Year 5 Knowledge Organiser
Computing – Programming

What I should already know.

- Solve open ended problems with a floor robot, screen turtle and other programmable devices.
- Design, write and run executable programs using a programming language e.g. that used for a floor robot, Scratch, Kodu, Espresso Coding.
- Be able to debug an algorithm (set of instructions) and correct any errors.
- Use repetition in programs to make them more efficient. E.g. Rpt4[FD5 RT90] to draw a square with Roamer.
- Be able to explore the effect of changing variables. Use them to make and test predictions.
- Use 'selection' in a programming sequence i.e. use 'if… then… else…' type actions or statements e.g. if a character is touching a wall then bounce back, else move forward.

What will I know by the end of the unit?

- Predict how a provided algorithm will behave before testing it (e.g. write a program or procedure in symbols and ask pupils to 'write the story' of the outcome before testing it).
- Represent an algorithm symbolically (e.g. as a flow chart) to plan a procedure.
- Develop algorithms which include 'if' statements (e.g. if the temperature drops below...) and loops (e.g. repeat [an instruction] 4 times).
- Develop more complex flow diagrams and procedures that draw on others (e.g. program traffic lights either end of a narrow bridge so that cars don't collide).
- Refine procedures (algorithms) to improve efficiency and achieve desired outcomes.
- Create a program which includes a method of scoring (e.g. each time a sprite bumps into a particular object increase the score and each time it bumps into another object decrease the score).
- Create a program that requires a timer and set the variables as appropriate to the program (e.g. set a timer for a contestant to solve a maze within 30 seconds).

Key Vocabulary

Algorithm

Procedure

If

Loops

Variables

Constants

Refine

Online Safety

Be E-safe and enjoy!

Key Knowledge

- Understand that algorithms may be decomposed into component parts (procedures), each of which is itself an algorithm.
- Understand the need for precision when creating algorithms.
- Understand the importance of planning, testing and correcting algorithms.
- Understand that an input can be used to control the behaviour of a program.
- Explain logically, using appropriate technical language, how some simple algorithms work.
- Understand the difference between constants and variables.

Software