

Autumn Term 1 – Python Revisited - Conditional Sequences and Functions

	Knowledge	Skills
Strong	- Clearly and accurately communicates how variables, inputs, outputs and data types interact within a Python program. - Clearly communicates the purpose and importance of casting when processing user input and data. - Clearly communicates how fStrings improve the clarity and maintainability of program output. - Clearly communicates how conditional sequences control program execution using IF, ELSE, ELIF and nested IF statements. - Clearly communicates how relational operators are used to evaluate conditions and support decision-making. - Clearly communicates how increasingly complex selection structures can be developed using nested conditions. - Clearly communicates how subprograms improve program structure, readability, maintainability and code reuse. - Clearly communicates the role of key programming terminology using appropriate technical language.	- Can independently design, develop and refine programs that combine variables, inputs, outputs, casting and conditional logic. - Can select and apply appropriate relational operators to create efficient decision-making processes. - Can develop complex conditional sequences using IF, ELSE, ELIF and nested IF statements. - Can create reusable subprograms that are called from multiple parts of a program. - Can structure code logically using meaningful variable names and appropriate formatting. - Can systematically test, debug and improve programs. - Can apply programming techniques to solve unfamiliar problems independently. - Can maintain comprehensive notes that explain programming concepts, solutions and coding decisions.
Confident	- Communicates how variables, inputs and outputs work together within a program. - Communicates the purpose of casting and how it enables calculations and comparisons. - Communicates how fStrings improve the readability and presentation of program output. - Communicates how conditional sequences use IF, ELSE and ELIF statements to control program flow.	- Can independently create programs using variables, inputs, outputs and casting. - Can use fStrings to produce clear and informative output. - Can create programs containing IF, ELSE and ELIF statements. - Can use a range of relational operators appropriately. - Can develop programs using nested IF statements. - Can create and use subprograms to organise code effectively. - Can test and debug programs to ensure they function correctly.

	- Communicates the purpose of relational operators within conditions. - Communicates how nested IF statements allow more complex decision-making. - Communicates how subprograms can be called from different parts of a program to improve organisation and reuse. - Communicates the meaning of key terminology including variable, condition, selection and relational operator.	Can maintain clear and organised programming notes.
Developing	- Identifies the purpose of variables, inputs and outputs in simple programs. - Describes the difference between strings and numbers. - Recognises that casting allows data to be changed from one type to another. - Describes how IF statements can be used to make decisions. - Recognises the purpose of ELSE and ELIF statements. - Describes what a condition is and identifies simple relational operators. - Recognises that subprograms can be used to organise code.	- Can create simple programs using variables, inputs and outputs. - Can use basic casting when shown appropriate examples. - Can create output using simple fStrings. - Can create programs using IF statements. - Can use simple relational operators such as ==, > and . - Can create and call simple subprograms with support. - Can complete structured programming tasks with guidance.
Emerging	- Recognises that programs can use variables to store information. - Recognises that users can type information into a program. - Recognises that programs can display information to a user. - Recognises that numbers and text are different types of data. - Recognises that programs can make decisions. - Recognises that programs can be broken into smaller sections called subprograms.	- Can run and modify simple Python programs with support. - Can create simple variables when given examples. - Can use basic input and output statements with guidance. - Can identify conditions in a program. - Can use a provided subprogram within a program. - Can record basic notes about programming concepts.

