Communicating safely online

Links with Digital Technologies curriculum areas

Strand: Digital Technologies Processes and Production Skills

Content description: Explore how people safely use common information systems to meet information, communication and recreation needs. (ACTDIP005)

Strand: Digital technologies knowledge and understanding

Content description: Create and organise ideas and information using information systems independently and with others, and share these with known people in safe online environments. (ACTDIP006)

One of the aims of the Digital Technologies curriculum is to apply protocols and legal practices that support safe, ethical and respectful communications and collaborations with known and unknown audiences.

The lesson Sharing online can be taught in context with the learning areas, HPE and English. Teachers may also decide to integrate the lesson with Cybersafety and Netiquette concepts.

There are a number of opportunities for students to participate in classroom discussions where they can speak, listen and exchange ideas with peers in pairs, small groups and whole-class situations. Students are also provided with opportunities to construct texts using word processing and presentation software. This will be dependent on the age and experience of students, level of support required and resources available. A strategy that is mutually beneficial is the use of older students acting as buddies to support and teach younger students.

Year level	**Digital Technologies**	HPE	**English**
Foundation	Explore how people safely use common information systems to meet information, communication and recreation needs Create and organise ideas and information using information systems	**Being healthy, safe and active** Identify people and demonstrateprotective behaviours and other actions that help keep themselves safe and healthy **Communicating and interacting for health and wellbeing** Practise personal and social skills to interact positively with others	**Interacting with others** Use interaction skills including listening while others speak, using appropriate voice levels, articulation and body language, gestures and eye contact **Creating texts** Construct texts using software including wordprocessing programs

DIGITAL TECHNOLOGIES HUB

Year 1	independently and with others, and share these with known people in safe online environments	**Being healthy, safe and active** Practise strategies they can use when they feel uncomfortable, unsafe or need help with a task, problem or situation **Communicating and interacting for health and wellbeing** Describe ways to include others to make them feel they belong Collect, check and classify data Create displays of data using lists, table and picture graphs and interpret them	**Interacting with others** Engage in conversations and discussions, using active listening behaviours, showing interest, and contributing ideas, information and questions **Creating texts** Construct texts that incorporate supporting images using software including word processing programs
Year 2			**Interacting with others** Use interaction skills including initiating topics, making positive statements and voicing disagreement in an appropriate manner, speaking clearly and varying tone, volume and pace appropriately **Creating texts** Construct texts featuring print, visual and audio elements using software, including word processing programs

Communicating safely online

Strand: Digital Technologies Processes and Production Skills

Content description: Analyse and visualise data using a range of software to create information, and use structured data to model objects or events (ACTDIP026).

Explanation: When developing a solution to any problem, be it digital or not, one of the biggest mistakes one can make is diving straight into the implementation phase without thinking about the design of the key elements of the problem and the design of the solution. To that end, it is important to model the system that you will be using – this includes both clearly defining the process that needs to be followed to develop the solution, and careful consideration of the nature of the data you're working with.

Any algorithm that is to be translated into software must be able to be represented using the structure, rules and conventions of the programming language being used. This means that it isn't enough to define the data being used in broad terms such as 'all of the votes', you also need to understand how that might be structured inside a program. For something like a collection of votes, the logical data structure that would be used in a program would be something like a list or array – something that allows you to store a collection of similar things in a way that can be accessed and manipulated for access to individual items or to the collection as a whole.

If an incorrect structure is used to model or represent data in a program, working out how to use that data becomes difficult and students risk over-complicating the problem, which can be very frustrating. Thus, being able to take a real world object (such as a ballot box) and think about it in general terms that are analogous to some form of data structure (a list, or collection of votes) is a skill that is fundamental to all software and application development.

Although not covered in this activity, databases are an example of a system that requires careful consideration of how data should be structured. A correct data structure provides the user with great flexibility and power when accessing and analysing the data, whereas a poorly constructed database would hinder their ability to draw relationships between elements of the data set and therefore make it difficult to draw conclusions about the data.

Content description: Define and decompose real-world problems taking into account functional requirements and economic, environmental, social, technical and usability constraints (ACTDIP027).

Explanation: By analysing the problem and visualising the steps in the process, students can determine what kinds of operations it will be necessary to perform on the data they are using. This helps them work out how the data in their program needs to be structured, but is integral to the selection of appropriate operations and sequencing in their algorithm design.

Flowcharts are one example of a useful tool for this process. They can be done on paper or using online tools – it is usually faster to generate them in draft form on paper before publishing the solution in an electronic format. The flowcharts can then be used as the basis for algorithm implementation, and also provide a means of identifying natural control structures needed for the algorithm, such as branching and iteration.

Since decomposing the problem into smaller problems helps identify how individual components or elements of the solution are affected, students can think about these smaller problems when determining the constraints that might be applicable to the problem they're solving. Maybe it will only deal with numbers. Perhaps it only works for valid votes because assumptions are made about the data. Can the application be used by someone who is vision-impaired? These kinds of questions often lead to more robust solutions, or give students an incentive to think about the implications of their software's use or misuse.

Content description: Implement and modify programs with user interfaces involving branching, iteration and functions in a general-purpose programming language (ACTDIP030).

Explanation: This activity focuses primarily on the processing of data rather than the design of a user interface. The solutions provided also don't make use of any user-defined functions, although there are clearly opportunities to do so when determining the lowest and highest candidate in each round of voting. This has the potential to simplify the flowcharts and can be used to introduce the concept of subroutines – whether or not to do this will be dependent on the experience of students in the group.

The use of Python as the programming language of choice is a carefully considered decision. Python is a powerful language with relatively simple grammar and syntax conventions that encourage good programming practice. It maps nicely to the flowcharts that students will design, and doesn't require any kind of wrapping functions or class declarations that beginning programmers don't understand.

The choice of problem was also deliberate – it provides strong curriculum links to Civics and Citizenship, has relevance to current events (and elections occur regularly), and the problem can be stated in increasingly complex terms. The implementation of a 'First past the post' solution requires the use of branching, iteration and input processing while keeping the problem definition relatively simple, whereas the preferential system adds additional layers of complexity for students ready for a greater challenge.

Content description: Evaluate how student solutions and existing information systems meet needs, are innovative, and take account of future risks and sustainability (ACTDIP031).

Explanation: Once a working solution is developed, students can then alter the data being provided to see the direct consequences of changes to their program. The process of testing also has the potential to reveal indirect consequences of their solution – if incorrect results are found when you change the votes being put into the system and you were to use your program to determine the result of a federal election, what does this mean? Would it be possible for someone to undermine the political process and our democratic system? How could we prevent this from happening? Those kinds of questions and discussions once the software is complete are an important part of the learning process that teachers need to facilitate in order to give students a reason to think beyond the immediate future and application of their solution.