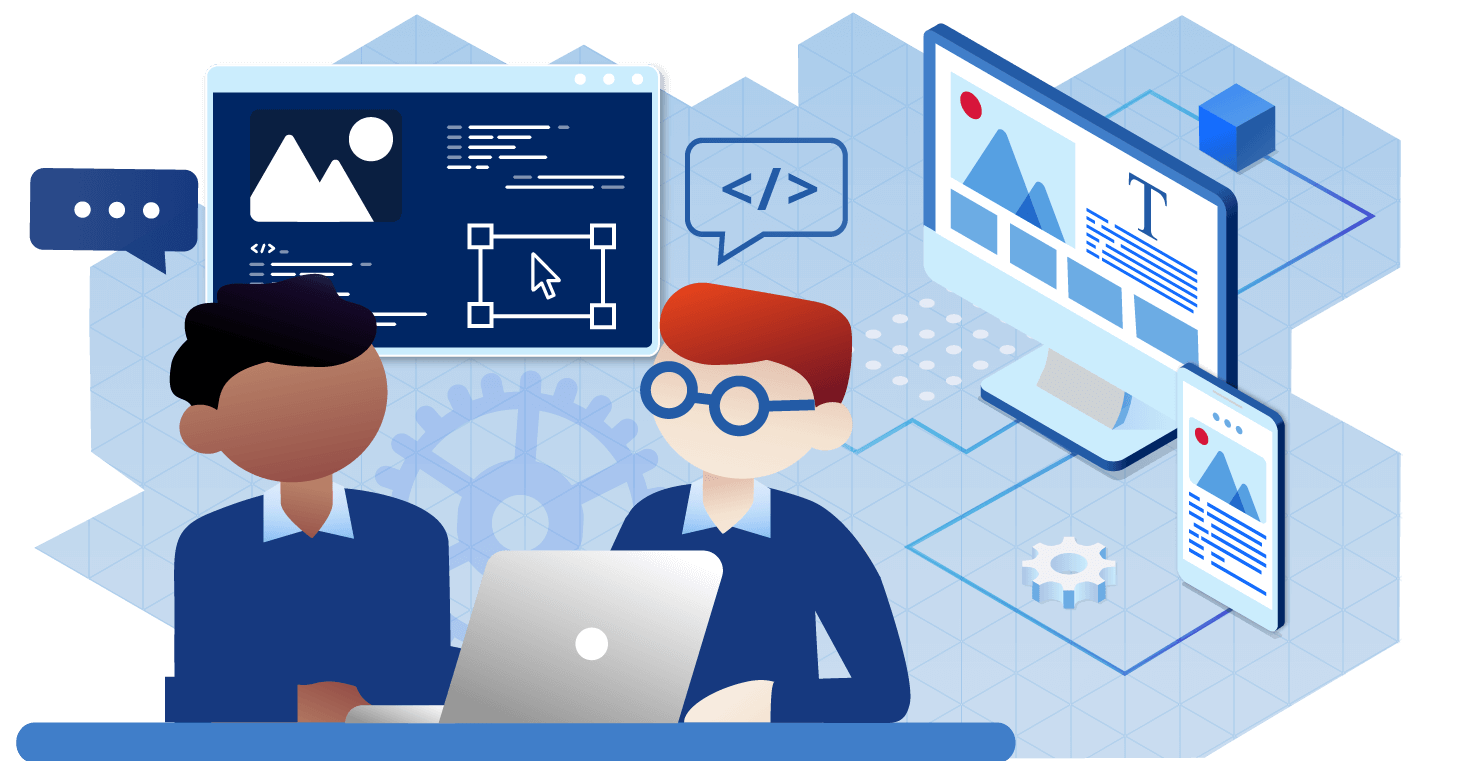
# Software Engineering Stage 6 (Year 11) – teacher support resource

**Programming fundamentals**



**Teacher note:** this resource has been designed to facilitate conversion into a student workbook by removing the answers within the response windows.

Student name:

Class:

Teacher:

Contents

[Software Engineering Stage 6 (Year 11) – teacher support resource 1](#_Toc138256371)

[Unit overview 4](#_Toc138256372)

[Assessment overview 5](#_Toc138256373)

[Steps to success 6](#_Toc138256374)

[What is the teacher looking for? 7](#_Toc138256375)

[Glossary 8](#_Toc138256376)

[NESA glossary keywords 13](#_Toc138256377)

[Strategies to teach programming 14](#_Toc138256378)

[The design and production process 15](#_Toc138256379)

[Software development 17](#_Toc138256380)

[Explore fundamental software development steps used by programmers when designing software 17](#_Toc138256381)

[Project Advice 29](#_Toc138256382)

[Research and evaluate the prevalence and use of online code collaboration tools 35](#_Toc138256383)

[Designing algorithms 37](#_Toc138256384)

[Apply computational thinking and algorithmic design by defining the key features of standard algorithms, including sequence, selection, iteration and identifying data that should be stored 37](#_Toc138256385)

[Apply ‘divide and conquer’ and ‘backtracking’ as algorithmic design strategies 40](#_Toc138256386)

[Assessment task guide part 2 42](#_Toc138256387)

[Develop structured algorithms using pseudocode and flowcharts, including the use of subprograms 59](#_Toc138256388)

[Use modelling tools including structure charts, abstraction, and refinement diagrams to support top-down and bottom-up design 64](#_Toc138256389)

[Analyse the logic and structure of written algorithms 65](#_Toc138256390)

[Identify procedures and functions in an algorithm 69](#_Toc138256391)

[Experiment with object-oriented programming, imperative, logic and functional programming paradigms 74](#_Toc138256392)

[Data for software engineering 85](#_Toc138256393)

[Investigate the use of number systems for computing purposes, including binary, decimal and hexadecimal 85](#_Toc138256394)

[Represent integers using two’s complement 99](#_Toc138256395)

[Investigate standard data types 102](#_Toc138256396)

[Create data dictionaries as a tool to describe data and data types, structure data and record relationships 112](#_Toc138256397)

[Use data structures of arrays, records, trees and sequential files 114](#_Toc138256398)

[Developing solutions with code 116](#_Toc138256399)

[Apply skills in computational thinking and programming to develop a software solution. 116](#_Toc138256400)

[Implement data structures that support data storage 117](#_Toc138256401)

[Compare the execution of the Waterfall and Agile project management models as applied to software development 125](#_Toc138256402)

[Test and evaluate solutions, considering key aspects including functionality, performance, readability of code and quality of documentation 128](#_Toc138256403)

[Use debugging tools 129](#_Toc138256404)

[Determine sets of suitable test data 131](#_Toc138256405)

[Determine typical errors experienced when developing code, including syntax, logic, and runtime, and explain their likely causes 137](#_Toc138256406)

[Appendix 1 – basic programming concepts pre-test 140](#_Toc138256407)

[References 155](#_Toc138256408)

## Unit overview

In this unit, students will develop foundational knowledge and understanding of software programming. The lessons and sequences in this teacher resource are a guide for students to develop algorithms, learn about standard control structures and how to apply and evaluate them in the design of software.

During Weeks 1–3 students explore fundamental software development steps used by programmers when designing software including requirements definitions and determining specifications.

They participate in a guided scenario that requires them to use Python to program an educational software product for a Primary teacher and their class. This software enables input of 3 lengths (the sides of a triangle: a, b and c) and will output the type of triangle: equilateral, scalene, isosceles or right angle.

This guided project allows exploration into the fundamentals of programming. Students refer to the key concepts introduced in this experience of software development to revisit foundational knowledge, as well as broadening and deepening their understanding.

In Weeks 4–6 students refer to their educational software to investigate algorithm design and data for software engineering.

They experiment with other paradigms in solving the same triangle recognition problem.

During Weeks 7–10 students apply computational thinking and the development approach used in their triangle recognition program to develop their own project that is the assessment task for this unit.

Students who have not completed the Computing Technology Stage 5 course and/or are inexperienced in programming may be encouraged to add functionality to their existing triangle recognition educational software as a working submission for their assessment task.

Other projects could include the conversion of a standard board game into a computer game, designing a memory game, maze, or arcade-style video game.

## Assessment overview

**Type of task:** create and document educational software created in a high-level general-purpose programming language.

**Outcomes:**

A student:

* describes methods used to plan, develop and engineer software solutions **SE-11-01**
* explains how structural elements are used to develop programming code **SE-11-02**
* applies tools and resources to design, develop, manage and evaluate software **SE‑11-06**
* implements safe and secure programming solutions **SE-11-07**

[Software Engineering 11–12 Syllabus](https://curriculum.nsw.edu.au/syllabuses/software-engineering-11-12-2022?tab=course-overview) © NSW Education Standards Authority (NESA) for and on behalf of the Crown in the State of New South Wales, 2022.

**Suggested weighting: 25%**

You are to design, code and document a software solution for an educational product. The software is to be developed in a high-level general-purpose programming language.

The educational piece of software is to be an interactive solution. The software could be designed for either primary or secondary students. It will include an intuitive command line interface that displays directions and responses to the user’s requests.

### Steps to success

Table 1 – assessment preparation schedule

|  |  |
| --- | --- |
| Steps | What I need to do |
| Software Development  Explore the fundamental development steps in relation to their project. | Write a requirements definition for the given real-world problem.  Identify the user specifications for the chosen software solution. |
| Designing Algorithms | Develop a structured algorithm using pseudocode or flowcharts including the use of subprograms and passing parameters.  Algorithms should include sequence, selection, iteration and subprograms and be described using a structure chart. |
| Data for Software Engineering | Define and discuss the use of following data types. Select a minimum of 3 to use in your project:   * char and string * Boolean * real * single precision floating point * integer * date and time.   Create a Data Dictionary for use with your project.  Define and discuss the use of the following data structures. Select a minimum of 2 to use in your project:   * arrays * records * trees * sequential files. |
| Developing Solutions with Code | Convert your algorithm into code using:   * control structures * data structures * standard modules * subprograms (including parameter passing).   Define and discuss the following debugging tools used in your project:   * breakpoints * single line stepping * watches * interfaces between functions * debugging output statements.   Document and implement at least one appropriate data structure that supports data storage.  Describe the errors you experienced in the coding of your solution including:   * syntax * logic * runtime. |

### What is the teacher looking for?

This task will require students to choose a section of the curriculum associated with either primary or secondary students. For example, the software could help students learn their times table, recognise types of triangles or learn a new language. Students design, develop and document a software solution coded in a high-level general-purpose programming language that assists students in learning their chosen content area.

