# Lesson plan - “Slider snake”

## Using the MonkMakes Slider for micro:bit component

## Introduction

The particular focus of the lesson is collections of data and specifically lists. The program is a really interesting use of lists and demonstrates to students how a programmer can use the index list position command to alter the items in a list. The logic of the program is quite tricky to follow so we recommend that you increase the sleep command at the end of the program to slow the program down so that the students can observe what is happening to the LEDs as the slider is moved about. It is also important to note that if the students struggle with the logic of the whole program there is still a lot of benefit to understanding individual elements of the program.

## Learning objectives

* To successfully set up a micro:bit/slider snake circuit and flash code to the micro:bit
* Apply knowledge of a for loop
* To be able to read and interpret a program containing a list
* To be able to read and interpret a program containing user defined functions
* To be able to read and interpret a program containing a while loop
* To be able to read and interpret a program containing a for loop
* To be able to successfully modify a program containing multiple programming constructs

## Keywords

Ordered, changeable, duplicate,methods, index, set, dictionary, list, tuple, while, for, range, variable, function, in-built, user-defined.

## Preparation

**Subject knowledge:**

This lesson is suitable for a class who already have a basic knowledge of programming using python. The coding for the micro:bit is written in a reduced version\* of python called micropython but for practical purposes the syntax will appear the same to the user. In addition the students should have already used the micro:bit, practising simple set up and smaller micro:bit specific programs so that they are used to building and flashing programs. Initially, building the circuit for the “slider snake” is a question of copying a diagram so a practical demonstration of handling the components gently and clipping with alligator clips would suffice.

**Pedagogical approach:**

The lesson is planned using the PRIMM pedagogy which stands for:

P - Predict

R - Run

I - Investigate

M - Modify

M - Make

For this reason it is important that the class have quick and easy access to the coding of the program which can be accessed via: https://tinyurl.com/cvcvzr7y

The very first step is for the students to view the code and attempt to predict what the code does. We use a tool called a **code conversation** which provides the teacher with a conversation style **talkthrough** of the micropython code for you to support your students’ developing knowledge.

**Practical set-up and development environment:**

We recommend either using the on-line Python editor (https://python.microbit.org) or Mu (https://codewith.mu). Or you could manage your coding in micro:bit’s own environment <https://classroom.microbit.org/> which enables you to push code to the whole class and manage their activities.

**Resources that you need:**

* Presentation
* Starter worksheet questions
* Starter worksheet answers
* Code conversation full solution
* Code conversation template (level 1 - basic)
* Code conversation template (level 2 - intermediate)
* Code conversation template (level 3 - advanced)
* Plenary worksheet
* Plenary worksheet answers
* Hardware Per pair - 1 micro:bit, 1 USB/micro USB cable, 3 alligator clip leads, 1 slider
* Access to appropriate development environment (the on-line Python editor, Mu or micro:bit classroom - Mu is needed for the last lesson in the scheme)

## How the students’ progress is assessed

**Slider snake**

The presentation itself begins with a slide stating the difference between explicit and implicit type conversion followed by a worksheet checking if the students are already aware of the different data types in python. It is interspersed with pair or whole class discussion opportunities. As this is predominantly a practical lesson observing the students’ success in flashing the program, compiling the circuit and making small successful modifications to the program will also enable you to assess their progress. The learners also complete a worksheet at the end of the lesson which demonstrates their understanding of key principles conveyed during the lesson.

## Plan (with approximate timings)

|  |  |
| --- | --- |
| Starter activity 5 mins | **Lists revision**  The students are given a worksheet containing questions about lists. There are some prompts on the sheet for any students who need extra support with this topic. |
| Focus tasks 10 mins | **List consolidation**  In this section the students are given some explicit teaching about the use of lists. This includes the features of lists, some scaffolding of simple code for lists, further consolidation of index positions and examples of specific list methods. In addition there is a focus on list index and shuffling the values using a for x in range loop. This particular focus should really help with understanding of the snake program later. |
| PRIMM 8 mins | **Predict: students view code and attempt overview code conversation**  The students are shown the code and are asked to predict what the whole program does i.e. an overview in a couple of sentences.  For example: This program uses the Slider to make a wiggly snake animation when you move the slider. It takes a reading from the slider, puts it into a list and then lights up the LEDS as the slider is moved about. The scroll\_up function (which is running forever, due to the while loop) first shuffles all the dot positions up one place using a for loop. It then clears the display and then sets the position of the bottom most dot to be determined by the slider position. It then draws all the dots. The time.sleep(0.1) just  slow things down so you can see what is happening. |
| PRIMM 8 mins | **Run: students compile the circuit, flash and run the code**  Share the slide with instructions on the board to enable the students to build the circuit, flash and run the code. Ask them if the circuit behaves as they would have expected from the **predict** phase of the lesson. Encourage the students to move the slider up and down and see how the micro:bit LEDs respond. The slides for this section also reflect what happens at the beginning of the slider snake program, which is that the list is initially set to [2,2,2,2,2] with the right most element [4] being set to the reading from the slider. |
| PRIMM 13 mins | **Investigate: teacher and students use correct terminology to identify the syntax and features of the program**  Using the A4 worksheet, the learners attempt to complete the code conversation matching the correct explanation to each line of code.  There are three levels of the code conversation:  Level 1 - basic (some less challenging code lines to complete), Level 2 - intermediate (more challenging code lines to complete), Level 3 (all lines of code to complete)  Using the slides talk through the different steps in the program. |
| PRIMM 5 mins | **Modify: students modify the code to create a new program**  Students are invited to modify the program by first declaring what their new program should do, then by making the modification, saving the new program and testing it by flashing it to the micro:bit.  Suggestions for achievable alterations:-  Change the names of variables and functions  Reverse the motion of the snake  Get the micro:bit to produce an interesting pattern in response to the slider |
| **Plenary**  6 mins | **Slider snake plenary worksheet**  Students complete the plenary worksheet testing knowledge of lists. |
| **Homework** | **Consider what other modifications you could make to the program, still using the slider component** |

## The Author

This lesson plan and all its parts were created by Dr. Paula Beer of Beer Academic Consultancy in collaboration with Monk Makes Ltd.



Dr Paula Beer has taught Computer Science and IT education to new and established teachers since 2007. Her own research has focused on the use of play and collaboration in computer science. She enjoys supporting practicing teachers by designing accessible lesson planning materials to get students engaged in computer science through play and collaboration. Paula has also produced educational materials for The Raspberry Pi Foundation, been a secondary school teacher, written a successful book (Hello App Inventor!) and has previously worked in IT project management for a media company and for the NHS.