
Environmental and Legal Issues

LESSON SUMMARY

In this lesson pupils will look at some of the environmental and legal issues businesses have to consider with regards to their use of computers.

OBJECTIVES

- describe Computer Science technologies with consideration of legal issues, for example:
  - use of computer to commit crime (hacking)
  - risks of access to people's data
- describe Computer Science technologies with consideration of environmental issues, for example:
  - recycling and waste
  - energy use
  - improvements in manufacturing

RESOURCES

- Ewaste Management Video
- How much energy does the internet use? Video
- Data Protection Video

NOTES

- *Note that this ties into the following lesson (computer science legislation)*
- *This lesson lends itself to "fake debate". One side could argue for financial reasons & new technology, the other for privacy or the environment.*

Entry Level Computer Science
Computer Science Legislation

LESSON SUMMARY

In this lesson pupils learn about some of the key features of British legislation that effects the use of computers.

OBJECTIVES

- state the purpose of each of the following legislations:
  - Data Protection Act (1998)
  - Computer Misuse Act (1990)
  - Copyright, Designs and Patents Act (1998)

RESOURCES

- BBC Bitesize Legal Framework Section
- Schooltube OCR Computer Misuse Act Video (with links to other two acts)
- Legislation Worksheet

NOTES

- *No notes yet!*

Entry Level Computer Science
Moral Issues

LESSON SUMMARY

In this lesson pupils will explore some of the moral issues that have arisen as the result of developments in technology.

OBJECTIVES

- describe Computer Science technologies with consideration of moral issues, for example:
  - replacing of humans with computers
  - changing the shape of the world
  - spreading information and right of privacy

RESOURCES

- BBC – Privacy vs Security
- "Robots taking your job" Video
- AI news piece – Video
- "Moral dilemma with self driving cars" - Video

NOTES

- *No notes yet!*

Main Menu

7 8 9 10 11

Back

Entry Level Computer Science
Open Source and Proprietary Software

Langtree Computing

LESSON SUMMARY

In this lesson pupils will be introduced to two of the main forms of software licencing, "Open Source" and "Proprietary" software.

OBJECTIVES

- open source versus proprietary software
  - differences between cost, support, and customisation
- understand that laws exist that affect and control computer use

RESOURCES

- BBC bitesize page on topic

NOTES

- *No notes yet!*

Main Menu

7 8 9 10 11

Back

Entry Level Computer Science Programming Project

In this programming project, learners will be expected to plan, write, test and evaluate a simple coded program. The project will incorporate:

CLICK ON ANY SECTION FOR MORE GUIDANCE

The task will be chosen from three programming projects provided by OCR.
Learners will need to create suitable algorithms which will provide a solution to the problems identified in the task. They will then code their solution in a suitable programming language. The solution must be tested at each stage to ensure they solve the stated problem and learners must use a suitable test plan with appropriate test data. The code must be suitably annotated to describe the process. Test results should be annotated to show how these relate to the code, the test plan and the original problem. Learners will need to provide an evaluation of their solution based on the test evidence.
Group work may be used to deliver the content and skills but any work submitted must be the learner's own.
Learners should be innovative and creative in how they approach solving the tasks.
Learners can use any suitable programming language which allows them to access all of the programming techniques as listed within the subject content. *At Langtree, this is likely to be Scratch for block based projects or Python for text based projects.*
For further information, please read the "Teacher's Handbook" or OCR specification, both of which can be found on the OCR course website.

Programming Project – Mark Scheme

Main Menu

7 8 9 10 11

Back

Entry Level Computer Science
Programming Project – Success Criteria

OCR Says…

Success Criteria – learners will need to read and understand each component of the programming project to be able to create a list of success criteria.

We say…

- You need to **write a list of what your program will do**.
- Try to be as detailed as possible, but aim to complete this in 1 lesson or less.

Be SMART – Make your objectives:

- Specific – Exactly how will it work?
- Measurable – How can someone tell if you have succeeded?
- Achievable – Only put objectives you know how to achieve!
- Relevant – Make sure your objectives are linked to your instructions
- Time Bound – Don't put objectives that will take too long.

Success Criteria (0–3 marks available)

	Marks		
Marking criteria	**1**	**2**	**3**
AO1–0 AO2–3 AO3–0	• There is an attempt to identify some success criteria, but these only cover part of the solution and are incomplete	• There is an attempt to identify most success criteria adequately, and these relate to the majority of the requirements listed	• There is an attempt to convincingly identify the majority of success criteria, and these relate to the requirements listed

0 marks = no response or no response worthy of credit.

Entry Level Computer Science
Programming Project – Plan and Design

OCR Says…

Plan and Design – learners will be required to develop a flow chart solution to the problem based on their project success criteria. They will also be expected to identify some simple tests they may carry out. Learners will be expected to use Input, Output and Storage of data within their project.

We say…

- This section involves **writing a flowchart to show how your program will work**, and a **test plan to show how you will test it**.
- This section will take a bit of time (maybe 2 lessons), but don't go overboard with your flowcharts. You need to show a logical structure to your program but you do not need to show the tiniest detail for your Entry Level.

Planning and Design (0–6 marks available)

	Marks		
Marking criteria	**1–2**	**3–4**	**5–6**
AO1–0 AO2–2 AO3–4	• There is a statement(s) of what the intended program will do. This may not always reflect what the task requires • There is an attempt to create a flow chart for the program, but this is incomplete or appears non-functional. • There are some tests given, but these are basic and use normal data only • There is an attempt to identify any Input, Output and Processing needs.	• The learner has outlined how their program will work and this adequately matches any success criteria given • There is a flow chart produced that adequately maps a working solution to the problem, although it may contain some errors • There are a range of tests suggested using normal and erroneous data, but do not cover the entire solution proposed • Input, Output and Processing needs are identified and adequately meet the solution, although may not be complete.	• The learner has described how their program will work and this convincingly matches the needs of the task. • There is an accurate flow chart representing the proposed solutions that convincingly produces a functional solution. • There are normal and erroneous tests for all parts of the solution as needed and the tests would provide convincing evidence that the solution is effective. • Input, Output and Processing needs are clearly identified and cover all areas of the solution

0 = no response or no response worthy of credit.

Entry Level Computer Science
Programming Project – Development

OCR Says…

Development - learners will be expected to use combinations of sequence, selection, iteration, arithmetic operations and comments to create the solution to the task

We say…

- This is the section in which you **write your program**.
- This section will probably take the most time (maybe 3 or 4 lessons)
- Try and produce a basic working program, then gradually build on it until you have the result you need.
- Don't try and add any extra features unless you are certain your write-up for the other sections is 100% perfect

Development (0–5 marks available)

	Marks		
Marking criteria	**1**	**2 – 3**	**4–5**
AO1–0 AO2–0 AO3–5	• There is evidence of some of the following techniques: • Input • Output • Data Storage • Selection • Iteration • Arithmetic Operators • Comments	• There is evidence of a range of the following techniques, which may not always be used efficiently: • Input • Output • Data Storage • Selection • Iteration • Arithmetic Operators • Comments	• There is convincing evidence of a range of the following techniques, which are generally used efficiently: • Input • Output • Data Storage • Selection • Iteration • Arithmetic Operators • Comments
	• There is little evidence of the development of the program, which will be limited and may not fully describe the steps taken to reach a solution.	• There is adequate evidence showing the development of the solution, but this may contain omissions.	• There is convincing evidence of the development of the solution and it provides a full narrative of the process.
	• The explanations of the code leave doubt as to the understanding of the techniques used.	• The explanations adequately support the learners understanding of the techniques used.	• Explanations convincingly explain the learner's understanding of techniques used.

0 = no response or no response worthy of credit.

Main Menu

7 8 9 10 11

Back

Entry Level Computer Science
Programming Project – Testing and Remedial Action

OCR Says…

Testing and Remedial Action - learners will be expected to carry out some basic normal and erroneous testing to check that their solution works

We say…

- In this section you **complete the tests you planned** in your planning section.
- As these tests are already written, this shouldn't take long (maybe 1/2 lesson) although you should screenshot to evidence that your tests work/don't work.

Testing and Remedial Actions (0–3 marks available)

	Marks		
Marking criteria	**1**	**2**	**3**
AO1–0 AO2–1 AO3–2	• The results of limited tests are evidenced with respects to success or failure • There is no or little evidence given to show an attempt to correct errors that are found in the solution	• The results of most tests are evidenced with respect to success or failure • There is some evidence given to show that errors have been adequately solved and re-tested	• The results of all tests are evidenced with respect to success or failure • There is convincing evidence that errors have been corrected and the program is functional

0 = no response or no response worthy of credit.