## Glossary

Many of the following words will gather more meaning to you as you work through this booklet.

Each time you see an unfamiliar word in bold throughout this workbook you can add its definition in the table below in case you need to refer to it later.

**Teacher note:** these definitions should be removed prior to supplying this table to students

Table 2 – glossary

|  |  |
| --- | --- |
| Word | Definition |
| algorithm | A step-by-step procedure required to solve a problem. Algorithms may be presented in many ways, for example written instructions, flow charts or using a computer programming language. |
| app | A software program designed for a specific purpose to run on mobile devices or on a personal computer. An abbreviation of the word ‘application’. |
| application programming interface (API) | An interface that allows an application or website to plug into another program or website. |
| artificial intelligence (AI) | Intelligence demonstrated by machines. Sometimes called machine intelligence. |
| assets | Assets are the digital elements that can be used or incorporated into a multimedia presentation or production. They include graphics, photographs, videos, audio, animations or other artistic data. |
| big data | Extremely large datasets that may be analysed computationally to reveal patterns, trends and associations. |
| black box testing | A testing technique where there is no knowledge of the internal workings of an application. A user will provide inputs and observe the outputs in order to determine functionality of the program. |
| collaborate | Working with others towards a shared goal, through a variety of modes of communication. This may be achieved using a range of technologies, tools and processes. |
| computational thinking | A process in which a problem is analysed and solved so that a human, machine or computer can effectively implement the solution. It involves using strategies to organise data logically, break down problems into parts, interpret patterns and design, and implement algorithms to solve problems. |
| data | A discrete representation of information using number codes. Data may include characters (for example alphabetic letters, numbers and symbols), images, sounds and/or instructions that, when represented by number codes, can be manipulated, stored and communicated by digital systems. For example, characters may be represented using ASCII code or images may be represented by a bitmap of numbers representing each ‘dot’ or pixel. |
| data sovereignty | This is a global movement that refers to the right of Indigenous peoples to determine the creation, collection, ownership and application of data that is for and about Indigenous peoples. The term also refers to transnational data flow and a range of issues arising from the flow of data between nations. |
| data types | Data types used in computing are expressed as either:   * string/text * character * integer * floating point or real * date and time * Boolean. |
| design thinking | A process where a need or opportunity is identified, and a design solution is developed. The consideration of economic, environmental and social impacts that result from designed solutions are core to design thinking. Design thinking methods can be used when trying to understand a problem, generate ideas and refine a design, based on evaluation and testing. |
| digital technologies | Refers to electronic tools, systems, devices and resources that generate, process or store data, and may include applications, games, microcontrollers, mobile devices, multimedia, networks, robotics. |
| evaluate | Make a judgement based on criteria; determine the value of. |
| grey box testing | A testing technique where there is a partial knowledge of the internal workings of an application. Its purpose is to search for defects related to improper code structure or the improper use of functions. This technique gives the ability to test both the code and the presentation layer of an application.  **Note:** grey box testing and gray box testing mean the same. |
| information | The presentation of data in a manner that is readily understood. |
| information system | The combination of digital systems, people and processes that collect, manage, and analyse data. |
| model | A mathematical, conceptual or physical representation that describes, simplifies, clarifies or provides an explanation of the structure, workings or relationships within an object, system or idea. Models can provide a means of testing and predicting behaviour within limited conditions. Models may be physical or exist in digital form. |
| object-oriented programming (OOP) | A paradigm based on the concept of ‘objects’ that can contain data and code in the form of procedures. OOP language is a language based on the principles of ‘C’. |
| Object-Relational Mapping (ORM) | A technique used in object-oriented programming to query and manipulate data from a database. |
| procedural programming | A method of programming where the program is divided into functions. A program consists of data and procedures (modules) that operate on the data. Data and procedures are treated as separate entities. |
| prototype | A trial product or model built to test an idea or process to inform further design development. Its purpose is to see if and how well the design works and is tested by users and systems analysts. A prototype can be both a physical object and exist in digital form.  *See the definition of* model above. |
| pseudocode | A form of algorithm description that uses English-like statements with defined rules of structure and keywords. |
| refinement charts | A diagrammatic model that breaks down the steps of an outline solution into smaller and smaller steps. |
| scripting language | A programming language designed for integrating and communicating with other programming languages. |
| structured data | Data with a high degree of organisation, such that it can easily be analysed and processed using algorithms. The organisation of the data takes on a form that captures the properties and behaviours of the source the data represents. |
| structured query language (SQL) | Specialist programming language used to manage data and access data in relational database management systems. |
| systems thinking | An understanding of how related objects or components interact to influence how systems function. Students are provided with opportunities to recognise the connectedness of, and interactions between phenomena, people, places and events in local and wider contexts and consider the impact of their decisions. Understanding the complexity of systems and the interdependence of components is important for scientific research and for the creation of solutions to technical, economic and social issues. |
| user experience (UX) | The usability, ease of use and enjoyment provided in the interaction between the customer and the product. |
| user interface (UI) | The means by which users interact with computer hardware or software. In software, this usually comprises fields for text and number entry, mouse pointers, buttons and other graphical elements. In hardware, switches, dials and light-emitting diodes (LEDs) provide information about the interactions between a user and a machine. |
| W3C | The World Wide Web Consortium is the international standards organisation for the world wide web. All the standards used for web development are defined by this organisation. |
| white box testing | A testing technique used to examine the internal workings of an application including the design, code and inner workings of software. This technique gives the ability to verify the flow of inputs and outputs through the application. |

**Teacher note:** for students for whom English is an additional language or dialect (EAL/D), the glossary can be provided complete so that they have additional time to understand the key terms with bilingual dictionaries.

## NESA glossary keywords

[NESA keywords](https://educationstandards.nsw.edu.au/wps/portal/nesa/11-12/hsc/hsc-student-guide/glossary-keywords) can be used in the syllabus and in the HSC (Higher School Certificate) examination. Familiarisation with these keywords can assist in understanding how to write and respond to questions.

Table 3 – NESA glossary keywords

|  |  |
| --- | --- |
| Key term | Definition |
| analyse | identify components and the relationship between them; draw out and relate implications |
| apply | use, utilise, employ in a particular situation |
| compare | show how things are similar or different |
| identify | recognise and name |
| investigate | plan, inquire into and draw conclusions about |

**Teacher note:** ‘explore’, ‘develop’, ‘research’, ‘use’, ‘represent’, ‘create’, ‘implement’, ‘determine’, ‘test’ and ‘experiment’ are used in this topic and are not listed.

## Strategies to teach programming

**Teacher note:** teachers are advised to start activities with pseudocode and/or flowcharts to discuss concepts before converting into code to deal with: syntax errors, to use test data to debug the algorithm, and to check the algorithm’s structure.

It should be easier for students to write code in Python after they have worked on a pseudocode version, rather than starting with a programming language.

**Other strategies**:

* Ask students to write a program using pseudocode and then swap with a partner for them to write it in a programming language.
* Provide snippets of pseudocode to test basic misconceptions, for example in the use of recursion.
* Provide small programs that students can hand trace, writing out the values of variables as the program progresses, to check they understand the code.
* Provide snippets of pseudocode that includes sequences, selection and iteration although it is incorrectly ordered. Have students arrange these into the correct order. This strategy uses [Parson’s puzzles](https://computerscienced.co.uk/site/parsons-puzzles/) and uses [Parson’s problem solving.](https://parsons.problemsolving.io/)
* Students could be introduced to and use the [PRIMM](https://primmportal.com/) strategy where they can Predict, Run, Investigate, Modify and Make. Examples of simple programs delivered via worksheets can be viewed on the [PRIMM Portal](https://primmportal.com/2018/08/23/primm-materials-2018/).

## The design and production process

Throughout your study of technology subjects, you will have become familiar with design processes and how to apply them. In this course you will explore different types of design processes specific to Software Engineering and learn how to apply them in your project. These include the Waterfall and Agile approaches to Software development.

The generic design and production process is a sequence of organised steps which provide a solution to design needs and opportunities. It:

* may take a few seconds or minutes, such as when you select what clothes to wear, or may take years as is the case with the design of a motor vehicle
* may involve one person or many people
* may be simple or complex, depending on the task
* involves questioning (or evaluating) throughout the iterative process.

**Activity 1:** how have you applied this to a product you have made?

|  |
| --- |
| Student responses will vary.  They could be prompted to discuss a previous project from a TAS workshop. |

Figure 1 – flowchart of design and production process

The design and production process flowchart.
1. Identifying and defining
- identify and define the needs, opportunities and wants of a computing challenge.
- practise the technical skills.
- develop evaluation criteria.
2. Researching and planning
-research, generate and practise ideas.
- be creative and propose new approaches to problems.
- explore new design opportunities.
3. Producing and implementing
- build and implement ideas.
- apply a variety of skills and techniques to create products that meet set criteria.
- modify and iterate solutions.
4. Testing and evaluating
- test and evaluate solutions/products.
- evaluate quality and effectiveness  against the criteria
- make judgements throughout the solution and use these to refine the product.
The process will repeat itself if required to improve, and ongoing evaluation will take place throughout all the stages.


## Software development

### Explore fundamental software development steps used by programmers when designing software

**Activity 2:** [Think-Pair-Share](https://app.education.nsw.gov.au/digital-learning-selector/LearningActivity/Card/645#.Y_Kr3SagGrY.link) on the stimulus question below.

