Introduction to programming using MicroPython

Overview

This is a set of 3 courses that have been designed to be done in isolation or back to back. The resources are provided in open formats and can be adapted.

It is assumed learners will have access to a PC (with internet connectivity), micro:bit and a USB cable for the practical tasks.

Resources

Some of the projects involve 'making materials' and these will typically consist of:

- Coloured paper/card
- Corrugated card
- Glue
- Sellotape/fabric tape
- Velcro tape
- Pens, pencils, coloured pens/felt tips
- Assorted google eyes
- Scissors

Aims and outcomes

The aim of this course is to introduce learners to the key STEM topics, physical computing, sensors, programming and IOT through project based learning. Learners will be immersed in a series of projects which involves rapid team based development of a solution to meet a criteria. These sprints of project based work are interspersed with theory lessons that cover the core concepts necessary for the following projects.

Key features

- Team project-based learning
- Design ideation, development, evaluation and refinement
- Authentic STEM based contexts
- Hands on physical Computing and Engineering challenges
- IOT themes
- Entrepreneurship

Course structure

This course is split into 3 sections, 2 x 6 lesson units designed to fit into a half term and a 12 lesson pack to fit into a whole term. The courses are designed to run concurrently as follows:

- Intro to Computing with micro:bit (lessons 1-6)
- Sensors and Radio Communication (lessons 7-12)
- IoT on micro:bit (lessons 13-24)

The courses can be used independently of each other should learners already have experience of programming micro:bits.

🟦	This blue shading denotes a project lesson where learners should work in groups of 3-4 to achieve the success criteria collaboratively.

Healthy Eating

	Topics	**Resources**
1	Introduction to the course Getting started with micro:bit and micro python Mu – what is an IDE? Scrolling text “Hello world” Displaying built in images	Sample python files
2	Importing Creating your own images Variables Lists Functions as commands (arguments)	Access to PC and Mu editor Sample python files
3	Event handling Buttons / how they work Events Selection Event handling (interrupts)	Example Python files
4	Variables Strings Storing data in variables Selection Loops	Sample Python code Crocodile clips Tin foil Sticky tape
5	Project 1 Healthy Eating – intro and build (PBL)	Making material
6	Project 1 Healthy Eating – build and testing (PBL)	

Sound Sensors and Crypto

	Topics	**Resources**
7	Making music Speaker hardware – how a speaker works Setting up the speaker Modules Lists	Crocodile clips Speakers
8	Speech Poetry/rapper auto generator	Crocodile clips Speakers
9	Random and it uses True random? Random numbers Crypto and uses in real world	
10	Gestures and movement	Headphones (optional)
11	Direction (PBL) Compass	
12	Accelerometer (12) x, y, z, axis Taking a reading Responding to a reading Crash detector	Sticky tape

Micro PET

	Topics	Resources
13	Storage and files	
14	Machine related code The kernel	
15	Temperature sensor and readings	A thermometer
16	Light level readings	Black paper or card
17	LEDs and classes	Crocodile clips and LEDs
18	Radio 1 Channels	
19	Radio 2 Sending and receiving messages via the micro:bit	
20	Pin use and making touch buttons	Kitronik edge connector Male to female jumper wires Tin foil Sticky tape
21	Servo motor	A servo motor Crocodile clips
22	Servo arm movements	A servo motor Crocodile clips
23	micro:PET introduction, planning and design	Making material Micro:pet net Design sheets
24	micro:PET speech and interactions	

Micro:Python for micro:bit National Curriculum Mapping

National Curriculum Programme of Study	Lesson(s) Covered
design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems	Healthy Eating: 3, 4, 5, 6 Sound, Sensors and Crypto: Micro:PET: 17, 21, 22, 23, 24
understand several key algorithms that reflect computational thinking [for example, ones for sorting and searching]; use logical reasoning to compare the utility of alternative algorithms for the same problem	Healthy Eating: Sound, Sensors and Crypto: Micro:PET:
use 2 or more programming languages, at least one of which is textual, to solve a variety of computational problems; make appropriate use of data structures [for example, lists, tables or arrays]; design and develop modular programs that use procedures or functions	Healthy Eating: 1, 2, 3, 4, 5, 6 Sound, Sensors and Crypto: 7, 8, 9, 10, 11, 12 Micro:PET: 13, 14, 15, 16, 17, 18, 19, 21, 22, 23, 24
understand simple Boolean logic [for example, AND, OR and NOT] and some of its uses in circuits and programming; understand how numbers can be represented in binary, and be able to carry out simple operations on binary numbers [for example, binary addition, and conversion between binary and decimal]	Healthy Eating: Sound, Sensors and Crypto: Micro:PET:
understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems	Healthy Eating: 4 Sound, Sensors and Crypto: 7, 10, 11, 12 Micro:PET: 14, 15, 16, 17, 18, 19, 21
understand how instructions are stored and executed within a computer system; understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally, in the form of binary digits	Healthy Eating: 2 Sound, Sensors and Crypto: 8 Micro:PET: 13
undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting and analysing data and meeting the needs of known users	Healthy Eating: 5, 6 Sound, Sensors and Crypto: Micro:PET: 20, 23, 24
create, reuse, revise and repurpose digital artefacts for a given audience, with attention to trustworthiness, design and usability	Healthy Eating: 5, 6 Sound, Sensors and Crypto: 8 Micro:PET:
understand a range of ways to use technology safely, respectfully, responsibly and securely, including protecting their online identity and privacy; recognise inappropriate content, contact and conduct, and know how to report concerns	Healthy Eating: Sound, Sensors and Crypto: 9 Micro:PET:

Pedagogy

This is a blended learning course where all the resources are hosted on a VLE. Learners can interact with the resources as and when they need them and this interaction with the platform is interspersed with formal delivery from the teacher where necessary. The course also includes many projects and employs a Project Based Learning (PBL) approach to delivering this content that builds on prior learning, challenges learners and encourages collaborative ideation and creation to solve authentic problems. The approach is outlined below:

- Explore and define
  - Abstraction
- Imagine and identify processes
  - Imagine
  - Generalisation/pattern recognition
  - Decomposition
  - Algorithm
- Do and review
  - Collaboration
  - Perseverance
  - Fail early, fail often (FEFO)
- Evaluation, iteration and celebration
  - Generalisation/pattern recognition

The projects are designed to require minimum directed teaching but a "Lesson flow" is provided to give the sessions some structure and to introduce some of the more challenging topics and concepts. As the teacher is doing minimal delivery of the content they are free to circulate amongst the learners and troubleshoot where needed, this allows weaker learners to be better supported and more able learners to be pushed harder when appropriate. There are stretch activities in every project that extend the knowledge and application of the skills for these learners.

Whilst circulating amongst learners teachers should be mindful of the "Key concepts" and "Key words" and should quiz learners at appropriate times on their knowledge of the concepts by getting them to explain the code/blocks they are using and also the thinking behind their designs for the making activities. Learners also may need reminding of the success criteria.

To aid the teacher in delivering the lessons, detailed plans are provided that include:

- The big picture – a brief overview of why this lesson is relevant
- Learning objectives
- How to engage learners
- Assessment for learning – expected, good and exceptional progress statements
- The key concepts of the lesson
- Key words – to be used for questioning during the lesson
- Differentiation advice
- Resources – what hardware and software are required
- Lesson flow – this is the sequence of events for the lesson
- Making – a description of any practical activities
- Questions – some questions to interrogate learners understand of the core concepts

A lesson plan template is in Appendix 1.

Making

The making element of all these projects is what makes these engaging STEM activities as they draw together learning about computer science, maths and engineering (designing and making) throughout the activities. Whilst the making element allows for effective differentiation and group work all learners should be given the opportunity to both code and make as both activities are rewarding. The design and iterative improvement of the physical parts of the projects represent the engineering element and learners should be encouraged to design, prototype and refine their work. The designs can be presented in many ways but should show the product, describe its purpose and function and how it has designed to suite its use with the project. It is vital that learners are made to design the physical products before starting to make them. Once the product is made it should be tested, evaluated and then improved.

Fail early, fail often

STEM is a mix of different disciplines and learners may initially struggle with the content. This is something to be celebrated as failure is an opportunity to learn and to improve. This mindset of failing early and failing often breeds resilience in learners and is an important mindset to cultivate in the STEM field. When learners are stuck they should be encouraged to seek assistance from the project resources, then the internet, then their peers and finally their teacher.

Test everything

Testing is an important aspect of iterative design. This is true of programs and of physical products. Learners should be encouraged to constantly test their products and improve them in an iterative fashion.

Evaluation and Success criteria

Each project has a list of "Success criteria" at the beginning of the project and these are referred to throughout the project. It is important to emphasise the importance of this criteria as this keeps the learners focussed and forms a checklist of what needs to be achieved. This also allows their work to be evaluated against the criteria. This is important as a STEM skill as projects and products always have criteria that needs to be met to be successful.

