***Self-Directed Activities: Coding***

These self-directed activities will help you to learn some of the basics of coding, using Scratch, Kodu and Python.

**Activity 1 - Introduction to Computer Science**

Begin by watching the Hour of Code video:

<https://www.khanacademy.org/computing/hour-of-code/hour-of-drawing-code/v/welcome-hour-of-code>

After watching the video, continue using this site on your own to complete the coding challenges.

**Activity 2 - Introduction to Scratch**

Take a look at this interactive animation built in Scratch:

<https://scratch.mit.edu/projects/151267787/>

Looking “inside” the animation, observing how it is made up of code. Try altering a part of the code, noting how this affects the game.

Then make sure you have your own Scratch account.Try to create your own simple interactive animation using Scratch in which a sprite is controlled using the arrow keys. As an extension, program your sprite to respond to another sprite or colour in the background (e.g. perhaps the sprite can be programmed to disappear if it touches water).

**Activity 3 - Make a sprite and a background**

In this activity, you will be using Scratch to build a maze game. Take a look at this example: <https://scratch.mit.edu/projects/69535782/>



Note that the sprite is our character or any graphical object, which is controlled by the code. In this game, the main sprite is the mouse. Can you see any other sprites in the game?

Begin by making your main sprite and background. You will also need to make other sprites that their main sprite will interact with, e.g. a mousetrap, a door, etc.

Watch the following tutorial videos, which will help you to make your Scratch maze game:

[Part 1](https://www.youtube.com/watch?v=du4LJWxcrck)

[Part 2](https://www.youtube.com/watch?v=uX--NIhKr4w)

[Part 3](https://www.youtube.com/watch?v=4bUsug56OdQ)

**Activity 4 - Create a setting in Kodu**

Kodu is another application designed to help you learn programming skills.  As with Scratch, you can use this application to create a computer game.  In this activity, you will begin by designing a virtual world where your game will take place.

Open up Kodu and select New World.  Observe how you can add more land using the Ground brush tool and how the Up/Down tool can be used to create hills or valleys:





Try inserting objects like trees into your world by clicking first on the Object tool, and then once an object is selected, right clicking on the place where you want the object to appear.



**Activity 5 - Program Kodu to move and collect objects to score points**

In this activity, you will create a game that involves one character collecting objects in order to earn points.

Follow these programming steps:



Observe which code does what:

* Move the character using the arrow keys;
* Program the character to collect objects (e.g. an apple) and earn points each time an object is collected;
* When a certain number of points are earned, the game is won;
* Insert a timer, so that after a set time, the game ends.

Try making changes in this code, to see how it affects the game. Consider what you can do to the code to make your game more challenging.

**Activity 6 - Program enemies that can shoot**

One of the things that makes games more interesting and more challenging is to insert an enemy character, which shoots at the main character the user controls.  Have a go at following these programming steps:



Notice that when playing this game, it is very difficult because the enemy character (based on the programming above) shoots too fast.

Go into the settings in order to alter the blip reload rate by right clicking on the Kodu character and clicking on Change Settings:



As an extension, see if you can work out how to display the main character’s health and if you are able to display the countdown timer.

**Activity 7 - Complete Kodu Maze Tutorials**

From the Kodu homepage, click on Load World, where you can select additional challenges to complete, to develop your coding skills:

****

**Activity 7 - Use Python to create an Adventure Game**

For these remaining activities, you will use Python. The Python code language is used by millions of developers worldwide, including developers at organisations like NASA, Google and CERN. You will be using concepts such as using a sequence of instructions to make something happen, loops, conditionals and variables – concepts which are common throughout all programming languages.

Use a Text Editor to create a code file. To do this, open up the Python programming environment, IDLE. Then click File from the menu at the top and select New Window to open an untitled text file. Notice that this creates a new text editor file in which you can type all your code and save it, before you test that it works by running IDLE.

Save the file as Inventory. Then type the following:



Each piece of text data in quotation marks (called a **string**) represents an item on the inventory list.

Click Run, then Run Module. Your inventory list will be printed to the IDLE window.

Now adapt the last line of code to read:

print(inventory [3])

Observe what happens. Why do you think it printed Catapult and not Rubber band? Clue: it is to do with the way that Python numbers items in a list.

**Activity 8 - Use the Python time and random modules**

Python has a large number of **modules** – useful blocks of code that you can reuse to avoid having to rewrite the same code over and over each time you need a program that performs the same task.

In order to use a module within a Python program, you use the Python word *import* followed by the name of the module. As you write the code, you will include comments.

Comments are notes within your code that explain what a line or section of code is intended to do. Each comment line begins with a #symbol, which tells the computer running the program to ignore that line. Comments help you remember what each line of code is doing, and they help others see what you have already done.

With the Inventory.py file that you created earlier, type the following code:



Note: the \n at the end of the string for quest\_item input makes an extra new line below, which makes the text easier to read.

Challenge: As the code is written at the moment, when the user inputs an item from the inventory list, e.g. torch, the program will print that the item could not be found. Can you think of a way to improve the program so that it checks to see if the item is already in the list?

Solution to challenge from Activity 8:



**Activity 9 -** For this task you will create a Python Text Adventure Game that uses text to direct the player through the game. The program will ask the player to make decisions on what to do next.

As the text adventure game relies on the player (user) to interact with the game to make decisions, you will need to use the input() function. This line of code asks the player to answer a question, and the program will wait for the player to type an acceptable response – one that the program will understand.

The while True: condition will loop the question over and over until the Player enters either left or right so that the game does not end if the player types anything else.

Remember: Indentation refers to how far from the margin line a code is typed. Indentation of code in Python is very important, especially when you begin to add more structure to the code by using conditionals and loops.

Now that you have learned a little about the input function and while True condition, type the following code:

*Using conditionals in a Python adventure game*



Here, you have used conditionals in the Python adventure game. Now, to make sure that the player types in the correct lower case response that you require to meet the conditions, you can use a lower case function so that if the player types in capital letters, the program will turn the text into lower case, which is a recognised response to the if, elif, and else conditions.

direction1 = direction1.lower()

Add this line before the first if statement used, and after asking the player to input a direction.

*Adding* direction1 = direction1.lower()



*Using a while loop in the adventure game*



Now, put all the elements together of the text adventure game by typing the following program:



**Activity 10 -** At this stage you will do a process called refactoring - a way of restructuring code you have already written to make it more efficient and easy to read, and to avoid bugs. One way of doing this is to use two functions: get\_input() and handle\_room().

The get\_input function will keep on asking the payer to enter inputs until it matches one of the accepted inputs (in this case, the accepted inputs are left or right).

Up to this point, you wrote all of your game logic in the while loop. With the following code, you move most of the logic into separate functions, avoiding repetitive code. The new loop calls the handle\_room() function to perform a task appropriate for the current room, and then updates the location variable with the new room. This more advanced code will require you to check and double-check that your indentation is correct and you have not made any syntax errors.

Open a new text editor window and save as AdventureGame2.py. Follow the steps on the next page to add functions to your adventure game.

Note: Remember to indent your code correctly here, as three levels of indentation are being used.





Finally, test that you game works, checking for any bugs.