Autumn Term 2 – Python – Loops and Lists

	Knowledge	Skills
Strong	- Clearly and accurately communicates how different loop structures can be selected to solve different programming problems. - Clearly communicates the differences, advantages and applications of conditional while loops and while True loops. - Clearly communicates how the arguments within the range() function affect the behaviour of a for loop. - Clearly communicates how lists provide an efficient way of storing and managing collections of related data. - Clearly communicates how list functions support the manipulation and organisation of data. - Clearly communicates how iteration can be used to process, analyse and modify data stored within a list. - Clearly communicates how more complex data structures, such as two-dimensional lists, can be used to represent and search data.	- Can independently design, develop and refine programs that incorporate loops and lists effectively. - Can select the most appropriate loop structure for a given problem. - Can create efficient solutions using the full functionality of the range() function. - Can manipulate list data using a variety of methods and functions. - Can use iteration to process large collections of data efficiently. - Can develop solutions involving nested data structures such as two-dimensional lists. - Can implement and explain a simple linear search within a two-dimensional list. - Can systematically test, debug and improve programs. - Can apply programming techniques to solve unfamiliar problems independently. - Can maintain comprehensive notes that explain programming concepts, solutions and coding decisions.
Confident	- Communicates the differences between conditional while loops and while True loops. - Communicates how for loops use the range() function to control repetition. - Communicates the purpose of the start, stop and step arguments within the range() function. - Communicates that the stop value in a range is not included in the sequence. - Communicates how lists can be used to store and organise collections of data. - Communicates how list functions can be used to add, remove and manipulate data.	- Can independently create programs using both while loops and for loops. - Can use conditional while loops and while True loops appropriately. - Can use the start, stop and step arguments within the range() function. - Can create, edit and manipulate lists using a range of list functions. - Can access and modify individual items stored within a list. - Can iterate through a list to perform calculations, searches and other processing tasks. - Can test and debug programs to ensure they function correctly.

	- Communicates how items can be accessed using their position within a list. - Communicates how iteration can be used to process every item in a list.	Can maintain clear and organised programming notes.
Developing	- Describes the purpose of while loops and for loops. - Recognises that while loops repeat instructions while a condition remains true. - Identifies that for loops can repeat instructions a fixed number of times. - Describes a list as a collection of multiple values stored in a single variable. - Recognises that items in a list can be accessed individually. - Describes how loops can be used to process items stored in a list.	- Can create simple while loops and for loops. - Can use the range() function in basic programs. - Can create and use simple lists. - Can access individual list items using their index position. - May use some basic list functions when provided with examples. - Can complete structured programming activities with guidance.
Emerging	- Recognises that loops allow instructions to be repeated. - Recognises that there are different types of loops in Python. - Recognises that lists can store multiple items. - Recognises that items can be stored in a particular order within a list.	- Can write and run simple Python programs with support. - Can create simple loops by adapting provided examples. - Can identify items within a list. - Can access individual items from a list with guidance. - Can record basic notes about programming concepts.

Spring Term 1 – How Computers Work

	Knowledge	Skills
Strong	- Clearly and accurately communicates how the major hardware components of a computer work together to process and store data. - Clearly communicates the purpose of the BIOS and the stages involved in the boot process. - Clearly communicates how instructions are fetched, decoded and executed, including the roles of the Control Unit, buses, primary memory and ALU. - Clearly communicates how CPU characteristics such as clock speed, number of cores and RAM influence computer performance. - Clearly communicates why secondary storage is necessary and how it differs from volatile primary memory. - Clearly communicates how magnetic, optical and solid-state technologies store and retrieve data. - Clearly communicates the role of peripherals within a computer system and the methods used to connect them. - Clearly communicates the purpose of device drivers in enabling communication between hardware and the operating system. - Clearly communicates how an operating system manages hardware resources, memory, processes and the user interface. - Clearly communicates the relationship between operating systems and applications. - Clearly communicates the characteristics, advantages and limitations of open-source and proprietary software, using Linux as an example.	- Can analyse how hardware and software components work together to enable a computer system to function. - Can interpret and explain complex diagrams showing the fetch-decode-execute cycle and data movement within a computer. - Can compare and justify the suitability of different CPU, memory and storage technologies for specific scenarios. - Can analyse the advantages and limitations of different peripheral connection methods. - Can explain the impact of hardware specifications on system performance. - Can evaluate the suitability of open-source and proprietary software for different users and organisations. - Can use technical terminology accurately and consistently in extended written responses. - Can maintain comprehensive notes demonstrating a detailed understanding of computer systems and architecture.