Entry Level Computer Science
Programming Project – Evaluation

Langtree Computing

OCR Says…

Evaluation – learners will be expected to show that they have reflected on their solutions, and compare this solution to the initial goals they set

We say…

- In this section you **write about how well your program works**.
- This should be linked to your success criteria from the start. Have you achieved them?

Evaluation (0–3 marks available)

	Marks		
Marking criteria	**1**	**2**	**3**
AO1–2 AO2–0 AO3–1	• There are limited statements about whether the solution has been successful • The link between evidence of testing and Success Criteria is weak and vague	• There are some statements that adequately review the success of the project • The link between evidence of testing and Success Criteria is adequate	• There is full coverage of statements to reflect the Success Criteria • The link between evidence of testing and Success Criteria is convincing and covers all Success Criteria.

0 = no response or no response worthy of credit

Entry Level - Tests

- Each test is 30 minutes long
- There are 20 marks available in a test
- Most questions are short answer or multi-choice, but look out for 3 or 4 mark questions which require a longer answer.
- We intentionally do the tests in the order show below (3, 4, 1, 2)

- You can try our practice style tests by opening them from the links below.

PRACTICE TEST 3
(Logic & Algorithms)

PRACTICE TEST 4
(Programming and Data Representation)

PRACTICE TEST 1
(Hardware & Software)

PRACTICE TEST 2
(Memory & Morality)

Langtree Creativity Week

- Various content/trips linked to a creative theme.
- Not specific to the Computing Course
- Likely that staff are allocated to trips etc

Final Week Fun

- “Fun” activities linked to the learning of the year

Main Menu

7 8 9 10 11

Back

Langtree School
PUTTING LEARNING FIRST

Langtree Computing

Enjoy Your Holidays!

Main Menu

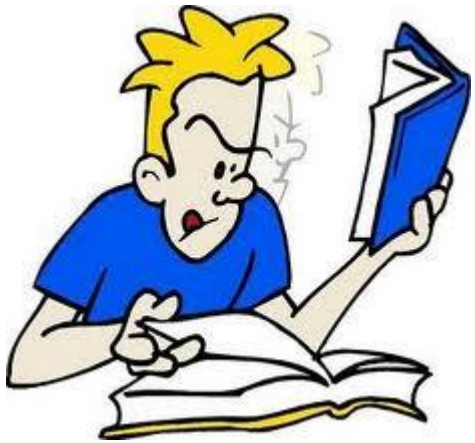
7 8 9 10 11

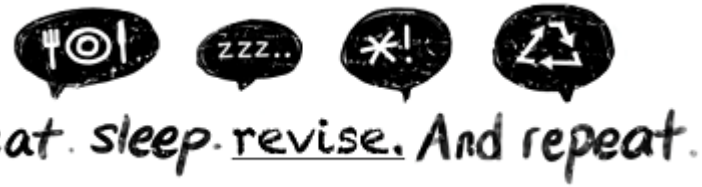
Back

Good Luck!

Langtree Computing

1) Find out about the exam paper
2) Gather and organise revision material
3) Decide what to revise
4) Make a revision timetable
5) Understand the course content
6) Practise answering exam questions
7) Repeat steps 5 to 7





Main Menu

Back

Key Stage 4 Computing Programming

Langtree Computing

Objectives From Specification

Python: Magic 8

Python: Cheat Sheet

Python: String Manipulation

Python: Understanding Code Challenge

Python: Turtle

Useful Websites

codecademy

CODE COMBAT

C O
D E

Main Menu

7 8 9 10 11

Back

Key Stage 4 Theory
2.2 & 2.3 Programming

Learners should have studied the following:
(2.2)
• the use of variables, constants, operators, inputs, outputs and assignments
• the use of the three basic programming constructs used to control the flow of a program:
- sequence
- selection
- iteration (count and condition controlled loops)
• the use of basic string manipulation
• the use of basic file handling operations:
- open
- read
- write
- close
• the use of records to store data
• the use of SQL to search for data
• the use of arrays (or equivalent) when solving problems, including both one and two dimensional
arrays
• how to use sub programs (functions and procedures) to produce structured code

• the use of data types:
- integer
- real
- Boolean
- character and string
- casting
• the common arithmetic operators
(2.3)
• defensive design considerations:
- input sanitisation/validation
- planning for contingencies
- anticipating misuse
- authentication
• maintainability:
- comments
- indentation
• the purpose of testing
• types of testing:
- iterative
- final/terminal
• how to identify syntax and logic errors
• selecting and using suitable test data.

Main Menu

7 8 9 10 11

Back

Controlled Assignment 1

- This time will be used to complete the first controlled assignment.

Key Stage 4 Theory Lesson Sets

Langtree Computing

- 1.1 Systems Architecture
- 1.2 Memory
- 1.3 Storage
- 1.4 Networks + 1.5 Network Topologies and Protocols
- 1.6 System Security
- 1.7 Software
- 1.8 Ethical, Legal, Social and Environmental Concerns
- 2.1 Algorithms
- 2.2 Programming + 2.3 Robust Programs
- 2.4 Computational Logic
- 2.5 Translators and facilities of languages
- 2.6 Data Representation

THEORY SUPPORT WEBSITE

Main Menu

7 8 9 10 11

Back

Key Stage 4 Theory
1.1 Systems Architecture

Learners should have studied the following:
• the purpose of the CPU
• Von Neumann architecture:
- MAR (Memory Address Register)
- MDR (Memory Data Register)
- Program Counter
- Accumulator
• common CPU components and their function:
- ALU (Arithmetic Logic Unit)
- CU (Control Unit)
- Cache
• the function of the CPU as fetch and execute instructions stored in memory
• how common characteristics of CPUs affect their performance:
- clock speed
- cache size
- number of cores
• embedded systems:
- purpose of embedded systems
- examples of embedded systems.

Lesson 1	The CPU
Lesson 2	Von Neumann Architecture
Lesson 3	Embedded Systems

Main Menu

7 8 9 10 11

Back

Systems Architecture 1.1
The CPU

LESSON SUMMARY

OBJECTIVES

- the purpose of the CPU
- how common characteristics of CPUs affect their performance:
  - clock speed
  - cache size
  - number of cores
- the function of the CPU as fetch and execute instructions stored in memory
- common CPU components and their function:
  - ALU (Arithmetic Logic Unit)
  - CU (Control Unit)
  - Cache

RESOURCES

- https://www.youtube.com/channel/UCrQ9NI6V73zY48Zd4pMkoHw/playlists
- *next*

NOTES

- *No notes yet!*

Main Menu

7 8 9 10 11

Back

Systems Architecture 1.1
Von Neumann Architecture

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- Von Neumann architecture:
  - MAR (Memory Address Register)
  - MDR (Memory Data Register)
  - Program Counter
  - Accumulator

RESOURCES

NOTES

- *No notes yet!*

Systems Architecture 1.1
Embedded Systems

LESSON SUMMARY

OBJECTIVES

- embedded systems:
- purpose of embedded systems
- examples of embedded systems.

RESOURCES

NOTES

- *No notes yet!*

Main Menu

7 8 9 10 11

Back

Key Stage 4 Theory
1.2 Memory

Learners should have studied the following:
- the difference between RAM and ROM
- the purpose of ROM in a computer system
- the purpose of RAM in a computer system
- the need for virtual memory
- flash memory.

Lesson 1	RAM & ROM
Lesson 2	Virtual Memory

Main Menu

7 8 9 10 11

Back

1.2 Memory
RAM & ROM

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- the difference between RAM and ROM
- the purpose of ROM in a computer system
- the purpose of RAM in a computer system
- flash memory

RESOURCES

NOTES

- *No notes yet!*

Main Menu

Back

1.2 Memory
Virtual Memory

LESSON SUMMARY

OBJECTIVES

RESOURCES

NOTES

- *No notes yet!*

Main Menu

Back

Key Stage 4 Theory
1.3 Storage

Learners should have studied the following:
• the need for secondary storage
• data capacity and calculation of data capacity requirements
• common types of storage:
- Optical
- magnetic
- solid state
• suitable storage devices and storage media for a given application, and the advantages and disadvantages of these, using characteristics:
- capacity
- speed
- portability
- durability
- reliability
- cost

Lesson 1	Data Capacity Calculations
Lesson 2	Storage Devices

Main Menu

7 8 9 10 11

Back

1.3 Storage
Data Capacity Calculations

LESSON SUMMARY

OBJECTIVES

- the need for secondary storage
- data capacity and calculation of data capacity requirements

RESOURCES

NOTES

- *No notes yet!*

1.3 Storage
Storage Devices

LESSON SUMMARY

OBJECTIVES

- common types of storage:
  - Optical
  - magnetic
  - solid state
- suitable storage devices and storage media for a given application, and the advantages and disadvantages of these, using characteristics:
  - capacity
  - speed
  - portability
  - durability
  - reliability
  - cost

RESOURCES

NOTES

- *No notes yet!*

Main Menu

7 8 9 10 11

Back

Key Stage 4 Theory
1.4 & 1.5 Networks

Learners should have studied the following:
(1.4)
• types of networks:
- LAN (Local Area Network)
- WAN (Wide Area Network)
• factors that affect the performance of networks
• the different roles of computers in a client-server and a peer-to-peer network
• the hardware needed to connect stand-alone computers into a Local Area Network:
- wireless access points
- routers/switches
- NIC (Network Interface Controller/Card)
- transmission media
• the internet as a worldwide collection of computer networks:
- DNS (Domain Name Server)
- hosting
- the cloud
• the concept of virtual networks.
(1.5)
• star and mesh network topologies
• Wifi:
- frequency and channels
- encryption
• ethernet
• the concept of layers
• packet switching.

