Subject: Computing **Year: 9** **Term: 2**

Title of Scheme of Learning:	Overview of topic:	How does this SOL relate to previous schemes?	Common Misconceptions in the unit:
Python Programming	Python Programming – This unit introduces pupils to a text-based programming language, building from output and calculations to user input, selection and iteration. They practise the core constructs through short programs, then apply them to an interactive multiple-choice quiz that awards points and restricts access using a passcode. Evidence is gathered in a programming e-portfolio. **Key Skills to be covered:** • Writing and running Python code using print() and arithmetic operators • Using named values, variables, input() and numeric type conversion • Creating decisions with if, elif, else and comparison operators • Using for and while loops, including Turtle shape-drawing tasks • Identifying and correcting syntax and logic errors • Creating a scored quiz with multiple-choice answers and passcode access • Testing programs, explaining code and recording screenshots in an e-portfolio **SMSC/PSHE/CV/Catholic Social Teaching Overview:** Develop logical thinking, resilience and confidence through debugging (S). Give constructive feedback and recognise the value of different solutions (M/S). Consider accessible, inclusive software design and respectful collaboration (C/CST). Explore careers including software developer, games programmer and automation engineer (CV).	• Year 7 Scratch develops sequence, selection and iteration in a block-based environment. • Year 8 EduBlocks and TuringLab activities introduce programming constructs and computational thinking through visual tasks. • Year 9 SuperTeams video editing and Inkscape develop planning, testing, independent problem-solving and evaluation. **Next topic:** To be confirmed in the Year 9 sequence. **Links to Future Learning** • OCR GCSE Computer Science (J277): programming fundamentals, constructs, data types, testing and algorithms. • Further Python projects involving functions, lists, validation and more complex algorithms. • Programming for automation, data applications, software development and games. • Career pathways: software developer, games programmer, data analyst and cybersecurity professional.	Misconception: print() asks the user a question. Reality: print() displays output; input() receives user input. Misconception: input() returns a number automatically. Reality: Input arrives as text until converted, e.g. with int() when appropriate. Misconception: An equals sign tests equality. Reality: = assigns a value; == compares values. Misconception: Indentation is optional in Python. Reality: Indentation defines the body of selection statements and loops. Misconception: A for loop and a while loop are interchangeable. Reality: for commonly iterates over a sequence or range; while repeats while a condition remains true. Misconception: A program that runs is always correct. Reality: Logic errors can give incorrect results without producing an error message.

Learning episodes (inc no. of lessons)	Learning Question(s) & Key Vocabulary	Content *(What should be covered and planned time for pupils to respond to feedback)*	Feedback & Assessment Points *(Including assessment methods and opportunities to respond to feedback)*
1	How can we use Python to display information and perform calculations? **Vocabulary** • Python • Output • print() • String • Arithmetic operator • Assignment	• RECALL: Identify arithmetic symbols and predict the result of simple expressions. • ACTIVATE: model print() for text and numbers, storing named values and using +, -, *, /, //, % and ** where appropriate. • DEVELOP: complete worksheet calculations and e-portfolio challenges, including rectangle area/perimeter, year of birth and mean average. • Exit Summary: predict output and explain the difference between text and a numeric result.	• Teacher questioning and short code-prediction checks during demonstration. • E-portfolio screenshots of functioning programs for the Week 1 challenges. • Peer or teacher feedback on output and calculations; allow 5 minutes to correct errors and retest.
2	How can Python collect and use information entered by a user? **Vocabulary** • Input • input() • Variable • String • Integer • int() • Casting	• RECALL: Match common data types to examples and revisit print(). • ACTIVATE: model input(), storing user responses, type conversion with int() and arithmetic using entered numbers. • DEVELOP: adapt calculator-style programs; complete e-portfolio challenges such as user-entered rectangle measurements, mean average and power/efficiency calculations. • Exit Summary: explain why two number inputs sometimes need converting before addition.	• Live test using two contrasting input values; pupils predict and explain the output. • E-portfolio evidence of input, conversion and tested results. • Mini debugging activity for a string-versus-integer error; allow 5 minutes to amend and rerun.
3	How can we use selection to make a program respond differently? **Vocabulary** • Selection • Condition • if • elif • else • Comparison • Indentation	• RECALL: interpret a decision flowchart controlling a fan. • ACTIVATE: model if/else and if/elif/else, including comparison operators (==, !=, >, <, >=, <=) and required indentation. • DEVELOP: adapt a number-comparison program and complete e-portfolio conditional tasks, including temperature, light level and score-to-grade decisions. • Exit Summary: identify which branch runs for a given input.	• Cold-call predictions of which branch executes and why. • E-portfolio screenshots from Week 3 selection challenges, tested with more than one input. • Teacher feedback on indentation and operator mistakes; 5 minutes to improve one program.

Learning episodes (inc no. of lessons)	Learning Question(s) & Key Vocabulary	Content *(What should be covered and planned time for pupils to respond to feedback)*	Feedback & Assessment Points *(Including assessment methods and opportunities to respond to feedback)*
4	How can loops make programs more efficient? **Vocabulary** • Iteration • for loop • while loop • Counter • Condition • Turtle	• RECALL: compare an inefficient repeated set of drawing instructions with a loop. • ACTIVATE: model a counter-controlled for loop and a condition-controlled while loop; show a loop drawing a repeated Turtle shape. • DEVELOP: draw polygons using loops, then extend to user-selected number of sides and shape selection. Record Week 4 evidence. • Exit Summary: explain when a for or while loop is appropriate and identify a potential endless loop.	• Code tracing and mini-checks about the number of iterations. • E-portfolio screenshots of working Turtle shapes and, where achieved, a shape chosen from user input. • Feedback on loop conditions and turning angles; 5–10 minutes to debug drawings.
5	How can we combine programming techniques into an interactive quiz? **Vocabulary** • Program • Quiz • Score • Passcode • Syntax error • Logic error • Debugging	• RECALL: identify deliberate syntax and logic errors in short examples. • ACTIVATE: examine the client requirements for a multiple-choice quiz with correct/incorrect answers, cumulative scoring and a passcode prompt. • DEVELOP: build and test the quiz using input/output, selection, variables and a while loop for the login; use the sample code as scaffolding if required. • Exit Summary: review whether all three project requirements have been met.	• Teacher checks the program against the stated project requirements. • E-portfolio evidence: first question and welcome message, last question and final score, and passcode/while-loop section. • Peer test the program using correct, incorrect and invalid responses; time to correct one defect.
6	How can we demonstrate and evaluate our Python knowledge? **Vocabulary** • Program • Sequence • Selection • Iteration • Testing • Evaluation	• RECALL: review core vocabulary and trace an example involving input, conditions or loops. • DEVELOP: complete the supplied end-of-unit Microsoft Forms assessment and finish outstanding portfolio evidence. • Check the quiz project against the assessment criteria for programming skills and application. • Exit Summary: identify one construct understood well and one improvement target.	• End-of-unit online knowledge assessment (provided in the source assessment folder. • Teacher assessment of e-portfolio challenges (Lessons 1–4) and practical quiz using source Level 1–4 criteria. • Individual feedback and time to correct or complete missing programming evidence.