Confident	- Communicates the role of the CPU, motherboard, primary memory and secondary storage within a computer system. - Communicates the purpose of BIOS and the boot process. - Communicates how instructions are processed using the fetch-decode-execute cycle. - Communicates the roles of the Control Unit and ALU within the CPU. - Communicates how CPU performance is affected by clock speed, RAM and the number of CPU cores. - Communicates why secondary storage is required and why primary memory alone is insufficient. - Communicates how magnetic, optical and solid-state storage technologies store data. - Communicates the purpose of peripherals and the differences between input and output devices. - Communicates how peripherals can connect using technologies such as USB and Bluetooth. - Communicates the role of an operating system, including providing a user interface and managing hardware resources. - Communicates the purpose of applications and how they interact with the operating system. - Communicates the differences between open-source and proprietary software.	- Can explain how hardware components interact within a computer system. - Can interpret diagrams showing the fetch-decode-execute cycle. - Can compare storage technologies using their characteristics and uses. - Can identify factors that affect CPU performance. - Can explain the role of peripherals and suitable connection methods for different devices. - Can distinguish between operating systems and applications in real-world scenarios. - Can use appropriate technical terminology in written and verbal responses. - Can maintain organised and detailed notes throughout the unit.
Developing	- Describes the purpose of the CPU, motherboard, primary memory and secondary storage. - Describes what happens when a computer starts up. - Explains that the CPU processes instructions.	- Can label the main components of a computer system. - Can identify the role of common hardware components. - Can classify devices as input, output, primary memory or secondary storage.

	• Describes the difference between primary memory and secondary storage. • Explains that storage devices can be optical, magnetic or solid state. • Describes the difference between input and output devices. • Explains that operating systems enable computers to function. • Describes the difference between applications and operating systems. • Explains the difference between proprietary and open-source software.	• Can identify examples of optical, magnetic and solid-state storage. • Can classify software as either an operating system or an application. • Can identify examples of open-source and proprietary software. • Can complete structured written tasks using appropriate terminology.
Emerging	• Recognises that computers contain different components which work together. • Recognises that the CPU is an important part of a computer. • Recognises that computers use memory and storage. • Recognises that computers use input and output devices. • Recognises that operating systems and applications are different. • Recognises that different operating systems exist.	• Can identify common computer components from images or diagrams. • Can identify examples of storage devices and peripherals. • Can recognise input and output devices. • Can use basic computing terminology with support. • Can complete structured note-taking activities.

Spring Term 2 – Artificial Intelligence

	Knowledge	Skills
Strong	- Clearly and accurately communicates the key concepts of artificial intelligence, machine learning, Large Language Models and Artificial General Intelligence. - Clearly communicates how AI systems are trained and how they differ from traditional rule-based systems. - Clearly communicates the purpose, strengths and limitations of the Turing Test as a measure of machine intelligence. - Clearly communicates the opportunities created by AI in areas such as science, accessibility, productivity and automation. - Clearly communicates the environmental, ethical and societal challenges associated with AI technologies. - Clearly communicates the risks posed by misinformation, deepfakes and AI-generated content. - Clearly communicates the role of AI agents in supporting workplace productivity and decision-making. - Clearly communicates how AI is changing the way humans create software, websites and digital products. - Clearly communicates a reasoned prediction about how AI may influence future careers, lifestyles and society.	- Can analyse real-world examples of AI and explain the technologies involved. - Can assess the benefits and drawbacks of AI applications in different contexts. - Can evaluate the reliability of AI-generated content and identify potential misinformation. - Can explain how AI agents could be applied to improve productivity in a range of professional settings. - Can investigate and demonstrate how AI-assisted development tools can support the creation of software and web applications. - Can construct balanced arguments about the societal, ethical and environmental impacts of AI. - Can use technical AI terminology accurately and consistently in extended written responses. - Can produce a detailed and well-reasoned prediction about the future impact of AI on careers, lifestyles and society. - Can maintain comprehensive notes demonstrating a detailed understanding of AI concepts and their implications.
Confident	- Communicates the definition of artificial intelligence and how it differs from traditional computer programs. - Communicates how machine learning enables AI systems to improve their performance using data. - Communicates the differences between task-specific AI, Large Language	- Can explain the differences between machine learning, task-specific AI, LLMs and AGI. - Can analyse examples of AI use and identify benefits, limitations and risks. - Can distinguish between reliable information and examples of AI-generated misinformation. - Can explain the role of AI agents in professional environments.