Differentiation

There is always a vast spectrum of ability and pre-existing experience with Computing and theses courses have been designed to flexible to accommodate being used with different year groups with different and overlapping abilities. The stretch tasks will give more able learners more challenging tasks and additional projects can be utilised should some learners race ahead.

One element that can be used to increase complexity without modifying the tasks would be to provide the tasks and projects to learners using Python (https://python.microbit.org/v/1.1). This would be suitable for learners already familiar with Python and could also be used to stretch more able learners.

Teacher CPD

This type of delivery and assessment needs specific interventions. Teachers need to intervene only when necessary and should only do enough to allow the learners to progress independently. Teachers should use questioning to determine the motivation and purpose of the learner's actions and decisions and teachers should allow learners to make mistakes and then help them troubleshoot only if required. This methodology will be further explored in the teacher CPD course that will be developed alongside this course.

Assessment overview

This course includes two sets of 20 multiple choice questions (MCQs). These can be used as formative or summative assessment depending when used. The MCQs can be administered manually or via any platform that typically users the MCQ format.

Assessment objectives

AO1	Demonstrate knowledge and understanding of technology
AO2	Apply knowledge and understanding of technology
AO3	Analyse and evaluate problems
AO4	Demonstrate application of knowledge and understanding to solve problems

Grade scale and descriptors

This course is not formerly assessed but some schools may find the assessment rubric below useful to assess learner progress. This framework can also be easily adapted to fit into a pre-existing assessment schema.

This framework can either be used to assess learners using the provided assessments or can be applied synoptically through observation.

Pass	• Learners will have demonstrated limited knowledge and understanding of the concepts and principals involved in the course. • Learners will have applied the principals and concepts using some analytical and evaluative thinking and practice to solve a problem. • Learners will have demonstrated some ability to apply knowledge and understanding to solve problems. • Learners will have collaborated with their peers and demonstrated some communication and teamwork.
Intermediate	• Learners will have demonstrated mostly accurate knowledge and understanding of the concepts and principals involved in the course. • Learners will have appropriately applied the principals and concepts using analytical and evaluative thinking and practice to solve a range of problems. • Learners will have demonstrated their ability to apply knowledge and understanding to solve problems. • Learners will have collaborated reasonably successfully with their peers and demonstrated mostly effective communication as well as effective teamwork.
Higher	• Learners will have demonstrated relevant and comprehensive knowledge and understanding of the concepts and principals involved in the course. • Learners will have effectively applied the principals and concepts using sustained analytical and evaluative thinking and practice to solve a range of problems. • Learners will have successfully demonstrated their ability to apply knowledge and understanding to solve substantial problems in an efficient manner. • Learners will have collaborated successfully with their peers and demonstrated effective communication as well as efficient and effective teamwork.

Appendix 1 – Lesson plan template

Lesson plan example

The big picture – why is this relevant?	Learning objectives:
•	•
Engagement – How can I engage learners?	Assessment for learning
•	**Expected progress:** **Good progress:** **Exceptional progress:** •
Key concepts:	Key words:
•	•
Differentiation:	Resources:
	•
Lesson flow	
•	
Making	
•	

Appendix 2 – PRIMM: A scaffolded approach to teaching programming

PRIMM is a teaching methodology based on educational research that scaffolds programming activities to maximise engagement and understanding of the core programming concepts. The best approach to ensuring all learners fully understand and are able to apply their understand is to utilise a blended approach to delivery, combining group discussion, paired work, scaffolded practical tasks and creative, engaging tasks and challenges.

PRIMM stands for the following:

- Predict
- Run
- Investigate
- Modify
- Make

Scaffolding

Learners typically need a lot of support to understand programming concepts, especially at the beginning. There are many techniques and concepts that whilst individually are straight forward to understand and apply in a given context, applying the same technique in an unfamiliar context is challenging. This means that strategies are needed that scaffold the learning as well as encouraging discussion about what is occurring and why. The following teaching approaches can be employed to support learners when working with code.

Stage	**Activities**	**Resources**
Predict	Review code in pairs and predict the output Group discussion on larger examples of code Black box flow charts where learners predict what code is in the box Predict the purpose of a function based on the input/output	Pre prepared code examples
Run	Run any code examples provided	Pre prepared code examples
Investigate	Comment up some provided code Find a specific technique in the code and highlight it Trace tables Turn a flowchart into code and vice versa	Intentionally buggy code Pre prepared code examples
Modify	Bug fix some broken code Fix a logic error Improve an algorithm Add functionality to a program Modify a programme to be functional Re-factor an inefficient program	Pre prepared code examples
Make	Create a program to a brief Add functionality to a program Create a function based on arguments from other functions	Challenge briefs Pre prepared code examples