Learners should have studied the following:
• the uses of IP addressing, MAC addressing, and protocols including:
- TCP/IP (Transmission Control Protocol/Internet Protocol)
- HTTP (Hyper Text Transfer Protocol)
- HTTPS (Hyper Text Transfer Protocol Secure)
- FTP (File Transfer Protocol)
- POP (Post Office Protocol)
- IMAP (Internet Message Access Protocol)
- SMTP (Simple Mail Transfer Protocol)

1.4 & 1.5 Networks
Networks Introduction

Langtree Computing

LESSON SUMMARY

This lesson will introduce students to the concept of a "network" and should give some overview of the different types of network.

OBJECTIVES

- types of networks:
  - LAN (Local Area Network)
  - WAN (Wide Area Network)
- factors that affect the performance of networks
- the different roles of computers in a client-server and a peer-to-peer network

RESOURCES

- Starter Activity: Why share resources
- Introduction to Networks: video lecture
- LAN/MAN/WAN video
- Website about LAN vs WAN
- Client-Server & peer-to-peer diagram

NOTES

- *No notes yet!*

1.4 & 1.5 Networks
Network Hardware

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- the hardware needed to connect stand-alone computers into a Local Area Network:
- wireless access points
- routers/switches
- NIC (Network Interface Controller/Card)
- transmission media

RESOURCES

NOTES

- *No notes yet!*

Main Menu

Back

1.4 & 1.5 Networks
Network Topologies

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- star and mesh network topologies

RESOURCES

NOTES

- *No notes yet!*

Main Menu

Back

1.4 & 1.5 Networks
The Internet

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- the internet as a worldwide collection of computer networks:
  - DNS (Domain Name Server)
  - hosting
  - the cloud
- the concept of virtual networks.

RESOURCES

NOTES

- *No notes yet!*

Main Menu

7 8 9 10 11

Back

1.4 & 1.5 Networks
Network Protocols

LESSON SUMMARY

OBJECTIVES

• the uses of IP addressing, MAC addressing, and protocols including:
- TCP/IP (Transmission Control Protocol/Internet Protocol)
- HTTP (Hyper Text Transfer Protocol)
- HTTPS (Hyper Text Transfer Protocol Secure)
- FTP (File Transfer Protocol)
- POP (Post Office Protocol)
- IMAP (Internet Message Access Protocol)
- SMTP (Simple Mail Transfer Protocol)

RESOURCES

NOTES

- *No notes yet!*

1.4 & 1.5 Networks Everything Else

LESSON SUMMARY

OBJECTIVES

- Wifi:
- frequency and channels
- encryption
- ethernet
- the concept of layers
- packet switching.

RESOURCES

NOTES

- *No notes yet!*

Main Menu

7 8 9 10 11

Back

Key Stage 4 Theory
1.6 System Security

Learners should have studied the following:
• forms of attack
• threats posed to networks:
- malware
- phishing
- people as the 'weak point' in secure systems (social engineering)
- brute force attacks
- denial of service attacks
- data interception and theft
- the concept of SQL injection
- poor network policy
• identifying and preventing vulnerabilities:
- penetration testing
- network forensics
- network policies
- anti-malware software
- firewalls
- user access levels
- passwords
- encryption.

1.6 System Security
Malware

LESSON SUMMARY

OBJECTIVES

- threats posed to networks:
- malware

RESOURCES

NOTES

- *No notes yet!*

Main Menu

Back

1.6 System Security
Security Software

LESSON SUMMARY

OBJECTIVES

- anti-malware software
- firewalls
- user access levels
- passwords
- encryption
- network policies

RESOURCES

NOTES

- *No notes yet!*

1.6 System Security
Forms of Attack

LESSON SUMMARY

OBJECTIVES

- brute force attacks
- denial of service attacks
- data interception and theft
- the concept of SQL injection
- poor network policy
- phishing
- penetration testing

RESOURCES

NOTES

- *No notes yet!*

1.6 System Security
User Vulnerability

LESSON SUMMARY

OBJECTIVES

- people as the 'weak point' in secure systems (social engineering)

RESOURCES

NOTES

- *No notes yet!*

Key Stage 4 Theory
1.7 Software

Learners should have studied the following:
• the purpose and functionality of systems software
• operating systems:
- user interface
- memory management/multitasking
- peripheral management and drivers
- user management
- file management
• utility system software:
- encryption software
- defragmentation
- data compression
• the role and methods of backup:
- full
- incremental.

Lesson 1	Operating Systems
Lesson 2	Utility Software
Lesson 3	Backups

1.7 Software Operating Systems

LESSON SUMMARY

OBJECTIVES

- operating systems:
- user interface
- memory management/multitasking
- peripheral management and drivers
- user management
- file management

RESOURCES

NOTES

- *No notes yet!*

Main Menu

7 8 9 10 11

Back

1.7 Software Utility Software

LESSON SUMMARY

OBJECTIVES

- utility system software:
  - encryption software
  - defragmentation
  - data compression

RESOURCES

NOTES

- *Worth mentioning that previously studied security software is also utility software*

1.7 Software Backups

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- the role and methods of backup:
  - full
  - incremental

RESOURCES

NOTES

- *Should tie into storage lessons*

Key Stage 4 Theory
1.8 Ethical, legal, cultural and environmental concerns

Learners should have studied the following:
• how to investigate and discuss Computer Science technologies while considering:
- ethical issues
- legal issues
- cultural issues
- environmental issues.
- privacy issues.
• how key stakeholders are affected by technologies
• environmental impact of Computer Science
• cultural implications of Computer Science
• open source vs proprietary software
• legislation relevant to Computer Science:
- The Data Protection Act 1998
- Computer Misuse Act 1990
- Copyright Designs and Patents Act 1988
- Creative Commons Licensing
- Freedom of Information Act 2000.

Lesson 1	Computing and the Environment
Lesson 2	Cultural Implications of Computer Science
Lesson 3	Computer Science and the Law
Lesson 4	Data Protection Act
Lesson 5	Open Source Software
Lesson 6	Exam Technique – "Discuss"

1.8 Ethical, legal, cultural and environmental concerns
Computing and the Environment

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- how key stakeholders are affected by technologies
- environmental impact of Computer Science

RESOURCES

NOTES

- *No notes yet!*

Main Menu

7 8 9 10 11

Back

1.8 Ethical, legal, cultural and environmental concerns
Cultural Implications of Computer Science

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- cultural implications of Computer Science

RESOURCES

NOTES

- *No notes yet!*

Main Menu

Back

1.8 Ethical, legal, cultural and environmental concerns
Computer Science and the Law

LESSON SUMMARY

OBJECTIVES

• legislation relevant to Computer Science:
- (*The Data Protection Act 1998)*
- Computer Misuse Act 1990
- Copyright Designs and Patents Act 1988
- Creative Commons Licensing
- Freedom of Information Act 2000.

RESOURCES

NOTES

- *No notes yet!*

1.8 Ethical, legal, cultural and environmental concerns
The Data Protection Act

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- legislation relevant to Computer Science:
- The Data Protection Act 1998

RESOURCES

NOTES

- *No notes yet!*

1.8 Ethical, legal, cultural and environmental concerns
Open Source Software

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- open source vs proprietary software

RESOURCES

NOTES

- *No notes yet!*

Main Menu

Back

1.8 Ethical, legal, cultural and environmental concerns
Exam Technique – "Discuss"

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- how to investigate and discuss Computer Science technologies while considering:
  - ethical issues
  - legal issues
  - cultural issues
  - environmental issues
  - privacy issues

RESOURCES

NOTES

- *No notes yet!*

Key Stage 4 Theory
2.1 Algorithms

Learners should have studied the following:
• computational thinking:
- abstraction
- decomposition
- algorithmic thinking
• standard searching algorithms:
- binary search
- linear search
• standard sorting algorithms:
- bubble sort
- merge sort
- insertion sort
• how to produce algorithms using:
- pseudocode
- using flow diagrams
• interpret, correct or complete algorithms.