How is designing software different to designing a pencil box, cake, clothes, car, or any other product?

| Sample answer:  All these listed examples are physical objects made with ‘visible’ materials. The selection of these materials during the design process is a critical factor in determining the product.  Software is ‘made from’ writing digital instructions (coding) in a programming language.  At the fundamental level these instructions are binary-zeros and ones (electrical high and low voltages). This software, once designed and developed, needs to be installed inside computers and integrated into existing digital systems. It needs to be tested and debugged thoroughly and maintained in case something changes within the system. |
| --- |

**Activity 3:** what other fundamental steps would be required because of these differences?

|  |
| --- |
| Sample answer:   * Integration * Testing and debugging * Installation * Maintenance |

**Activity 4:** design and production flowchart

Modify and redraw the design and production process flowchart to indicate the fundamental steps used by programmers when designing software.

Label each process step:

* Requirements definition
* Determining specifications
* Design
* Development
* Integration
* Testing and debugging
* Installation
* Maintenance

|  |
| --- |
| Sample answer:  Student flowcharts may indicate maintenance as an iterative process (loop).  Students may also indicate testing and debugging as iterative subprocesses. |

**Teacher note:** the following activities are designed to provide students with both an introduction to the steps used by software developers and an opportunity to practise simple programming fundamentals. They can be delivered as formative assessment tasks which will inform and culminate the final project and summative assessment task for this focus area.

#### Pre-test

**Teacher note:** Computing Technology 7–10 is not a pre-requisite to study Software Engineering in Stage 6.

Students in the Software Engineering class may have a diverse range of programming experience, knowledge, and skills.

To determine where class time is best spent and encourage students to identify gaps in their own understanding, they should complete a pre-test of key programming concepts (Appendix 1).

#### Requirements definition

Consider the following scenario:

Your old primary school teacher has heard that you are studying Software Engineering and has approached you with a request for help.

‘My Year 4 maths class are learning about different [types of triangles](https://www.bbc.co.uk/bitesize/topics/zvmxsbk/articles/zggsfrd) and I would like some educational software that would help them identify each type.’

#### Determining specifications

In discussion with this ‘client’ you have familiarised yourself with the rules for identifying triangles:

* All triangles have 3 sides.
* Equilateral: equilateral triangles have 3 equal sides and 3 equal angles of 60°.
* Isosceles: isosceles triangles have 2 equal sides and 2 equal angles.
* Right-angled: one of the angles is a right angle (90°) in right-angled triangles.
* Scalene: scalene triangles have no equal sides and no equal angles.

You have also agreed that to keep your software simple and age appropriate you will keep the entry of data into the software as the length of each side of the triangle:

* Side a=?
* Side b=?
* Side c=?

#### Design

##### IPO Chart

A useful tool to better understand any problem is to use an IPO (Input, Process, Output) chart.

**Step 1**: Start with the output (for example, what it is you want the program to do)

Table 4 – IPO (Input, Process, Output) step 1

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
|  |  | Identify triangle type |

**Step 2**: What are the inputs you will need to get to the output?

Table 5 – IPO (Input, Process, Output) step 2

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| Length of Side a  Length of Side b  Length of Side c |  | Identify triangle type |

**Step 3**: What processes will change the inputs you will need to get to the output?

Table 6 – IPO (Input, Process, Output) step 3

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| Length of Side a  Length of Side b  Length of Side c | Compare the lengths of the 3 sides to each other | Identify triangle type |

The Process column of the IPO chart describes **how** the input will be processed (changed, transformed, or given meaning) into the output.

**Teacher note:** students may benefit from a revision of IPO charts in a non-computing scenario and complete an IPO for a product of their choice.

**Steps to complete an IPO chart 1 of 3**

Table 7 – steps to complete an IPO chart – step 1

|  |  |  |
| --- | --- | --- |
| INPUT | PROCESS | OUTPUT |
|  |  | Sponge Cake |

**Steps to complete an IPO chart 2 of 3**

Table 8 – steps to complete an IPO chart – step 2

|  |  |  |
| --- | --- | --- |
| INPUT | PROCESS | OUTPUT |
| Flour  Icing  Egg |  | Sponge Cake |

**Steps to complete an IPO chart 3 of 3**

Table 9 – steps to complete an IPO chart – step 3

|  |  |  |
| --- | --- | --- |
| INPUT | PROCESS | OUTPUT |
| Flour  Icing  Egg | Step 1: separate the eggs  Step 2: prepare the cake pan  Step 3: aerate the flour  Step 4: beat the egg yolks  Step 5: beat the egg whites  Step 6: fold the ingredients together  Step 7: pour batter into the pan and bake | Sponge Cake |

Notice in this sponge cake scenario each step in the process is a sub-process that requires further description. Providing more detail and description is called refinement.

**Activity 5:** create an IPO chart

The Process: ‘Compare the lengths of each of the 3 sides’ from the triangle identification IPO (Input, Process, Output) chart is broad and needs more detailed explanation.

**Teacher note:** results from the pre-test (Appendix 1) should indicate whether students need better understanding of the comparison process.

To understand how the computer compares, students may need to complete unplugged activities on [logical (AND, OR, NOT) and relational operators (<, >, = , !, =).](https://www.cs.odu.edu/~zeil/cs250PreTest/latest/Public/operators/index.html)

These could include comparing playing cards or comparing students’ heights.

**Activity 6:** rules

In the space below list the rules of triangle categories and write detailed descriptions on the sub process steps involved in ‘Compare the lengths of each of the 3 sides’

|  |
| --- |
| Sample answer:  IF SideA = SideB =SideC? it is an equilateral triangle.  IF SideA = SideB OR SideC? (or other combination of 2 sides equal?) it is an isosceles triangle.  IF SideA2 = SideB2 + SideC2? (or other combination of Pythagoras rule) it is a right-angle triangle  IF SideA != SideB != SideC? it is a scalene triangle. |

Teacher-led discussion revisits the processes involved in turning inputs into outputs during non-computing production. For example: making a chair (output) from timber (input) requires measuring, cutting, chiselling, planning, sanding, and finishing processes.

The processes used to turn inputs into outputs in software development are at the fundamental level only 3 main categories.

1. Sequence
2. Selection
3. Iteration

These processes can be combined in many ways to solve most solvable problems.

They are called ‘control structures’ and are written as algorithms.

* Sequence is step by step.
* Selection, also called branching, can be binary IF statements or multiway.
* Iteration is looping (counted, pre- or post-test loops).

**Activity 7:** in the space below, identify the processes in comparing triangle side length.

|  |
| --- |
| **Sample answer:**  The comparisons use an IF (binary selection) with conditional and logical operators to decide upon the type of triangle. |

Identify how these processes could be described and expressed as algorithms. **Hint**: whenever a comparison is made an IF statement or decision based on a criteria is not far away.

**Teacher note:** students that have experience in coding will be tempted to complete the program prior to writing the complete pseudocode (algorithm).

Novices may require assistance to combine the control structures into a solution and should be stepped through the process from simple to complex.

**Activity 8:** pseudocode

Write pseudocode that compares the lengths of 3 sides of a triangle to determine if it is equilateral.

|  |
| --- |
| **Sample answer:**  BEGIN Equilateral  Get SideA  Get SideB  Get SideC  IF SideA = SideB = SideC THEN  Display “Equilateral triangle”  ENDIF  END Equilateral |

A broad and conceptually useful equation to introduce students to computer programming and how it works is:

**programming = do + store**

**‘Do’** refers to the control structures (sequence, selection, and iteration) that are combined into algorithms that are used to manipulate or process stored data.

‘**Store’** refers to the data structures or ways to store the data (data types, variables, arrays, records) on which the algorithms work (do).

The above example uses sides A, side B and side C which would be variables, the contents of which would change depending on the length of side entered.

While determining the control structures to use and the algorithm required the programmer also considers the data types used for these variables.

**Teacher note:** the ‘programming = do + store’ equation is a useful descriptor of the imperative programming paradigm and can be used to discuss extension work on computing history including [Von Neumann architecture](https://www.geeksforgeeks.org/computer-organization-von-neumann-architecture/).

Students should be advised that other paradigms to be studied in this course include the Object-Oriented Programming languages which combine DO and STORE into an OBJECT which has Methods and Attributes.

**Activity 9:** control structures

Complete the following questions.

What control structures can be seen in the algorithm?

|  |
| --- |
| **Sample answers (in bold):**  BEGIN Equilateral  Get SideA **SEQUENCE** (STEP by STEP)  Get SideB  Get SideC  IF SideA = SideB = SideC THEN (SELECTION or BRANCHING)  Display “equilateral triangle”  ENDIF  END Equilateral |

What control structure can be seen if we wish to run this code more than once with different inputs?

|  |
| --- |
| **Sample answer:**  A loop or **ITERATION** would be added. |

**Activity 10:** which data type is most appropriate for this scenario?

What data type will be used for variables side a, side b and side c? Integers or floats?\*

|  |
| --- |
| **Sample answer:**  Given the scenario requires input data from primary school students the most appropriate data type would be integers to enable entry of whole numbers.  Students may consider other scenarios where floats could be used. |

Data types are a key feature of this focus area and will be revisited in detail and depth.

Meanwhile, as a simple demonstration of why programmers need to consider which data type to select, consider making a program for the teacher to calculate the average height of students in her class.

Even if the height of each student is rounded to an integer the division by the number of students may produce an average height that is a decimal point number (floating point\*).

**Teacher note:** students will consider data types in detail later in this unit.

**Activity 11:** program

****Write this pseudocode into a programming language (Python), run the code and debug any syntax or runtime errors. Keep track of all errors in a ‘Bug Book’ (see Activity 120).

**Activity 12:** [Jigsaw](https://app.education.nsw.gov.au/digital-learning-selector/LearningActivity/Card/546#.ZGG8C-CIDu8.link)

Students write the pseudocode for each of the other types of triangles.