	Models and Artificial General Intelligence (AGI). - Communicates the purpose and significance of the Turing Test. - Communicates how AI can support scientific advancement, independence and efficiency. - Communicates the environmental, privacy and misinformation challenges associated with AI. - Communicates how AI-generated content can contribute to misinformation and "AI slop". - Communicates the role of AI agents in improving professional productivity. - Communicates how AI tools are changing the way software and websites can be developed.	- Can explore and evaluate AI-assisted approaches to software and web development. - Can use appropriate AI terminology accurately in written and verbal responses. - Can maintain clear and organised notes throughout the unit. - Can produce a well-structured written response about the future impact of AI.
Developing	- Describes AI as computer systems that can perform tasks that normally require human intelligence. - Explains that machine learning allows computers to learn patterns from data. - Recognises that some AI systems are designed for specific tasks. - Describes what a Large Language Model (LLM) is. - Recognises that AGI refers to a more general form of intelligence than current AI systems. - Describes the purpose of the Turing Test. - Explains some benefits and risks associated with AI. - Recognises that AI can contribute to misinformation.	- Can identify different types of AI from descriptions and examples. - Can distinguish between AI and non-AI technologies. - Can identify examples of machine learning and Large Language Models. - Can identify positive and negative impacts of AI from given scenarios. - Can organise research and notes about AI topics. - Can complete structured written activities using appropriate terminology.
Emerging	- Recognises that artificial intelligence (AI) is a type of computer technology. - Recognises that AI can carry out tasks that normally require human intelligence. - Recognises that AI is used in everyday life.	- Can identify examples of AI in everyday life. - Can recognise simple examples of AI-generated content. - Can participate in discussions about the impact of AI. - Can complete basic note-taking activities.

	• Recognises that AI can have both positive and negative effects. • Recognises that computers can generate text, images and other content using AI.	• Can use AI-related terminology with support.
--	---	--

Summer Term 1 – IT Skills For Life

	Knowledge	Skills
Strong	- Clearly and accurately communicates the purpose and functionality of a wide range of tools and features within Word, PowerPoint and Outlook. - Clearly communicates how office applications can be combined to create professional and effective communication materials. - Clearly communicates how formatting, layout and design influence the effectiveness of digital documents. - Clearly communicates how audience, purpose and context affect decisions made during document creation. - Clearly communicates how professional communication tools support collaboration, organisation and productivity.	- Can independently plan, create and refine a complete set of professional promotional materials using Word, PowerPoint and Outlook. - Can confidently select and apply advanced tools and features to improve the quality and effectiveness of documents. - Can make informed design decisions based on audience and purpose. - Can produce consistently well-formatted and professionally presented work. - Can use Outlook effectively to produce professional communications relating to an event. - Can integrate content across multiple applications to create a cohesive promotional campaign. - Can evaluate and improve documents to enhance clarity, presentation and impact. - Can maintain a highly organised digital workspace and file structure throughout the project.
Confident	- Communicates the purpose and functionality of a range of tools within Word, PowerPoint and Outlook. - Communicates how different office applications can be used together to create effective promotional materials. - Communicates the importance of structure, formatting and design when producing professional documents. - Communicates how audience and purpose influence design choices. - Communicates how Outlook can be used to support professional communication and organisation. - Communicates the benefits of selecting appropriate software tools for different tasks.	- Can independently create well-structured documents, presentations and emails using Word, PowerPoint and Outlook. - Can apply a range of formatting and design tools appropriately. - Can select suitable layouts, images and content for a target audience. - Can use Outlook tools effectively to support communication and organisation. - Can combine work from multiple applications to produce coherent promotional materials. - Can create professional-quality documents that promote a TGGS school event. - Can maintain clear and organised digital files throughout the project.