Lesson 1	Introduction to Computational Thinking
Lesson 2	Flow Charts
Lesson 3	Pseudocode
Lesson 4	Searching Algorithms
Lesson 5	Sorting Algorithms
Lesson 6	Algorithms Practice

2.1 Algorithms
Introduction to Computational Thinking

LESSON SUMMARY

OBJECTIVES

- computational thinking:
  - abstraction
  - decomposition
  - algorithmic thinking

RESOURCES

NOTES

- *No notes yet!*

Main Menu

7 8 9 10 11

Back

2.1 Algorithms Flowcharts

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- how to produce algorithms using:
- using flow diagrams

RESOURCES

- Presentation – Algorithms and Flow Charts
- Interactive Flow Chart Tasks
- Flowcharts – Worksheet
- Flowcharts – Breakdown Challenge

NOTES

- *No notes yet!*

2.1 Algorithms
Pseudocode

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- how to produce algorithms using:
  - pseudocode

RESOURCES

NOTES

- *No notes yet!*

Main Menu

Back

2.1 Algorithms
Searching Algorithms

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- standard searching algorithms:
  - binary search
  - linear search

RESOURCES

NOTES

- *No notes yet!*

2.1 Algorithms
Sorting Algorithms

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- standard sorting algorithms:
  - bubble sort
  - merge sort
  - insertion sort

RESOURCES

NOTES

- *No notes yet!*

Main Menu

7 8 9 10 11

Back

2.1 Algorithms
Algorithms Practice

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- interpret, correct or complete algorithms.

RESOURCES

NOTES

- *No notes yet!*

Key Stage 4 Theory
2.4 Computational Logic

Learners should have studied the following:
- why data is represented in computer systems in binary form
- simple logic diagrams using the operations AND, OR and NOT
- truth tables
- combining Boolean operators using AND, OR and NOT to two levels
- applying logical operators in appropriate truth tables to solve problems
- applying computing-related mathematics:
- Addition +
- Subtraction –
- Division /
- Multiplication *
- Exponentiation (^)
- MOD %
- DIV

Lesson 1	Logic Gates
Lesson 2	Truth Tables
Lesson 3	Why Binary

2.4 Computational Logic
Logic Gates

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- simple logic diagrams using the operations AND, OR and NOT

RESOURCES

NOTES

- *No notes yet!*

2.4 Computational Logic
Truth Tables

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- truth tables
- combining Boolean operators using AND, OR and NOT to two levels
- applying logical operators in appropriate truth tables to solve problems

RESOURCES

NOTES

- *No notes yet!*

Main Menu

7 8 9 10 11

Back

2.4 Computational Logic
Why Binary

LESSON SUMMARY

OBJECTIVES

- why data is represented in computer systems in binary form
- applying computing-related mathematics:
  - Addition +
  - Subtraction –
  - Division /
  - Multiplication *
  - Exponentiation (^)
  - MOD %
  - DIV

RESOURCES

NOTES

- *No notes yet!*

Main Menu

7 8 9 10 11

Back

Key Stage 4 Theory

2.5 Translators and facilities of languages

Learners should have studied the following:
• characteristics and purpose of different levels of programming language, including low level languages
• the purpose of translators
• the characteristics of an assembler, a compiler and an interpreter
• common tools and facilities available in an integrated development environment (IDE):
- editors
- error diagnostics
- run-time environment

Lesson 1	Low Level vs High Level Languages
Lesson 2	Assemblers, compilers and interpreters
Lesson 3	Intergrated Development Environments

2.5 Translators and facilities of languages
Low Level vs High Level Languages

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- characteristics and purpose of different levels of programming language, including low level languages
- the purpose of translators

RESOURCES

NOTES

- *No notes yet!*

Main Menu

Back

2.5 Translators and facilities of languages
Assemblers, compilers and Interpreters

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- the characteristics of an assembler, a compiler and an interpreter

RESOURCES

NOTES

- *No notes yet!*

Main Menu

Back

2.5 Translators and facilities of languages
Integrated Development Environments

Langtree Computing

LESSON SUMMARY

OBJECTIVES

- common tools and facilities available in an integrated development environment (IDE):
  - editors
  - error diagnostics
  - run-time environment

RESOURCES

NOTES

- *No notes yet!*

Key Stage 4 Theory
2.6 Data Representation

Learners should have studied the following:

Units
- bit, nibble, byte, kilobyte, megabyte, gigabyte, terabyte, petabyte
- how data needs to be converted into a binary format to be processed by a computer.

Numbers
- how to convert positive denary whole numbers (0–255) into 8 bit binary numbers and vice versa
- how to add two 8 bit binary integers and explain overflow errors which may occur
- binary shifts
- how to convert positive denary whole numbers (0–255) into 2 digit hexadecimal numbers and vice versa
- how to convert from binary to hexadecimal equivalents and vice versa
- check digits.

Characters
- the use of binary codes to represent characters
- the term 'character-set'
- the relationship between the number of bits per character in a character set and the number of characters which can be represented (for example ASCII, extended ASCII and Unicode).

Images
- how an image is represented as a series of pixels represented in binary
- metadata included in the file
- the effect of colour depth and resolution on the size of an image file

Sound
- how sound can be sampled and stored in digital form
- how sampling intervals and other factors affect the size of a sound file and the quality of its playback:
  - sample size
  - bit rate
  - sampling frequency.

Compression
- need for compression
- types of compression:
  - lossy
  - lossless

2.6 Data Representation Units

LESSON SUMMARY

OBJECTIVES

- bit, nibble, byte, kilobyte, megabyte, gigabyte, terabyte, petabyte
- how data needs to be converted into a binary format to be processed by a computer.

RESOURCES

NOTES

- *No notes yet!*

2.6 Data Representation Numbers

LESSON SUMMARY

OBJECTIVES

- how to convert positive denary whole numbers (0–255) into 8 bit binary numbers and vice versa
- how to add two 8 bit binary integers and explain overflow errors which may occur
- binary shifts
- how to convert positive denary whole numbers (0–255) into 2 digit hexadecimal numbers and vice versa
- how to convert from binary to hexadecimal equivalents and vice versa
- check digits.

RESOURCES

NOTES

- *This is probably at least 2 lessons work in reality*

Main Menu

7 8 9 10 11

Back

2.6 Data Representation
Characters

LESSON SUMMARY

OBJECTIVES

- the use of binary codes to represent characters
- the term 'character-set'
- the relationship between the number of bits per character in a character set and the number of characters which can be represented (for example ASCII, extended ASCII and Unicode).

RESOURCES

NOTES

- *No notes yet!*

Main Menu

7 8 9 10 11

Back

2.6 Data Representation
Images

LESSON SUMMARY

OBJECTIVES

- how an image is represented as a series of pixels represented in binary
- metadata included in the file
- the effect of colour depth and resolution on the size of an image file

RESOURCES

NOTES

- *No notes yet!*

Main Menu

Back

2.6 Data Representation
Sound

LESSON SUMMARY

OBJECTIVES

- how sound can be sampled and stored in digital form
- how sampling intervals and other factors affect the size of a sound file and the quality of its playback:
  - sample size
  - bit rate
  - sampling frequency.

RESOURCES

NOTES

- *No notes yet!*

2.6 Data Representation
Compression

LESSON SUMMARY

OBJECTIVES

- need for compression
- types of compression:

- lossy
- lossless

RESOURCES

NOTES

- *No notes yet!*

Main Menu

Back

Python: Turtle

LESSON SUMMARY

In this lesson the students learn about the "Turtle" module within python and use this module to practise basic programming.

OBJECTIVES

- Understand how to import a python library into a program
- Use algorithms to solve problems involving turtle diagrams.

RESOURCES

- Turtles Worksheet
- Python Cheat Sheet

NOTES

- *No notes yet!*

Python: Magic 8

LESSON SUMMARY

In this lesson the students will practise their programming by making a simple "Magic 8 Ball" program in python.

OBJECTIVES

- Understand how to generate a random number within python.
- Be able to take a user's input and store it as a variable within python.