This is a continuation of Activities 7 and 8.

Teachers reinforce to students the use of logical and conditional operators and where these can be used in the pseudocode to determine the differences in the lengths of each side.

|  |
| --- |
| **Sample answer:**  BEGIN triangle  Get SideA  Get SideB  Get SideC  IF SideA = SideB OR SideB = SideC OR SideA = SideC THEN  Display “Isosceles triangle”  ENDIF  IF SideA <> SideB <> SideC THEN  Display “Scalene triangle”  ENDIF  IF SideC2 = SideA2 + SideB2 THEN  Display “Right angle triangle”  ENDIF  END triangle  \* Students should be made aware that in some languages not equal is typed as: != or !==  \* Students should be made aware that one is assuming C is the largest length given (for right-angle triangle)  \* The output should show that a right-angle triangle is also either an isosceles or scalene. |

**Teacher note:** if wanting to teach functions and parameter, passing/returning could be an idea to get the students to write a solution for each type of triangle in its own program or mainline and then discuss how these could be merged into one using functions (link to each solution made).

**Activity 13:** pseudocode

****Teacher leads the discussion on where best to arrange and nest these comparisons within the mainline to identify the correct type of triangle based on the lengths of the 3 sides. The flowchart and pseudocode for nested IF statements within the [Software Engineering Course Specifications](https://curriculum.nsw.edu.au/syllabuses/software-engineering-11-12-2022?tab=teaching-and-learning) document should be referred to in this activity.

Students add complexity in their pseudocode by including nested IF statements that compare for each outcome: Isosceles, Scalene, Right angle.

**Teacher note:** consider having teams of students work on the pseudocode for each type of triangle and then discuss how best to include this into the main algorithm. This could include modelling the program as a structure chart that includes a log in module.

#### Development

**Activity 14:** developing code

****Students code their solution using Python beginning with their Equilateral Triangle identifier.

Teacher leads discussion on type conversion in Python.

Students refer to their pseudocode to add complexity with nested IF statements inside a loop to gradually include identification of Isosceles, Scalene, and Right angle triangles.

Students test the results by running the code for each.

Students may benefit from completing [introductory tutorials to Python](https://pythontutorials.eu/).

**Teacher note:** teachers and observant students will identify the need to validate the input of side lengths to ensure they are integers that are greater than zero and not negative. This can be done after the students first solution to discuss logic errors.

##### Logic Errors

Test the students’ software solutions by entering 3 zeros.

**SideA = 0**

**SideB = 0**

**SideC = 0**

Students that have not considered this logical error may have programs that displays: ‘Equilateral’.

Triangles cannot be formed with sides of zero length or less and students will need to include code that validates that all sides are greater than zero (> 0).

In mathematics, the triangle inequality states that for any triangle to be valid, the sum of the lengths of any 2 sides must be greater than or equal to the length of the remaining side. Code should be written here to [validate the input data.](https://www.codesansar.com/python-programming-examples/check-validity-triangle-given-sides.htm)

Consider ways of reducing the length of their code, optimising it, and seeking efficiencies by using multiway selection (case statements), functions and an introductory discussion to OOP implementation using classes.

### Project Advice

This mini project can be extended to cater for more experienced programmers by including:

* a log in with password to use the program
* the use of functions and parameter passing
* a graphic representation of the triangle
* a list or array to store the lengths and angles
* an option to solve the identification using angles
* the first module of a more extensive educational suite
* a web-based application
* an OOP solution.

**Teacher note:** inexperienced programmers that struggle to complete their own Week 10 project could be required to extend the functionality of their triangle recognition program using one or more of the above.

#### Integration

Integration is the act of bringing together smaller components or information stored in different subsystems into a single functioning unit. In this scenario if the Triangle Identification software were used as a part of a larger educational product it could be said to have been integrated. Students may consider integrating their triangle program into a larger suite of educational software.

#### Testing and debugging

As well as testing for logical errors like entering zero values or values like: SideA = 15000, SideB = 1 and SideC = 2 which cannot make a triangle (see below) students should use Print statements and Python debugging tools like Breakpoints to test and debug their code.

In mathematics, the triangle inequality states that for any triangle to be valid, the sum of the lengths of any 2 sides must be greater than or equal to the length of the remaining side.

**Activity 15:** validating input data

****Add code to [validate the input data.](https://www.codesansar.com/python-programming-examples/check-validity-triangle-given-sides.htm)

**Activity 16:** reading and writing files

****Practice reading and writing files to and from Python.

Create [a text file and read this into Python](https://www.w3schools.com/python/python_file_open.asp).

**Activity 17:** .csv file

Create a .csv file with 3 column headings: SideA, SideB and SideC.

****[Populate the rows](https://docs.python.org/3/library/csv.html) with different side lengths and read from the .csv file into your triangle recognition program to test combinations and if their program works. Sample code can be seen below.

|  |
| --- |
| import csv, sys  def typeTriangle(sides):    sides.sort()  print(sides)  a = int(sides[0])  b = int(sides[1])  c = int(sides[2])  if (a + b >= c) and (b + c >= a) and (c + a >=b):    if a == b == c:  return("Equilateral triangle")  elif a != b != c:  if c\*\*2 == a\*\*2 + b\*\*2:  return("Right angle triangle")  else:  return("Scalene triangle")  elif a == b or b == c or a == c:  return("Isosceles triangle")  else:  return("Not a valid triangle")  else:  return("Not a valid triangle")  triSides = []  filename = 'triangle\_lengths.csv'  with open(filename, newline='') as f:  reader = csv.reader(f)  try:  for row in reader:  #print(row)  triSides.append(row)  except csv.Error as e:  sys.exit('file {}, line {}: {}'.format(filename, reader.line\_num, e))  #test triangle - assuming dictionary/2D array knowledge  #nested for loops  triSides.pop(0)  for i in range(len(triSides)):  print(typeTriangle(triSides[i])) |

**Teacher note:** both conditions could be met (right angle and scalene or isosceles).

More capable students will determine how to structure their code to display this possible result.

|  |
| --- |
| **Sample .csv file contents for Activity 17:**  Sample .csv file contents for Activity 17. |

**Teacher note:** without the line of code: sides.sort() values for a right triangle where A is the largest could be added to test validation. Note that in rows 3, 13 and 19 the C value is not the highest.

#### Installation

**Activity 18:** software installation

A Python file is merely a program or script written with Python source code and saved with the .py file extension.

Read about how to [install software](https://www.codecademy.com/article/basics-of-software-installation).

****Consider how you would install software:

* onto your byod
* onto the classroom computers.

**Teacher note:** restrictions on department devices may prohibit this from being achieved during class time. Teachers could discuss and model this process diagrammatically.

**Activity 19:** compiling and interpreting

**Homework**: watch the [Difference between Interpreter and Compiler (3:42).](https://www.youtube.com/watch?v=e4ax90XmUBc)

Class asynchronous discussion [(from the Digital Selector)](https://app.education.nsw.gov.au/digital-learning-selector/) on the differences between compiling and interpreting.

**Teacher note:** while there is no direct reference to compiling and interpreting within the syllabus these are important concepts that assist students to differentiate between Python and other languages.

**Activity 20:** Python libraries

****Explore the use of [Python libraries](https://www.datacamp.com/tutorial/two-simple-methods-to-convert-a-python-file-to-an-exe-file) to make their .py file an executable file.

#### Maintenance

**Activity 21:** maintenance

What is the maintenance phase in software engineering?

|  |
| --- |
| **Sample answer:**  The maintenance phase in the software development process is where the software is monitored to ensure it continues to function as it was designed to, and repairs or upgrades are performed as needed. After the software is released into production, updates or upgrades will need to be made. |

**Activity 22:** Brainstorm

[Brainstorm](https://app.education.nsw.gov.au/digital-learning-selector/LearningActivity/Card/542#.ZC4skSv0RLg.link) the types of upgrades or modifications that may be requested by their primary teacher client in this scenario.

|  |
| --- |
| **Sample answer:**  One example may be that the teacher has requested the lengths of the sides be input as decimal point numbers rather than integers and you as the programmer will change the data type from integer to float (real). |

### Research and evaluate the prevalence and use of online code collaboration tools

The mini project above could be delivered as a group task that requires collaboration.

**Activity 23:** collaboration

How do collaborative coding tools facilitate teamwork?

|  |
| --- |
| **Sample answer:**  Developers coding together in real-time from different locations is known as real-time collaborative coding.  This has given rise to the development of collaborative coding tools that allow developers and coders to collaborate and work in a distributed way while maintaining flexibility.  Some projects require a pair or team of developers/coders on projects that need interactive ideas and contributions in reviewing and writing code that is error free. |

**Activity 24:** collaboration tools

In the space below, list the online code collaboration tools you have researched.

|  |
| --- |
|  |

**Activity 25:** use a collaborative tool

Experiment with the free trial of [replit.com](https://replit.com/) to collaborate.

## Designing algorithms

### Apply computational thinking and algorithmic design by defining the key features of standard algorithms, including sequence, selection, iteration and identifying data that should be stored

Teacher provides a link to the [Software Engineering Course Specifications](https://curriculum.nsw.edu.au/syllabuses/software-engineering-11-12-2022?tab=teaching-and-learning) pages 14–21.

‘It is expected that students are able to develop and interpret algorithms represented as pseudocode and flowcharts.’

**Activity 26:** flowcharts and pseudocode

Compare and contrast flowcharts and pseudocode.

|  |
| --- |
| **Sample answer:**  **Compare**   * Both are ways of expressing an algorithm (the step-by-step procedure required to solve a problem).   **Contrast**   * Flowcharts are a graphic depiction of an algorithm. * Pseudocode is a plain text-based depiction of an algorithm. |

**Activity 27:** complete these pre-tests to gauge your computational thinking ability.

****The [Bebras Mini Challenge: Coding and Decoding p](https://www.csiro.au/en/education/programs/digital-careers/bebras/bebras-resources/bebras-mini-challenges)rovides an online challenge that explores computational thinking skills through a range of problems.