Developing	• Describes the purpose of Word, PowerPoint and Outlook. • Identifies common tools within each application and their basic functions. • Explains how documents, presentations and emails can be used to communicate information. • Describes the importance of organising information clearly. • Explains that documents should be appropriate for their intended audience. • Recognises that different software applications can be used together to create promotional materials.	• Can create documents, presentations and emails using the appropriate application. • Can use common formatting tools such as font styles, colours, alignment and images. • Can insert and edit content within Word and PowerPoint. • Can use basic Outlook features to compose and organise emails. • Can create simple promotional materials for a given purpose. • Can complete structured software tasks with occasional support.
Emerging	• Recognises that office software can be used to create documents and communicate information. • Recognises that Word, PowerPoint and Outlook have different purposes. • Recognises that software contains tools to help users create and edit work. • Recognises that documents can be designed for a particular audience. • Recognises that promotional materials are used to communicate information about events.	• Can open, edit and save documents using office software with support. • Can enter and format basic text. • Can use a small range of tools when guided. • Can create simple documents and presentations. • Can send or draft emails with support. • Can record notes about software features and their uses.

Summer Term 2 – Python RPG Game

	Knowledge	Skills
Strong	- Clearly and accurately communicates how decomposition allows a large program to be broken into smaller subprograms. - Clearly communicates how subprograms improve readability, organisation and maintainability. - Clearly communicates how variables, lists, selection and iteration combine to create interactive game experiences. - Clearly communicates how program structure influences user experience and gameplay. - Clearly communicates the role of testing, debugging and refinement within the software development process. - Clearly communicates how algorithms and program design can be used to create increasingly sophisticated game systems.	- Can independently design, develop and refine a complete RPG experience. - Can decompose a game into multiple well-structured subprograms. - Can create interconnected levels that respond appropriately to player choices. - Where appropriate - can effectively combine variables, lists, selection, iteration and subprograms. - Can systematically test, debug and improve gameplay. - Can write well-structured, readable and maintainable code. - Can create original content that enhances the quality and complexity of the final game.
Confident	- Communicates how subprograms help to structure larger programs. - Communicates how variables, selection, iteration and subprograms work together within a game. - Communicates how game levels can be designed as separate subprograms. - Communicates how player choices influence program flow. - Communicates the importance of testing and debugging throughout development. - Communicates how algorithms can be planned before implementation.	- Can independently create subprograms that form part of a larger RPG. - Can use variables, selection, iteration and subprograms effectively. - Can design and create original game levels. - Can link multiple subprograms together to create a playable experience. - Can test and debug programs systematically. - Can maintain clear and organised documentation for their game.
Developing	- Describes the purpose of a subprogram. - Explains how subprograms can make programs easier to organise.	- May create simple subprograms. - Can use inputs, outputs and variables within a game. - Can create simple decision-making using IF statements.

	• Explains how selection can be used to create different outcomes in a game. • Explains how loops can control repetition within a game. • Explains how variables can store information about a player or game state. • Explains the importance of planning before creating a program.	• May create basic game locations or events using subprograms. • Can test programs and fix straightforward errors. • Can create a simple level based on a provided structure.
Emerging	• Recognises that programs are designed to solve problems. • Recognises that programs are made up of different programming constructs. • Recognises that algorithms can be planned before coding. • Recognises that programs need to be tested.	• Can create and run simple Python programs with support. • May use provided subprograms within a larger program. • Can modify text and outputs within an existing RPG, but these may be inaccurate. • Can follow some of a provided design for a game level. • Can record notes about programming concepts and techniques.