RESOURCES

- Magic 8 Video
- Code Club Worksheet

NOTES

- *No notes yet!*

Langtree School
PUTTING LEARNING FIRST

Python: Cheat Sheet

Langtree Computing

Variable Assignment

```
integer = 1
string = "string"
unicode_string = u"unicode string"
mutli_line_string = """ multi-line
        string
        """
tuple = (element1, element2, element3, ...)
list = [ element1, element2, element3, ... ]
dictionary = { key1 : value1, key2 : value2, ... }
dictionary[key] = value
class_instance = ClassName(init_args)
```

Basic Arithmetic

```
i = a + b        i = a - b
i = a / b        i = a * b
i = a % b        e.g. 11 % 3 → 2
```

Frequently Used Built-in Types

True	**False**	**None**
str	**unicode**	**int**
float	**list**	**dict**

Other than **True**, **False** and **None**, these can also be used as functions to explicitly cast a value to that type

Comparisons

value1 == *value2*	"str" == "str" → **True**
value1 != *value2*	"str" != "str" → **False**
value1 < *value2*	1 < 2 → **True**
value1 <= *value2*	2 <= 2 → **True**
value1 > *value2*	2 > 3 → **False**
value1 >= *value2*	3 >= 3 → **True**
value **is [not] None**	
value **in** *list*	1 in [2,3,4] → **False**

isinstance(*class_instance*, *ClassName*)

Frequently Used String Manipulations

string1 + *string1*	"str" + "ing" → "string"
"%s%s" % (*string1*, *string2*)	"%s%s" % ("s", "g") → "sg"
string.split("*delim*", *limit*)	"s/g".split("/") → ["s", "g"]
string.strip()	" string ".strip() → "string"
string.startswith("*prefix*")	"str".startswith("s") → True
substring in *string*	"str" in "string" → True

print *string*

Imports

```
import module
from module import class, function, variable
```

Comments

```
"""                    # Line Comment
   Multi-line comment
"""
```

Functions

```
def function_name(arg1, arg2,
                  keyword1=val1, keyword2=val2, ...):
   <function body>
   return return_value
```

e.g.

```
def my_function(x, y, z=0):      my_function(1, 2) → 3
   sum = x + y + z               my_function(1, 2, 3) → 6
   return sum                    my_function(1, 2, y=4) → 7
```

Control Flow

```
if conditional:                  if i == 7:
   <body>                           print "seven"
elif conditional:          e.g.  elif i == 8:
   <body>                           print "eight"
else:                            else:
   <body>                           print str(i)
for value in list:               for i in [1, 2, 3, 4]:
   <body>                  e.g.     if i == 2: continue
   continue                         if i == 3: break
   break                            print i
while conditional:               while True:
   <body>                  e.g.     print "infinity"
   continue
   break
```

File & Path Manipulation

```
import os    # import the os module first
os.path.join(path_segment1, path_segment2, ...)
os.path.exists(path)
os.listdir(directory_path)
os.remove(file_path)
os.rmdir(directory_path)
file = open(path, "rw")
file.read()
string.write("string")
```

Classes

```
class ClassName(SuperClass):
   class_variable = static_value
   def __init__(self, value1, <...>):
      self.instance_variable1 = value1
      self.instance_function()
   def instance_function(self, arg1, <...>):
      <function body>
      return return_value
```

e.g.

```
class MyClass(object):           MyClass.offset → 1
   offset = 1
   def __init__(self, value):    c = MyClass(2)
      self.value = value         c.value → 2
   def get_offset_value(self):   c.get_offset_value() → 3
      return MyClass.offset +
                    self.value
```

List Comprehension

```
[ value for value in list if condition ]
```

e.g.

```
[x for x in [1,2,3,4,5,6,7,8,9] if x % 2 == 0] → [2,4,6,8]
```

Exceptions

```
try:                             try:
   <body>                           database.update()
   raise Exception()       e.g.  except Exception as e:
except Exception as e:              log.error(e.msg)
   <exception handling>             database.abort()
finally:                         finally:
   <clean-up>                       database.commit()
```

Accessing Variable Values

```
value = dictionary[key]
value = dictionary.get(key, default_value)
value = list[index]              e.g. [5,6,7][2] → 7
value = string[start:end]        e.g "string"[0:3] → "str"
value = list[start:end]          e.g. [1,2,3][1:2] → [2]
value = ClassName.class_variable
value = class_instance.instance_variable
value = class_instance.function(args)
```


See the original here

by Cottage Labs (http://cottagelabs.com)
for Dev8D (http://www.dev8d.org/)

Python: String Manipulation

Lesson 1	LESSON NAME
Lesson 2	LESSON NAME

Main Menu

7 8 9 10 11

Back

Python: Conversation Program 1

LESSON SUMMARY

In this lesson the students will learn about the "print()" and "input()" commands. They will also learn how to combine strings using the + operator.

OBJECTIVES

- Know how to use the print() command to display text.
- Be able to take a user's input and store it as a string variable within python.
- Be able to combine string variables to create new strings.

RESOURCES

NOTES

- *No notes yet!*

Python: Understanding Code Challenge

LESSON SUMMARY

This lesson, which does not require computers, challenges the pupils to work out the output of some code which in some cases has very poorly named variables!

OBJECTIVES

- Understand the difference between a variable name, function name and string.
- Understand how arguments are passed into functions.

RESOURCES

- Level 1 task
- Level 2 task
- Level 3 task
- Level 4 task

NOTES

- *No notes yet!*

PROGRAMMING TASK 1 – GUESSING GAME

Langtree Computing

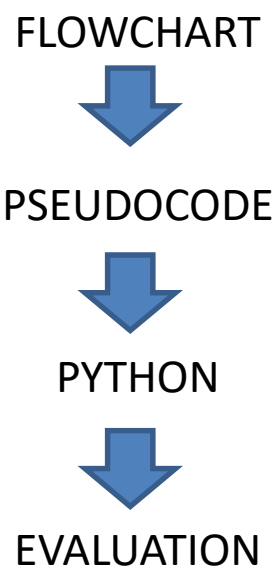
Write a guessing game where the user has to guess a secret number.
After every guess the program tells the user whether their number was too large or too small.
At the end the number of tries needed should be printed.
It only counts as one try if they input the same number multiple times consecutively.

1) Complete this task as a flowchart and then add the flowchart to a powerpoint as a picture or screenshot.

2) Next pseudocode the program. Screenshot this and add it to your powerpoint.

3) Now write the program using python. Screenshot the final outcome showing it works.

4) Write a brief evaluation. What has worked well? What could you change to improve in future?



Main Menu

Back

PROGRAMMING TASK 2 – SQUARE CODE

Langtree Computing

One classic method for composing secret messages is called a square code. The spaces are removed from the English text and the characters are written into a square (or rectangle). For example, the sentence "I love computing it is the best subject in school" is 40 characters long, so it is written into a rectangle with 6 rows and 7 columns. The coded message is obtained by reading down the columns going left to right. Write a program that converts a message to square code.

Example:

Ilovecomputingitisthebestsubjectinschool

↓

Imibjc lpteeh ouisco vtstto eitsil cnhun ogebs

i	l	o	v	e	c	o
m	p	u	t	i	n	g
i	t	i	s	t	h	e
b	e	s	t	s	u	b
j	e	c	t	i	n	s
c	h	o	o	l		

1) Complete this task as a flowchart and then add the flowchart to a powerpoint as a picture or screenshot.

2) Next pseudocode the program. Screenshot this and add it to your powerpoint.

3) Now write the program using python. Screenshot the final outcome showing it works.

4) Write a brief evaluation. What has worked well? What could you change to improve in future?

FLOWCHART

PSEUDOCODE

PYTHON

EVALUATION

Main Menu

7 8 9 10 11

Back

PROGRAMMING TASK 3 – PIG LATIN

Langtree Computing

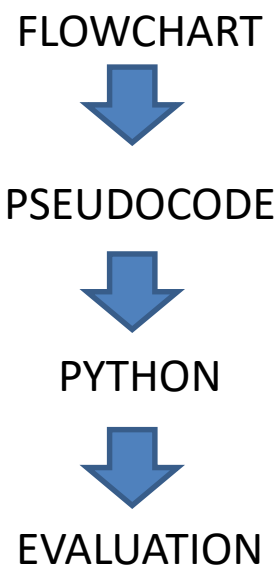
Write a program that translates a text to Pig Latin and back. English is translated to Pig Latin by taking the first letter of every word, moving it to the end of the word and adding 'ay'. "The quick brown fox" becomes "Hetay uickqay rownbay oxfay"

1) Complete this task as a flowchart and then add the flowchart to a powerpoint as a picture or screenshot.

2) Next pseudocode the program. Screenshot this and add it to your powerpoint.

3) Now write the program using python. Screenshot the final outcome showing it works.

4) Write a brief evaluation. What has worked well? What could you change to improve in future?



Main Menu

7 8 9 10 11

Back

PROGRAMMING TASK 4 – GAMING AI?