**Teacher note:** for students identified in the previous activity as requiring further support provide the following classroom activities:

* [Algorithms](https://www.csunplugged.org/en/at-a-distance/algorithms/)
* [Bebras unplugged](https://www.csiro.au/en/education/programs/digital-careers/program-resources/bebras-unplugged)
* [Overview of Code Studio Courses](https://code.org/educate/curriculum/courses)

**Activity 28:** complete the cloze passage using the following words:

instructions, output, sequence, instruction, options, digital devices, repeat, input, store

Computational algorithms (the kind that can run on **\_\_\_\_\_\_\_\_\_\_\_\_**) have relatively few ingredients because digital devices only have a few types of **\_\_\_\_\_\_\_\_\_\_\_\_** that they can follow; the main things they can do are receive **\_\_\_\_\_\_\_\_\_\_\_\_**, provide **\_\_\_\_\_\_\_\_\_\_\_\_**, **\_\_\_\_\_\_\_\_\_\_\_\_** values, follow **\_\_\_\_\_\_\_\_\_\_\_\_** in a **\_\_\_\_\_\_\_\_\_\_\_\_**, choose between **\_\_\_\_\_\_\_\_\_\_\_\_**, (selection) and **\_\_\_\_\_\_\_\_\_\_\_\_** instructions in a loop (iteration).

Despite how limited this range of instructions appears, it describes everything that digital devices can compute.

|  |
| --- |
| **Sample answer:**  Computational algorithms (the kind that can run on **digital devices**) have relatively few ingredients because digital devices only have a few types of **instruction** that they can follow; the main things they can do are receive **input**, provide **output**, **store** values, follow **instructions** in a **sequence**, choose between **options**, (selection) and **repeat** instructions in a loop (iteration).  Despite how limited this range of instructions appears, it describes everything that digital devices can compute. |

**Activity 29:**student teams are issued pseudocode and/or flowchart questions from past HSC examinations.

Students:

* refer to the [Course specifications document](https://curriculum.nsw.edu.au/syllabuses/software-engineering-11-12-2022?tab=teaching-and-learning) to identify the control structures: sequence, selection and iteration within each question
* identify data that should be stored
* desk check these pseudocode questions to determine where logic errors exist
* rewrite these pseudocode algorithms into Python and run the program.

## Apply ‘divide and conquer’ and ‘backtracking’ as algorithmic design strategies

**Activity 30:** as a class watch [Santa's Dirty Socks (Divide and Conquer Algorithms) (3:40)](https://youtu.be/wVPCT1VjySA)

In the space provided below, **describe the process used to solve the problem.**

|  |
| --- |
| **Sample answer:**  Halve the number of boxes, compare each half by weighing them against each other, and halve again. |

Using your description (above), identify the control structures (sequence, selection and iteration) used in the solution in the space below.

|  |
| --- |
| **Sample answer:**  Sequence: Halve the boxes and weigh them.  Selection: Compare the halves to find the heaviest box.  Iteration: Keep halving and weighing the boxes until the item is found. |

In the space below, use these control structures to design and write an algorithm in pseudocode or flowchart that could be implemented in an automated dirty sock seeking program.

|  |
| --- |
| **Sample answer:**  Student’s responses may vary though should include an outer loop (iteration) which continues to halve, weigh and compare (selection) the halves. |

**Activity 31:** Searching and Sorting are 2 powerful aspects of computing.

Research the use of [Divide and Conquer](https://www.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/divide-and-conquer-algorithms) strategies to design searching and sorting algorithms.

From your research explain how this search or sort algorithm uses the 3 parts of the divide-and-conquer strategy.

|  |
| --- |
| **Sample answer:**  Divide the problem into a number of sub-problems that are smaller instances of the same problem.  Conquer the sub-problems by solving them recursively. If they are small enough, solve the sub-problems as base cases.  Combine the solutions to the sub-problems into the solution for the original problem. |

**Teacher note:** Big O notation is an important description of the efficiencies of [sorting and searching algorithms](https://www.linkedin.com/advice/1/how-do-you-apply-divide-conquer-strategies-sort-search) and should form part of a teacher discussion around these concepts.

### Assessment task guide part 2

Your teacher client has agreed that your software can include a tic-tac-toe game to play when students have finished their work or to assist in deciding who gets to go first in turn taking class activities.

**Activity** **32:** Tic-tac-toe

In the space below, describe tic-tac-toe.

|  |
| --- |
| **Sample answer:**   * It’s a 2-player game played on a 3 by 3 grid. * Each player takes it in turns to put a nought or a cross on the grid. * The first player to get 3 in a row wins the game. |

Studentsplaytic-tac-toe.

Figure 2 – blank tic-tac-toe board

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

**Activity 33:** write down the steps used to play tic-tac-toe.

|  |
| --- |
| **Sample answer:**  Draw the board, which is a 3 × 3 grid of squares.  This means it has 3 rows of 3 squares.  Player 1 goes first putting an ‘X’, into a square of the grid, in an attempt to have 3 of them in a row.  Player 2 puts down an ‘O’ to try to block the first player from creating a row of 3, or focus on creating their row of 3.  Ideally, the player can do both.  Take turns until one of the players has 3 of their symbols in a row: vertically, horizontally or diagonally. The first player to do this, wins. |

****The steps to play tic-tac-toe could begin with an IPO chart with ‘Completed Grid’ as the output and symbols ‘X’ and ‘O’ as the input.

Identify appropriate data structures that will be needed to create the grid.

Given all squares in the grid look the same, how will the computer know where each square is?

**Teacher note:** numbering each grid box will allow the player to choose where they are going to put their nought or cross. While it may be easier to start with zero instead of one thus matching with Python (a one-to-one match for less experienced programmers), it is an important learning opportunity to show that the index for an array is different to the value held inside the array.

Figure 3 – numbered tic-tac-toe board

A 3x3 square, with each square labelled 1-9


**Activity 34:** creating tic-tac-toe.

Open Python and make the grid using a list (array).

To begin making this list, they type in:

board= [“1”,”2”,”3”,”4”,”5”,”6”,”7”,”9”]

To display these numbers in the grid, it’s important to remember that lists store the location of each item starting from zero and not from one.

The stored location (index) of each item is: 0 1 2 3 4 5 6 7 8.

This can be visualised on the board with an array of containers labelled with indexes 0 to 8 with the contents being the numbers 1 to 9.

Use the **PRIMM (Predict, Run, Investigate, Modify and Make)** model to code the grid solution.

**Predict** what the following code will do:

print (board[0], board[1], board[2],)

|  |
| --- |
|  |

**Run** the code.

**Investigate** the code and how it displays 1 2 3.

|  |
| --- |
| **Sample answer:**  The number “1” is stored in the board list and referenced by the index, board [0].  The number “2” is stored in the board list and referenced by the index, board [1].  The number “3” is stored in the board list and referenced by index, board [2]. |

**Modify** the code to include the next row of the grid.

|  |
| --- |
| **Sample answer:**  print (board[0], board[1], board[2],)  print (board[3], board[4], board[5],) |

Students **Make** the final grid.

|  |
| --- |
| **Sample answer:**  print (board[0], board[1], board[2],)  print (board[3], board[4], board[5],)  print (board[6], board[7], board[8],) |

Run this code. You will notice that the numbers appear, but they might be difficult to see without grid lines.

In the space below, propose solutions to depicting the grid lines.

|  |
| --- |
| **Sample answer:**  print (board[0],”:” board[1],”:” board[2],)  print(“………’)  print (board[3],”:” board[4],”:” board[5],)  print(“………’)  print (board[6],”:” board[7],”:” board[8],) |

**Teacher note:** teachers may choose to use the PRIMM strategy again to build on student knowledge by gradually introducing the addition of the following text strings “:” and ……… to represent the vertical and horizontal borders of the cells.

**Make the game for 2 players**

**Teacher note:** the code provided so far enables one player. Students add variables to provide for two players

Two players are required for this game and students should consider what data type will be used to store their names.

Use the input to ask a question.

The player1 variable will store player 1’s name and the player2 variable will store player 2’s name.

player1= input ( “Enter the name of the first player: ”)

player2= input ( “Enter the name of the second player: ”)

**Playing the game**

The game is set up and ready to play.

The first player is asked where they would like to add a cross to the grid. They do this by telling the program the number of the location.

The board is changed so that a cross is added at this location on the grid.

Discuss how this could be coded.

**Teacher note:** when coding, the number given is one more than the storage location in the board list.

For example, if a player types 1 then this will be located at 0 in the board list and if they say 7, then this would be located at 6 in the board list.

Add this code to the code you have produced to start playing the game.

|  |
| --- |
| #Tic tac toe  print (“Welcome to Tic-tac-toe”)  board= [“1”,”2”,”3”,”4”,”5”,”6”,”7”,”9”]  #Make the Grid  print (board [0],”:” board [1],”:” board [2],)  print (“………’)  print (board [3],”:” board [4],”:” board [5],)  print (“………’)  print (board [6],”:” board [7],”:” board [8],)  #Player names  player1= input (“Enter the name of the first player:”)  player2= input (“Enter the name of the second player:”)  #Play Tic tac toe  Print ( “hello”, player1)  Xloc=int (input( “Where do you want to place a cross ( choose from 1-9) : “))  board [Xloc-1]=”X”  print (board [0],”:” board [1],”:” board [2],)  print (“………’)  print (board [3],”:” board [4],”:” board [5],)  print (“………’)  print (board [6],”:” board [7],”:” board [8],) |

**Predict** what the code above will do:

|  |
| --- |
|  |

**Run** the code.

Notice in this game, the board will be displayed repeatedly.

A sub-program called a procedure will make the code a lot more efficient.