Langtree Computing

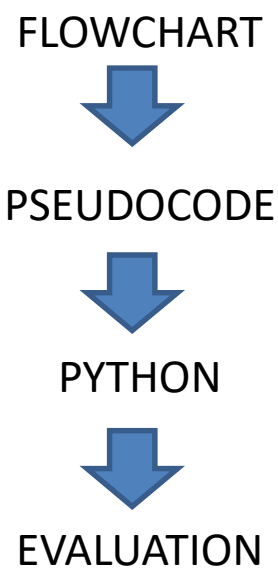
Write a program that plays Rock, Paper, Scissors against a human.
EXTENSION – Make the computer better at the game than a human (!) (HINT: Try to exploit that humans are very bad at generating random numbers)

1) Complete this task as a flowchart and then add the flowchart to a powerpoint as a picture or screenshot.

2) Next pseudocode the program. Screenshot this and add it to your powerpoint.

3) Now write the program using python. Screenshot the final outcome showing it works.

4) Write a brief evaluation. What has worked well? What could you change to improve in future?



Main Menu

7 8 9 10 11

Back

PROGRAMMING TASK 5 – MASTERMIND

Langtree Computing

This is a two player game. The first player chooses a secret 4-digit code (like 1492). The second player makes several attempts to guess this code. He or she can offer any combination of 4 digits. The first player should then give a hint. The hint consists of two values:

- *The first tells how many digits are guessed correctly and are in correct positions;*
- *The second tells how many digits are guessed correctly but are in wrong positions.*

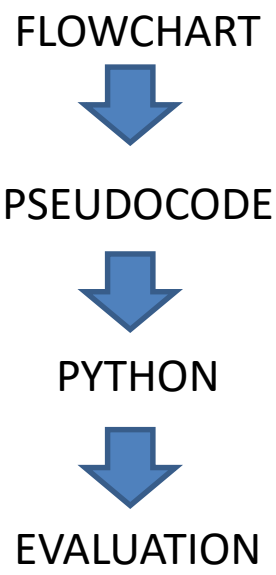
For example, if the secret value is 1492 and the Player 2's guess is 2013 – Player 1 should answer with 0-2.
If the guess is 1865, then the hint would be 1-0.
Write a program which lets you play this game with the computer.

1) Complete this task as a flowchart and then add the flowchart to a powerpoint as a picture or screenshot.

2) Next pseudocode the program. Screenshot this and add it to your powerpoint.

3) Now write the program using python. Screenshot the final outcome showing it works.

4) Write a brief evaluation. What has worked well? What could you change to improve in future?



Main Menu

7 8 9 10 11

Back

PROGRAMMING TASK EXTENSION 1 – CASTLE GAME

Langtree Computing

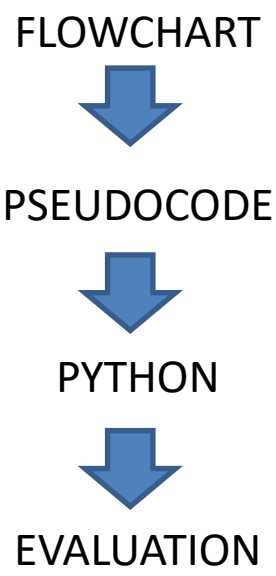
Write a program to play a simple "adventure"-style interactive game. The adventure world consists of up to five castles each of which has up to seven rooms (35 total). Each room has a treasure, worth a certain number of points, and a creature guarding the treasure. The treasure can be captured by bluffing or fighting the creature. Bluffing always has a 30% chance of succeeding. The odds for winning a fight vary from creature to creature. The object of the game is to visit different rooms and gain as many treasure points as possible. The player begins with 9 lives and each fight lost costs a life. (There's no penalty for losing a bluff.) When all the lives (or all treasures) are gone the game ends. The adventure world information (like castle and room names) is stored in a text file. The program must read the text file and create a data structure to represent the world. The program must handle interaction with the player, including display of menus for castle and room choices, display of current lives and treasure points accumulated, and responding to one-character commands to fight, bluff, or move around the world.

1) Complete this task as a flowchart and then add the flowchart to a powerpoint as a picture or screenshot.

2) Next pseudocode the program. Screenshot this and add it to your powerpoint.

3) Now write the program using python. Screenshot the final outcome showing it works.

4) Write a brief evaluation. What has worked well? What could you change to improve in future?



Main Menu

7 8 9 10 11

Back

PROGRAMMING TASK EXTENSION 2 – LAZY WRITER

Langtree Computing

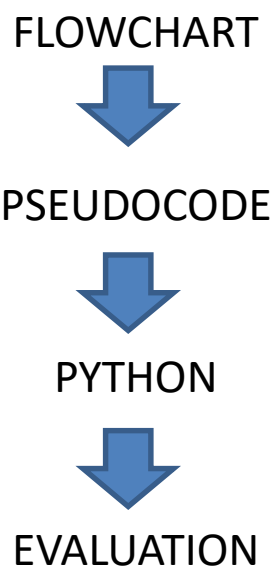
Write a program that automatically writes essays for you.
The user should be prompted to give some key words to include in the essay. The final essay does not have to make sense but should contain a range of sentences, which are different lengths, with a capitol letter at the start and a full stop at the end.

1) Complete this task as a flowchart and then add the flowchart to a powerpoint as a picture or screenshot.

2) Next pseudocode the program. Screenshot this and add it to your powerpoint.

3) Now write the program using python. Screenshot the final outcome showing it works.

4) Write a brief evaluation. What has worked well? What could you change to improve in future?



Main Menu

7 8 9 10 11

Back

PROGRAMMING TASK EXTENSION 3 – TETRIS

Langtree Computing

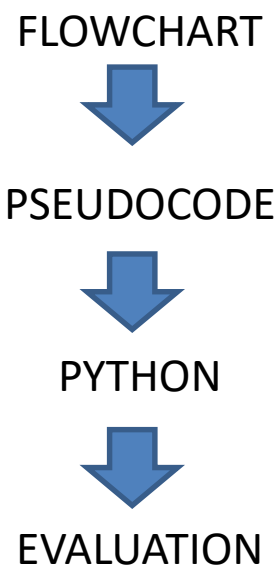
Create a tetris clone program.
[IF YOU HAVEN'T BEEN TAUGHT PYGAME] - Try to manage it by printing "X" where there is a block.

1) Complete this task as a flowchart and then add the flowchart to a powerpoint as a picture or screenshot.

2) Next pseudocode the program. Screenshot this and add it to your powerpoint.

3) Now write the program using python. Screenshot the final outcome showing it works.

4) Write a brief evaluation. What has worked well? What could you change to improve in future?



Main Menu

7 8 9 10 11

Back

PROGRAMMING TASK EXTENSION 4 – JOSEPHUS PROBLEM

About 2000 years ago there was a war, and during one of its battles some defendants were blocked by attackers in the cave.
To avoid capture they decided to stand in a circle and kill each third person until only one person remained - who was then supposed to commit suicide - though he actually surrendered to his enemies. The problem is named after this person - you may read the full story of Josephus and get a maths explanation of the problem in a Wikipedia article called "Josephus Problem"
Your task is to determine for given number of people N and constant step K the position of a person who remains the last - i.e. the safe position.
This description is taken from: http://www.codeabbey.com/index/task_view/josephus-problem where there is an example.

1) Complete this task as a flowchart and then add the flowchart to a powerpoint as a picture or screenshot.

2) Next pseudocode the program. Screenshot this and add it to your powerpoint.

3) Now write the program using python. Screenshot the final outcome showing it works.

4) Write a brief evaluation. What has worked well? What could you change to improve in future?

FLOWCHART

PSEUDOCODE

PYTHON

EVALUATION

Main Menu

7 8 9 10 11

Back

Langtree Computing

National Curriculum Section

- Pages after this contain some of the learning objectives taken from the New National Curriculum and GCSE Computing Schemes of Work.

Main Menu

7 8 9 10 11

Back

3.1.1 Constants, variables and data types

Students should:

- understand what is meant by the terms data and information
- be able to describe the difference between a constant and a variable
- understand when to use constants and variables in problem solving scenarios
- understand the different data types available to them. As a minimum, students should know about integer, Boolean, real, character and string data types and how these are represented in the programming language(s) they are using
- be able to explain the purpose of data types within code
- understand and be able to program with 1 and 2 dimensional arrays
- be able to use NOT, AND and OR when creating Boolean expressions and have experience in using these operators within coded solutions.

3.1.2 Structures

Students should:

- be able to explain what a data structure is
- be able to produce their own data types that go beyond the built in structures of the language(s) they are using, such as arrays or lists. These could include, for example, records in Delphi, structs in C or classes in Python and Java. The actual structures would depend on the language(s) being used by the students
- understand and be able to explain why data structures can make coding a solution simpler.

3.1.3 Program flow control

Students should:

- understand the need for structure when designing coded solutions to problems
- understand how problems can be broken down into smaller problems and how these steps can be represented by the use of devices such as flowcharts and structure diagrams
- understand and be able to describe the basic building blocks of coded solutions (ie sequencing, selection and iteration)
- know when to use the different flow control blocks (ie sequencing, selection and iteration) to solve a problem.

3.1.4 Procedures and functions

Students should:

- understand what procedures and functions are in programming terms
- know when the use of a procedure or function would make sense and would simplify the coded solution
- know how to write and use their own simple procedures and functions
- know about and be able to describe common built in functions in their chosen language(s)
- use common built-in functions in their chosen language(s) when coding solutions to problems
- understand what a parameter is when working with procedures and functions
- know how to use parameters when creating efficient solutions to problems
- understand the concepts of parameters and return values when working with procedures and functions.

3.1.5 Scope of variables, constants, functions and procedures

Students should:

- know what is meant by the scope of a variable, constant, function or procedure
- be able to identify what value a particular variable will hold at a given point in the code.

3.1.6 Error handling

Students should:

- be able to discuss and identify the different types of errors that can occur within code (ie syntax, run-time and logical)
- understand that some errors can be detected and corrected during the coding stage
- understand that some errors will occur during the execution of the code
- know how to detect errors at execution time and how to handle those errors to prevent the program from crashing where desirable
- be able to use trace tables to check their code for errors
- understand that computer programs can be developed with tools to help the programmer detect and deal with errors (eg Watch, Breakpoint, Step).

3.1.7 Handling external data

Students should:

- know how to use an external text file to read and write data in a way that is appropriate for the programming language(s) used and the problem being solved
- know how to read and write data from an external database in a way that is appropriate for the programming language(s) used and the problem being solved.

3.1.8 Computer Structure

Systems

Hardware

CPU (Central Processing Unit)

Memory

Secondary storage

3.1.8.1 Systems

Students should:

- be able to define a computer system (ie hardware and software working together to create a working solution)
- understand and be able to discuss the importance of computer systems to the modern world
- understand that computer systems must be reliable and robust and be able to discuss the reasons why this is important.

3.1.8.2 Hardware

Students should:

- be able to describe and explain the fundamental pieces of hardware required to make a functioning computer system
- be able to discuss how developments in different hardware technologies (including memory and processor) are leading to exciting innovative products being created, eg in the mobile and gaming industries
- be able to categorise devices as input or output depending on their function.

3.1.8.3 CPU (Central Processing Unit)

Students should:

- be able to describe the purpose of the processor (CPU)
- understand how different components link to a processor (ROM, RAM, I/O, storage, etc)
- be able to explain the effect of common CPU characteristics on the performance of the processor. These should include clock speed, number of cores and cache size/types.

3.1.8.4 Memory

Students should:

- know the differences between non-volatile and volatile memory
- understand the purpose of both types of memory and when each should be used
- be able to explain the purpose of virtual memory and cache memory
- be able to explain the concept that data and instructions are stored in memory and processed by the CPU.

3.1.8.5 Secondary storage

Students should:

- understand what secondary storage is and be able to explain why it is required
- be able to describe the most common types of secondary storage
- understand how optical media, magnetic media and solid state work.

3.1.9 Algorithms

Students should:

- understand that algorithms are computational solutions that always finish and return an answer
- be able to interpret simple algorithms to deduce their function
- be able to create algorithms to solve simple problems
- be able to detect and correct errors in simple algorithms.

3.1.10 Data representation

Students should:

- understand that computers use the binary alphabet to represent all data and instructions
- understand the terms bit, nibble, byte, kilobyte, megabyte, gigabyte and terabyte
- understand that a binary code could represent different types of data such as text, image, sound, integer, date, real number
- understand how binary can be used to represent positive whole numbers (up to 255)
- understand how sound and bitmap images can be represented in binary
- understand how characters are represented in binary and be familiar with ASCII and its limitations
- understand why hexadecimal number representation is often used and know how to convert between binary, denary and hexadecimal.

3.1.11 Software development life cycle

Students should:

- understand the software development life cycle
- be able to explain what commonly occurs at each stage of the software development life cycle
- be able to identify at which stage of the software development life cycle a given step would occur
- understand that there are several lifecycle models that can be used (eg cyclical, waterfall, spiral)
- be able to discuss the advantages and disadvantages of these lifecycle models.

3.1.11.1 Prototyping

Students should:

- understand what prototyping is
- be able to discuss the advantages and disadvantages of using prototyping when developing solutions
- have experience of using prototyping to create solutions to simple problems.

3.1.12 Application testing

Students should:

- understand the need for rigorous testing of coded solutions
- understand the different types of tests that can be used, including unit/modular testing
- be able to create suitable test plans and carry out suitable testing to demonstrate their solutions work as intended
- be able to hand test simple code designs/algorithms using trace tables.

3.1.13 Networking

Students should:

- understand what a computer network is
- be able to discuss the advantages and disadvantages of using a computer network
- be able to describe and explain the bus, ring and star networking topologies
- be able to discuss the advantages and disadvantages of each of these topologies.

3.1.13.1 Client server

Students should:

- understand the client-server model
- be able to explain, in simple terms, the handshake process used in most modern networking protocols
- be able to explain how coding for a client-server model is different from coding for a stand-alone application.

3.1.13.2 Web application concepts

Students should:

- understand the concept of coding at the server and client end
- know what can be coded at the server end
- know what can be coded at the client end
- have experience of coding solutions to simple web application problems.

3.1.14 Use of external code sources

Students should:

- know of the existence of external code sources
- know how to integrate code from these sources into their own code
- be able to explain the advantages and disadvantages of using such sources.

3.1.15 Database concepts

Students should:

- understand the basic concepts of a relational database as a data store
- be able to explain the terms record, field, table, query, primary key, relationship, index and search criteria.

3.1.15.1 Query methods (SQL)

Students should:

- be able to create simple SQL statements to extract, add and edit data stored in databases
- have experience of using these SQL statements from within their own coded systems.

3.1.15.2 Connecting to databases from applications and web based apps

Students should:

- be able to use databases from within their own web based applications.

3.1.16 The use of computer technology in society

Students should:

- be able to evaluate the effectiveness of computer programs/solutions
- be able to evaluate the impact of and issues related to the use of computer technology in society.

Algorithms

KEY STAGE 1
Algorithms are sets of instructions for achieving goals, made up of pre-defined steps *the 'how to' part of a recipe for a cake+.
Algorithms can be represented in simple formats [storyboards and narrative text].
They can describe everyday activities and can be followed by humans and by computers.
Computers need more precise instructions than humans do.
Steps can be repeated and some steps can be made up of smaller steps.

KEY STAGE 2
Algorithms can be represented symbolically [flowcharts] or using instructions in a clearly defined language [turtle graphics].
Algorithms can include selection (if) and repetition (loops).
Algorithms may be decomposed into component parts (procedures), each of which itself contains an algorithm.
Algorithms should be stated without ambiguity and care and precision are necessary to avoid errors.
Algorithms are developed according to a plan and then tested. Algorithms are corrected if they fail these tests.
It can be easier to plan, test and correct parts of an algorithm separately.

KEY STAGE 3
An algorithm is a sequence of precise steps to solve a given problem.
A single problem may be solved by several different algorithms.

The choice of an algorithm to solve a problem is driven by what is required of the solution [such as code complexity, speed, amount of memory used, amount of data, the data source and the outputs required].
The need for accuracy of both algorithm and data [difficulty of data verification; garbage in, garbage out]

Algorithms – Continued…

KEY STAGE 4
The choice of an algorithm should be influenced by the data structure and data values that need to be manipulated.
Familiarity with several key algorithms [sorting and searching].
The design of algorithms includes the ability to easily re-author, validate, test and correct the resulting code.
Different algorithms may have different performance characteristics for the same task.

EXTENTION
Modular arithmetic
Hashing
Distributed algorithms
Optimisation algorithms and heuristics; “good enough” solutions *genetic algorithms, simulated annealing];
Monte Carlo methods
Learning systems [matchbox computer]
Biologically inspired computing; artificial neural networks, Cellular automata, Emergent behaviour
Graphics [rotating a 3D model]

Programs

KEY STAGE 1

Computers (understood here to include all devices controlled by a processor, thus including programmable toys, phones, game consoles and PCs) are controlled by sequences of instructions.

A computer program is like the narrative part of a story, and the computer's job is to do what the narrator says. Computers have no intelligence, and so follow the narrator's instructions blindly.

Particular tasks can be accomplished by creating a program for a computer. Some computers allow their users to create their own programs.