Create a procedure to display the grid. Then ‘call’ this procedure by writing the name of the procedure and the grid will be displayed.

|  |
| --- |
| #Tic tac toe  print ("Welcome to Tic\_tac\_toe")  board=["1","2","3","4","5","6","7","8","9"]  #Procedure  def make\_board ():  print (board[0],":", board[1],":",board[2],)  print (".........")  print (board [3],":",board [4],":", board [5],)  print (".........")  print (board [6],":",board [7],":",board [8],)  #Calling the procedure  make\_board()  #Player names  player1 = input ("Enter the name of the first player:")  player2 = input ("Enter the name of the second player:")  #Play Tic tac toe  print ("Hello", player1)  Xloc=int (input("Where do you want to place a cross ( choose from 1-9) : "))  board [Xloc-1]= "X"  make\_board ( ) |

**Predict**, **Run** and **Investigate** the code then consider how to **Modify** the code to enable 2 players.

**Activity 35:** work in pairs to consider how to modify the code to enable 2 players to play the game.

|  |
| --- |
| **Sample answer:**  One solution is to use while loops.  Two while loops, the first nested inside the second will enable the game to keep being played and track whose turn it is.  The first while loop sets a condition that the game carries on until a player wins, and the second while loop decides whose turn it is and alternates between the 2 players. |

|  |
| --- |
| #Tic tac toe  print ("Welcome to Tic\_tac\_toe")  board=["1","2","3","4","5","6","7","8","9"]  #Player names  player1 = input ("Enter the name of the first player:")  player2 = input ("Enter the name of the second player:")  no\_win=True  turn=1  #Procedure  def make\_board ():  print (board[0],":", board[1],":",board[2],)  print (".........")  print (board [3],":",board [4],":", board [5],)  print (".........")  print (board [6],":",board [7],":",board [8],)  #Calling the procedure  make\_board()  while no\_win:  while turn==1 and no\_win:  print ("Hello", player1)  Xloc=int (input("Where do you want to place a cross ( choose from 1-9) : "))  board [Xloc-1]= "X"  make\_board ( )  turn=2  while turn==2 and no\_win:  print ("Hello", player2)  Xloc=int (input("Where do you want to place a cross ( choose from 1-9) : "))  board [Xloc-1]= "0"  make\_board ( )  turn=1 |

**Predict**, **Run** and **Investigate** the code to determine what else it requires.

**Activity 36:** winner

This code is yet to determine a winner. It currently alternates between each player.

How is a winner determined?

|  |
| --- |
| **Sample answer:**  The game is won and over when a player gets 3 noughts or crosses in a row. |

What code needs to be added to decide who won the game?

|  |
| --- |
| **Sample answer:**  Code that checks for all winning combinations. |

Plot then list all the possible combinations of numbers that can win.

Figure 4 – numbered board 1–9

|  |  |  |
| --- | --- | --- |
| **1** | **2** | **3** |
| **4** | **5** | **6** |
| **7** | **8** | **9** |

What are the possible winning combinations?

|  |
| --- |
| **Sample answer:**  1,2,3 or 4,5,6 or 7,8,9 or 1,4,7 or 2,5,8 or 3,6,9 or 1,5,9 or 3,5,7 |

How could these combinations be stored?

|  |
| --- |
| **Sample answer:**  These combinations could be stored in a 2D array or in Python as list of lists. |

How is this list of lists used to find a winner?

|  |
| --- |
| **Sample answer:**  Compare this list of lists with the locations chosen by the player.  If the chosen locations match a winning combination, then the player has won. |

How many lists are needed?

|  |
| --- |
| **Sample answer:**  Three lists. The first list contains the winning locations. The other 2 lists contain the choices made by player1 and player2.  win\_combo=[[0,1,2],[3,4,5],[6,7,8],[0,3,6],[1,4,7],[2,5,8],[0,4,8],[2,4,6]]  play1\_loc\_list=[]  play2\_loc\_list=[] |

**Teacher note:** remember that numbers 1,2,3 are found at locations 0,1,2

After each turn the players chosen location needs to be kept in memory (stored in the list) to see if it has made the winning combination.

What function can be used to add the location to the list?

|  |
| --- |
| **Sample answer:**  The append function.  play1\_loc\_list.append(loc\_attempt)  Python’s append() function inserts a single element into an existing list. The element will be added to the end of the old list rather than being returned to a new list. |

What if a player chooses a combination of numbers that are the same as the winning combinations but in a different order?

|  |
| --- |
| **Sample answer:**  If a player chooses numbers in a different order, you also need to sort the list so that it in ascending order (the same as the stored winning combo list).  play1\_loc\_list.sort() |

**Activity 37:** In the space below evaluate the use of subprograms in your Tic\_ tac\_toe algorithm

|  |
| --- |
| Students should make reference to the make board procedure and how it can be called in multiple places. |

|  |
| --- |
| **# Complete Tic tac toe code**  win\_combo=[[0,1,2],[3,4,5],[6,7,8],[0,3,6],[1,4,7],[2,5,8],[0,4,8],[2,4,6]]  play1\_loc\_list=[]  play2\_loc\_list=[]  print ("Welcome to Tic\_tac\_toe")  board=["1","2","3","4","5","6","7","8","9"]  #Player names  player1 = input ("Enter the name of the first player:")  player2 = input ("Enter the name of the second player:")  no\_win=True  turn=1  #Procedure  def make\_board ():  print (board[0],":", board[1],":",board[2],)  print (".........")  print (board [3],":",board [4],":", board [5],)  print (".........")  print (board [6],":",board [7],":",board [8],)  #Calling the procedure  make\_board()  while no\_win:  while turn==1 and no\_win:  print ("Hello", player1)  Xloc=int (input("Where do you want to place a cross ( choose from 1-9) : "))  board [Xloc-1]= "X"  loc\_attempt=Xloc-1  play1\_loc\_list.append(loc\_attempt)  play1\_loc\_list.sort()  for i in range (0,len(win\_combo)):  if win\_combo[i]==play1\_loc\_list:  print("you have won")  no\_win=False  make\_board ( )  turn=2  while turn==2 and no\_win:  print ("Hello", player2)  Xloc=int (input("Where do you want to place a cross ( choose from 1-9) : "))  board [Xloc-1]= "0"  loc\_attempt=Xloc-1  play2\_loc\_list.append(loc\_attempt)  play2\_loc\_list.sort()  for i in range (0,len(win\_combo)):  if win\_combo[i]==play2\_loc\_list:  print("you have won")  no\_win=False  make\_board ( )  turn=1 |

**Teacher note:** an [alternate solution](https://www.educative.io/answers/how-and-when-to-use-any-and-all-in-python) would include replacing the current:

for i in range (0, len(win\_combo)): and the if statement with the below two lines of code:

for sublist in win\_combo:

if all(elem in play1\_loc\_list for elem in sublist):

**Activity 38:** Examine sample code

**Predict** what this complete tic-tac-toe code will do?

|  |
| --- |
|  |

**Run** the code in Python or [Trinket](https://trinket.io/).

Do the inputs and outputs accord with your expectations? If not, explain the likely cause of this mismatch.

|  |
| --- |
|  |

**Teacher note:** by discussing the given problem, how to solve and debug the issues, or simply when assigning students to ‘find’ a solution to the identified problem, teachers reinforce ‘permission’ for students to produce a solution that may not work as expected but then have the opportunity to fix the problem.

This is both an iterative process and a demonstration of the principles of a growth mindset which is important in progressing through this course.

**Investigate** the code:

Identify by highlighting (or referring to code line numbers) all control structures: Sequence, Selection and Iteration (Counted and pre-test loops).

‘Reverse engineer’ the code by rewriting it into algorithms (pseudocode and/or a flowchart) including the use of sub routines and procedures.

Identify by highlighting (or referring to code line numbers) all data structures: data types, Variables, Lists (and lists of lists). This will be used to make a data dictionary.

**Teacher note:** these documents, flowcharts/pseudocode and data structures, will be referred to during future exercises.

**Extension activity 1:** write down the rules to play Sudoko.

****Draw an empty 9 × 9 Sudoku grid and fill it with numbers that obey the rules.

Notice it is quite challenging. Discuss the method you used to fill the grid.

Play a pre-filled game of Sudoku and write down the steps you used to solve the puzzle.

This could begin with an IPO chart with ‘Completed Grid’ as the output and integers 1–9 as the input.

The **‘**Process’ column of the IPO chart will include the steps used to solve the problem. Your description of the process used to investigate each cell within the grid to see which number fits is a process of backtracking.

Identify the 3 control structures (Sequence, Selection and Iteration) within your description, and use this to write a pseudocode description or flowchart of the Sudoku solution process.

Rewrite their pseudocode or flowchart in Python. These [video tutorials](https://www.techwithtim.net/tutorials/python-programming/sudoku-solver-backtracking/) may assist here.

Sudoku created using Excel OR using [Python using OOP](https://schoolsnsw.sharepoint.com/sites/Stage6-ComputingWritersandReviewers/Shared%20Documents/SE%20Year%2011%20-%20Programming%20Fundamentals/Sudoku%20Puzzle%20(inventwithpython.com)).

**Extension activity 2:** students follow along the [video series](https://youtu.be/zp4J0qGgN_w) to make a Sudoku generator in Excel using Macros.

**Teacher note:** pause the video when the code is being written and used, and/or provide the code to students.

**Extension activity 3:** students research the 8 queens puzzle.

[Students watch Eight Queens puzzle](https://youtu.be/jPcBU0Z2Hj8).

Students use their descriptions of the process to design an algorithm to solve the 8 queens puzzle**.**

**Extension activity 4:** students research Backtracking.

Students watch [Backtracking (Think Like a Programmer) (13:01)](https://youtu.be/gBC_Fd8EE8A) video.

### Develop structured algorithms using pseudocode and flowcharts, including the use of subprograms

**Activity** **39:** Pseudocode and flowcharts

The Primary School Teacher client has requested that a simple calculator be part of the educational software you are making.