Computers typically accept inputs, follow a stored sequence of instructions and produce outputs.

Programs can include repeated instructions.

KEY STAGE 2

A computer program is a sequence of instructions written to perform a specified task with a computer.

The idea of a program as a sequence of *statements* written in a programming language [Scratch]

One or more mechanisms for *selecting* which statement sequence will be executed, based upon the value of some data item

One or more mechanisms for *repeating* the execution of a sequence of statements, and using the value of some data item to control the number of times the sequence is repeated

Programs can model and simulate environments to answer "What if" questions.

Programs can be created using visual tools. Programs can work with different types of data. They can use a variety of control structures [selections and procedures].

Programs are unambiguous and that care and precision is necessary to avoid errors.

Programs are developed according to a plan and then tested. Programs are corrected if they fail these tests.

The behaviour of a program should be planned.

A well-written program tells a reader the story of how it works, both in the code and in human-readable comments

A web page is an HTML script that constructs the visual appearance. It is also the carrier for other code that can be processed by the browser.

Computers can be programmed so they appear to respond 'intelligently' to certain inputs.

Programs – Continued…

KEY STAGE 3

Programming is a problem-solving activity, and there are typically many different programs that can solve the same problem.
Variables and assignment.
Programs can work with different types of data [integers, characters, strings].
The use of relational operators and logic to control which program statements are executed, and in what order
o Simple use of AND, OR and NOT
o How relational operators are affected by negation *e.g. NOT (a>b) = a≤b+.
Abstraction by using functions and procedures (definition and call), including:
o Functions and procedures with parameters.
o Programs with more than one call of a single procedure.
Documenting programs to explain how they work.
Understanding the difference between errors in program syntax and errors in meaning. Finding and correcting both kinds of errors.

KEY STAGE 4

Manipulation of logical expressions, e.g. truth tables and Boolean valued variables.
Two-dimensional arrays (and higher).
Programming in a low level language.
Procedures that call procedures, to multiple levels. [Building one abstraction on top of another.]
Programs that read and write persistent data in files.
Programs are developed to meet a specification, and are corrected if they do not meet the specification.
Documenting programs helps explain how they work.

EXTENSION

Implementing recursive algorithms
Programming for the real world
Robotics
Other language types and constructs: object oriented and functional languages
App development
Developing for different environments
Programming using SDKs and other hardware
open source

Main Menu
7 8 9 10 11
Back

Data

KEY STAGE 1
Information can be stored and communicated in a variety of forms e.g. numbers, text, sound, image, video.
Computers use binary switches (on/off) to store information.
Binary (yes/no) answers can directly provide useful information (e.g. present or absent), and be used for decision.

KEY STAGE 2
Similar information can be represented in multiple.
Introduction to binary representation [representing names, objects or ideas as sequences of 0s and 1s].
The difference between constants and variables in programs.
Difference between data and information.
Structured data can be stored in tables with rows and columns. Data in tables can be sorted. Tables can be searched to answer questions.
Searches can use one or more columns of the table.
Data may contain errors and that this affects the search results and decisions based on the data. Errors may be reduced using verification and validation.
Personal information should be accurate, stored securely, used for limited purposes and treated with respect.

Main Menu
7 8 9 10 11
Back

Data – Continued…

KEY STAGE 3
Introduction to binary manipulation.
Representations of:
o Unsigned integers
o Text. [Key point: each character is represented by a bit pattern. Meaning is by convention only. Examples: Morse code, ASCII.]
o Sounds [both involving analogue to digital conversion, e.g. WAV, and free of such conversion, e.g. MIDI]
o Pictures [e.g. bitmap] and video.
Many different things may share the same representation, or "the meaning of a bit pattern is in the eye of the beholder" *e.g. the same bits could be interpreted as a BMP file or a spreadsheet file; an 8-bit value could be interpreted as a character or as a number].
The things that we perceive in the human world are not the same as what computers manipulate, and translation in both directions is required [e.g. how sound waves are converted into an MP3 file, and vice versa]
There are many different ways of representing a single thing in a computer. [For example, a song could be represented as:
o A scanned image of the musical score, held as pixels
o A MIDI file of the notes
o A WAV or MP3 file of a performance]
Different representations suit different purposes [e.g. searching, editing, size, fidelity].

KEY STAGE 4
Hexadecimal
Two's complement signed integers
String manipulation
Data compression; lossless and lossy compression algorithms (example JPEG)
Problems of using discrete binary representations:
o Quantization: digital representations cannot represent analogue signals with complete accuracy [e.g. a grey-scale picture may have 16, or 256, or more levels of grey, but always a finite number of discrete steps]
o Sampling frequency: digital representations cannot represent continuous space or time [e.g. a picture is represented using pixels, more or fewer, but never continuous]
o Representing fractional numbers

Data – Continued…

EXTENSION
List graphs and trees including binary tree traversals
Pointers and dynamic data structures
Handling very large data sets
Handling dynamic data sets (especially internet-based data)
Floating point representation

Computers

KEY STAGE 1
Computers are electronic devices using stored sequences of instructions.
Computers typically accept input and produce outputs, with examples of each in the context of PCs.
Many devices now contain computers

KEY STAGE 2
Computers are devices for executing programs.
Application software is a computer program designed to perform user tasks.
The operating system is a software that manages the relationship between the application software and the hardware
Computers consist of a number of hardware components each with a specific role [e.g. CPU, Memory, Hard disk, mouse, monitor].
Both the operating system and application software store data (e.g. in memory and a file system)

The above applies to devices with embedded computers (e.g. digital cameras), handheld technology (e.g. smart phones) and personal computers.
A variety of operating systems and application software is typically available for the same hardware.
Users can prevent or fix problems that occur with computers (e.g. connecting hardware, protection against viruses)
Social and ethical issues raised by the role of computers in our lives.

KEY STAGE 3
Computers are devices for executing programs
Computers are general-purpose devices (can be made to do many different things)
Not every computer is obviously a computer (most electronic devices contain computational devices)
Basic architecture: CPU, storage (e.g. hard disk, main memory), input/output (e.g. mouse, keyboard)
Computers are very fast, and getting faster all the time (Moore's law)
Computers can 'pretend' to do more than one thing at a time, by switching between different things very quickly

Main Menu
7 8 9 10 11
Back

Computers – Continued…

KEY STAGE 4
Logic gates: AND/OR/NOT. Circuits that add. Flip-flops, registers (**).
Von Neumann architecture: CPU, memory, addressing, the fetch-execute cycle and low-level instruction sets. Assembly code. [LittleMan]
Compilers and interpreters (what they are; not how to build them).
Operating systems (control which programs run, and provide the filing system) and virtual machines.

EXTENSION
Interrupts and real-time systems
Multiprocessor systems
Memory caches
Undecidability problems

Communication and the Internet

KEY STAGE 1
That the World Wide Web contains a very large amount of information.
Web browser is a program used to use view pages.
Each website has a unique name.
Enter a website address to view a specific website and navigate between pages and sites using the hyperlinks.

KEY STAGE 2
The Internet is a collection of computers connected together sharing the same way of communicating. The internet is not the web, and the web is not the internet.

These connections can be made using a range of technologies (e.g. network cables, telephone lines, wifi, mobile signals, carrier pigeons)
The Internet supports multiple services (e.g. the Web, e-mail, VoIP)
The relationship between web servers, web browsers, websites and web pages.
The format of URLs.
The role of search engines in allowing users to find specific web pages and a basic understanding of how results may be ranked.
Issues of safety and security from a technical perspective.

Communication and the Internet – Continued…

KEY STAGE 3
A network is a collection of computers working together
An end-to-end understanding of what happens when a user requests a web page in a browser, including:
o Browser and server exchange messages over the network
o What is in the messages [http request, and HTML]
o The structure of a web page - HTML, style sheets, hyperlinking to resources
o What the server does [fetch the file and send it back]
o What the browser does [interpret the file, fetch others, and display the lot]
How data is transported on the Internet
o Packets and packet switching
o Simple protocols: an agreed language for two computers to talk to each other. [Radio protocols "over", "out"; ack/nack; ethernet protocol: first use of shared medium, with backoff.]
How search engines work and how to search effectively. Advanced search queries with Boolean operators.

KEY STAGE 4
Client/server model.
MAC address, IP address, Domain Name service, cookies.
Some "real" protocol. *Example: use telnet to interact with an HTTP server.]
Routing
Deadlock and livelock
Redundancy and error correction
Encryption and security

EXTENSION
Asymmetric encryption; key exchange
Human Computer Interaction (HCI)
Recognition of the importance of the user interface; computers interact with *people*.
Simple user-interface design guidelines

Main Menu
7 8 9 10 11
Back