**Teacher note:** teacher could also provide students with a ‘broken’ calculator program to find and fix problems.

Using the Course Specification document, design an algorithm that will request 2 numbers from a user, add these 2 numbers and output the answer.

This can be done in pseudocode or as a flowchart.

|  |
| --- |
| **Sample answer:**  An algorithm in the form of a flowchart that will request 2 numbers from a user, add these 2 numbers and output the answer. |

Design an algorithm that will request 2 numbers from a user and subtract the second number from the first.

|  |
| --- |
| **Sample answer:**  An algorithm in the form of a flowchart that will request 2 numbers from a user and subtract the second number from the first. |

****Describe which parts of these algorithms are the same and which should be kept as the main algorithm?

|  |
| --- |
| **Sample answer:**  Only the process is different. |

How could this algorithm be modified to give the user the choice to add and subtract?

|  |
| --- |
| **Sample answer:**  An algorithm that will request 2 numbers from a user and give the user the choice to add and subtract. |

The sample response on the previous page demonstrates the use of user input to assist in providing options for the process of selection.

**Activity 40:** rewrite all flowcharts as pseudocode (and vice versa), using the Course Specifications.

|  |
| --- |
| **Sample answer:**  BEGIN calculator  Get Num1  Get Num2  Get menuOption  IF menuOption = “Add” THEN  Answer = Num1 + Num2  ELSE  Answer = Num1 - Num2  ENDIF  Display Answer  END calculator |

**Activity 41:** increasing functionality

As a class, discuss what will happen to the code as more functionality is introduced. Should the calculator also multiply and divide?

The code will get progressively longer and more confusing.

**Activity 42:** flowcharts

****Refer to the Course Specifications to create a flowchart that provides a multi-way selection of the case where the menuOption is Add or Subtract or Divide or Multiply.

|  |
| --- |
| **Sample answer:**  With reference to page 16 of the Course Specifications document, students should be able to create a multi-way selection where the expression ‘diamond’ is labelled menuOption and the Choice A is Add, Choice b is Subtract, and so on. |

**Activity 43:** compare and contrast

****Compare and contrast the multi-way selection algorithm with the nested IF algorithm using the ‘Simple calculator’ and the ‘Triangle recognition’ programs to explain the differences.

In the space below, record the differences between the multi-way selection algorithm and the nested IF algorithm.

|  |
| --- |
| **Sample answer:**  In multi-way selection algorithm, there can be several possible choices or cases. The path taken is determined by the evaluation of the expression. Once a relevant path has been determined and followed, execution of this expression finishes. Only one process is executed as a result of the implementation of the multi-way selection. Multi-way selection is often referred to as a case structure.  A nested IF allows the testing of multiple conditions with only one process executed  Both nested IF and multi-way IF-else can be used to control the flow of your program, but they are used in different situations. Nested IF is useful when you have multiple conditions that need to be checked at the same time, while multi-way IF-else is useful when you have multiple conditions that can be checked one after the other. |

**Activity 44:** Evaluate the use of subprograms in your calculator and/or triangle recognition algorithms.

|  |
| --- |
| **Sample answer:**  Students should refer to their flowchart algorithms in response to this question.  They should note that as the program gets bigger the use of subprograms makes it easy to follow the logic and to make correction if needed. |

### Use modelling tools including structure charts, abstraction, and refinement diagrams to support top-down and bottom-up design

**Activity 45:** structure chart

****Using the Course Specifications design a structure chart for their proposed educational software which includes:

* a log in procedure to access the program
* the simple calculator (with add, subtract, divide and multiply processes)
* the triangle recognition program (with equilateral, isosceles, scalene and right-angle outputs)
* the tic-tac-toe game

Students [Think-Pair-Share](https://app.education.nsw.gov.au/digital-learning-selector/LearningActivity/Card/645#.Y_Kr3SagGrY.link) to complete the structure chart below by adding parameters, flags, decisions and repetitions. Activity 56 should be delivered in conjunction with this task.

|  |
| --- |
| A structure chart. |

### Analyse the logic and structure of written algorithms

**Activity 46:** visit the sample [online HSC Examination](https://fam.hsconline.nesa.nsw.edu.au/) and inspect the following algorithm. Each line of the algorithm has been numbered for reference.

1. BEGIN determineGrade

2. Get Mark

3. CASEWHERE Mark

4. >=80: Grade "H"

5. >=50: Grade "P"

6. OTHERWISE: Grade = "N"

7. ENDCASE

8. Display Grade

9. End determineGrade

#### Determining inputs and outputs

**Activity 47:** determining inputs and outputs

****What are the inputs and outputs?

|  |
| --- |
| **Sample answer:**  Inputs: Marks (given the conditional statements it could be presumed that the mark is out of 100).  Outputs: Grades (H, P or N. Students may infer that these correlate to High, Pass and Nil). |

#### Determining the purpose of the algorithm

**Activity 48:** determine the purpose**** of the above algorithm.

|  |
| --- |
| **Sample answer:**  This algorithm is an example of a multi-way selection.  Its purpose is to get a mark (possibly out of 100) and determine the grade. |

#### Desk checking and peer checking

**Activity 49:** students should desk check every algorithm they encounter. Many examples can be extracted from past HSC Software Design and Development (SDD) examination papers.

****This can work best in a collaborative environment where an algorithm is written on a whiteboard and students discuss what is happening in every line of pseudocode.

This involves identifying and labelling the variables (in this case Mark and Grade) and putting test data through the algorithm to analyse the language and structure.

#### Determining connections of written algorithms to other subroutines or functions

**Activity 50:** determining connections

How might the above algorithm from Activity 46 be incorporated into a larger program?

|  |
| --- |
| **Sample answer:**  The determined grades could be part of a larger program that opens the teachers mark book file and reads through the names and marks, allocates a grade dependent upon the case statements (and conditions) and writes that grade back to the mark book.  Students might consider how the teacher may wish to have their own grading criteria and what that may look like in code. |

****An engaging activity to assist students without experience in computer programming that may be struggling with pseudocode or flowcharts and variables is to issue 2 paper cups.

One labelled ‘X’ and the other labelled ‘Y’ (teachers should point out that these are not very descriptive names for variables).

The teacher places a piece of paper with the integer ‘5’ written on it into the cup labelled ‘X’ and another with the integer ’8’ written on it into the cup labelled ’Y’.

**Activity 51:** Students are encouraged to write the pseudocode to swap the values in the cups.

X = 5

Y = 8

The assignment statement ‘X = 5’ does not mean ‘equals to’, a common mistake by novice coders, but rather 5 goes into the variable X.

Many students will write:

1. X = 5
2. Y = 8
3. X = Y

A quick desk check reveals the problem:

Table 11 – desk check

|  |  |
| --- | --- |
| X | Y |
| 5 |  |
|  | 8 |
| 8 |  |

Notice each line of code has a row.

Both values are the same and a swap hasn’t taken place.

A student physically demonstrating the attempt at swap will need to understand the rule that the variable is overwritten. Students can consider a possible solution.

Students with some coding experience or insight into the logic will immediately see that a third cup or variable called 'temp’ will be required.

The solution can be written and demonstrated physically using the 3 cups.

Table 12 – solution using the 3 cups

|  |  |  |
| --- | --- | --- |
| X | Y | temp |
| 5 |  |  |
|  | 8 |  |
|  |  | X |
| Y |  |  |
|  | temp |  |

1. X = 5
2. Y = 8
3. temp= X
4. X=Y
5. Y = temp

This Swap algorithm could be useful as a subroutine in a range of programs.

****In the space below suggest where the Swap algorithm could be useful as a subroutine:

|  |
| --- |
| **Sample answer:**  A card game, inserting a new node into a list, basic sort algorithm. |

### Identify procedures and functions in an algorithm

When programming an application, it is good practice to avoid long repetitive programs.

When a big application is broken down into several sub programs it makes it much easier to understand.

Well-structured applications are easier to upgrade and easier to fix if they go wrong.

Sub-programs include procedures and functions.

**Teacher note:** students may read that ‘Functions return values back to the main program. Procedures do not return a value back to the main program’. Many languages differentiate procedures from functions by saying that functions return a value, but procedures don't. Since languages like C/C++ and Java have a void or empty return type the distinction doesn’t makes sense.

**Activity 52:** cloze passage

**information**, **called, operation, function, run, arguments**

A **\_\_\_\_\_\_\_\_\_\_** is an independent, named snippet of code that performs a specific **\_\_\_\_\_\_\_\_\_\_.** It can be **\_\_\_\_\_\_\_\_\_\_** or be **\_\_\_\_\_\_\_\_\_\_** by referring to it by name with any **\_\_\_\_\_\_\_\_\_\_** called **\_\_\_\_\_\_\_\_\_\_** it needs to do its job.

|  |
| --- |
| **Sample answer:**  A **function** is an independent, named snippet of code that performs a specific **operation.** It can be **run** or be **called** by referring to it by name with any **information** called **arguments** it needs to do its job. |

You will have had lots of experience calling built-in functions like **input** or **int**.

Both **input** and **int** take a single argument.

For input this is the message to display to the user, for example **'Enter the length of side a:'**

For **int** it is the value to be converted into an **integer**, for example **int(“10”)**.

Some languages and textbooks use different terminology for functions including: routines, subroutines, subprograms, procedures, messages or methods.

**Activity 53:** Functions

****In the space below describe the advantages of using functions.

|  |
| --- |
| **Sample answer:**  Functions enable the same snippet of code to be called again and again, rather than having to be written out multiple times. If changes need to be made to the code, it only has to be in one place rather than each copy. Functions can be used to hide complicated code that doesn’t have to be thought about while being used. |

To call a function you need to know how many arguments the function is expecting and what type of values they are.

For example, the **abs** built-in function takes one argument which must be an integer or float:

print(abs(4))

print(abs(-11))

print(abs(-3.14159))

**Activity 54:** identify procedures and functions

****Use your tic-tac-toe program and ‘reversed engineered algorithm’ to identify all the procedures and functions used.

|  |
| --- |
| **Sample answers:**  Make\_board was a procedure.  Functions included print, sort and append.  Students should consider what control structures are ‘inside’ functions. For example, using the function sort in Python.  Python's default sort uses Tim Sort, which is a combination of both merge sort and insertion sort. |

**Activity 55:** refer to the Course specifications document to create a main algorithm that calls on subroutines that add and subtract.

|  |
| --- |
| A main algorithm that calls on subroutines that add and subtract.  A main algorithm that calls on subroutines that add and subtract. |

**Activity** **56:** BournetoCode

****Visit [BournetoCode](https://bournetocode.com/projects/GCSE_Computing_Fundamentals/pages/3-2-7-subroutines.html) and complete the online activities on subroutines including Python coding using Trinket.

**Activity** **57:** Sub Procedures

As a class, watch [Python Programming Sub Procedures (11:13).](https://www.youtube.com/watch?v=K05_CozFPeE)

Use Python to create a simple calculator.

In python there isn't much difference between a procedure and a function.

**Teacher note:** students should refer to their pseudocode/flowcharts to complete this activity.

**Activity 58:** Statements and Functions

As a class watch [Programming Basics: Statements & Functions (11:56)](https://youtu.be/l26oaHV7D40).

Speculate and discuss the statements and functions used in a computing game you have played.

**Teacher note:** teachers may decide to deliver the tic-tac-toe activity after the triangle and simple calculator tasks.

### Experiment with object-oriented programming, imperative, logic and functional programming paradigms

There are several kinds of major programming paradigms:

* [Imperative](http://www.cs.ucf.edu/~leavens/ComS541Fall97/hw-pages/paradigms/major.html#imperative)
* [Logical](http://www.cs.ucf.edu/~leavens/ComS541Fall97/hw-pages/paradigms/major.html#logical)
* [Functional](http://www.cs.ucf.edu/~leavens/ComS541Fall97/hw-pages/paradigms/major.html#functional)
* [Object-Oriented](http://www.cs.ucf.edu/~leavens/ComS541Fall97/hw-pages/paradigms/major.html#object)

‘Any solvable problem using one of these paradigms can be solved using the others.

Some problems lend themselves more naturally to specific paradigms.’ (University of Central Florida, n.d.)

**Activity 59:** [Jigsaw](https://app.education.nsw.gov.au/digital-learning-selector/)

****Students are allocated one of 4 teams, each representing a different paradigm.

Each team research and reports to their class upon their allocated paradigm and its use in developing a game of tic-tac-toe.

The class presentation or report contains the following headings:

* Introduction
* Description
* Advantages and disadvantages
* Languages
* Tic-tac-toe example
* Conclusion including recommendation as a paradigm to create a tic-tac-toe game.

The following pages provide sample answers.

|  |
| --- |
| **Sample answer:**  [The Imperative paradigm](https://www.ionos.com/digitalguide/websites/web-development/imperative-programming/)  **Introduction**  The imperative programming paradigm uses control structures and data structures (Do + Store). The ‘do’ includes sequence, selection and iteration which can be represented using algorithms and the store (data structures) which includes the data types, variables, arrays. Computations are performed through a guided sequence of steps, in which these variables are referred to or changed. The order of the steps (algorithms) is crucial because a given step will have different consequences depending on the current values of variables when the step is executed.  **Description**  Popular programming languages have often been imperative ones because they closely resemble how the computer works, so the programmer is much closer to the machine.  **Advantages**   * Efficient * Close to the machine * Popular * Familiar   **Disadvantages**   * Programs can be complex to understand or prove. * Debugging can be harder. * The order of the control flow is crucial, though this isn’t always the best way to solve the problem.   **Languages**  Fortran, Java, Pascal, ALGOL, C, C#, C++, Assembler, BASIC, COBOL, Python, Ruby  **Tic-tac-toe example:**  An example can be found on the ‘[Build a Tic-Tac-Toe Game With Python and Tkinter](https://realpython.com/tic-tac-toe-python/)’ webpage.  **Conclusion including recommendation as a paradigm to create a tic-tac-toe game.** |
| **Sample answer:**  The logical paradigm  **Introduction**  The logical paradigm takes a declarative approach to problem-solving. Various logical assertions about a situation are made, establishing all known facts. Then queries are made. The role of the computer becomes maintaining data and logical deduction.  **Description**  A logical program is divided into 3 sections:   * a series of definitions/declarations that define the problem * statements of relevant facts * statement of goals in the form of a query.   Any deducible solution to a query is returned.  The definitions and declarations are constructed entirely from relations, ie X is a member of Y, or X is in the internal between a and b etc.  **Advantages**  The system solves the problem, so the programming steps themselves are kept to a minimum.  Proving the validity of a given program is simple.  **Disadvantages**  It can be d**ifficult to see what is happening computationally. Models that are non-intuitive are used.**  **Languages**  Prolog, answer set programming (ASP) and Datalog.  **Tic-tac-toe examples:**  [In Prolog](https://courses.cs.washington.edu/courses/cse341/03sp/slides/PrologEx/tictactoe.pl.txt)  [In swish](https://swish.swi-prolog.org/p/Tic-Tac-Toe.swinb)  **Conclusion including recommendation as a paradigm to create a tic-tac-toe game.** |
| **Sample answer:**  [The Functional paradigm](https://www.geeksforgeeks.org/functional-programming-paradigm/)  **Introduction**  The functional programming paradigm views all subprograms as functions in the mathematical sense-informally, they take in arguments and return a single solution. The solution returned is based entirely on the input, and the time at which a function is called has no relevance. The computational model is therefore one of function application and reduction.  **Description**  Functional languages are created based on the functional paradigm. Such languages permit functional solutions to problems by permitting a programmer to treat functions as first-class objects (they can be treated as data, assumed to have the value of what they return; therefore, they can be passed to other functions as arguments or returned from functions).  **Advantages**  The high level of abstraction, especially when functions are used, supresses many of the details of programming and thus removes the possibility of committing many classes of errors.  The lack of dependence on assignment operations, allows programs to be evaluated in many different orders. This evaluation order independence makes function-oriented languages good candidates for programming massively parallel computers.  The absence of assignment operations makes the function-oriented programs much more amenable to mathematical proof and analysis than are imperative programs, because functional programs possess referential transparency.  **Disadvantages**  Perhaps less efficient.  Problems involving many variables, or a lot of sequential activity are sometimes easier to handle imperatively or with object-oriented programming.  **Languages**  Common Lisp, Scheme, Clojure, Wolfram Language, Racket, Erlang, Elixir, OCaml, Haskell, and F#.  **Tic-tac-toe example**  [In Haskell](https://gist.github.com/momirza/aec1ee5e86f315e85c5cd84867d2ff86)  [In Scheme](http://people.eecs.berkeley.edu/~bh/ssch10/ttt.html)  **Conclusion including recommendation as a paradigm to create a tic-tac-toe game.** |
| **Sample answer:**  [Object-Oriented](https://www.educba.com/object-oriented-programming-paradigm/)  **Introduction**  Object Oriented Programming (OOP) is a paradigm in which real-world objects are each viewed as separate entities having their own state, which is modified only by built-in procedures, called methods. Because objects operate independently, they are encapsulated into modules which contain both local environments and methods. Communication with an object is done by message passing.  **Description**  Objects are organised into classes, from which they inherit methods and equivalent variables. The object-oriented paradigm provides key benefits of reusable code and code extensibility.  **Advantages**  A new class may be derived from another class (called a base class or superclass) by a mechanism called inheritance. The derived class inherits all the features of the base class: its structure and behaviour (response to messages). In addition, the derived class may contain additional state (instance variables) and may exhibit additional behaviour (new methods to respond to new messages). The derived class can also override behaviour corresponding to some of the methods of the base class: there would be a different method to respond to the same message. Also, the inheritance mechanism is allowed even without access to the source code of the base class.  The ability to use inheritance is the single most distinguishing feature of the OOP paradigm. Inheritance gives OOP its chief benefit over other programming paradigms - relatively easy code reuse and extension without the need to change existing source code.  The mechanism of modelling a program as a collection of objects of various classes and describing many classes as extensions or modifications of other classes, provides a high degree of modularity. Ideally, the state of an object is manipulated and accessed only by that object's methods. (Most O-O languages allow direct manipulation of the state, but such access is stylistically discouraged). In this way, a class' interface (how objects of that class are accessed) is separate from the class' implementation (the actual code of the class' methods). Thus, encapsulation and information hiding are inherent benefits of OOP.  **Languages** Ada, ActionScript, C++, Common Lisp, C#, Dart, Eiffel, Fortran 2003, Haxe, [Java](https://people.cs.pitt.edu/~eth13/cs0007/Project4.pdf), JavaScript, Kotlin, logo, MATLAB, Objective-C, Object Pascal, Perl, PHP, Python, R, Raku, Ruby, Scala, SIMSCRIPT, Simula, Smalltalk, Swift, Vala and Visual Basic.NET.  [**Tic-tac-toe example**](https://codereview.stackexchange.com/questions/274308/tic-tac-toe-in-python-with-oop)  **Conclusion including recommendation as a paradigm to create a tic-tac-toe game.** |

**Teacher note:** the second focus area of the Software Engineering Course is OOP. The following activities could be delivered as an introduction to that area or extended and integrated into an assessment task that combines the 2 focus areas.

**Programming Fundamentals and Object-Oriented Programming**

**Activity 60: PRIMM**

Read and predict using Al Sweigart’s ‘Tic-Tac-Toe' on the '[Programming and Classes](https://inventwithpython.com/beyond/chapter15.html)' webpage.

**Teacher note:** because the code is linked to the site, students will simply copy and paste the answers to these questions. Teachers should consider providing the Python code only and have a reference to the source but not the direct link.

**Predict** what the code will do.

|  |
| --- |
| Creates a board to play tic tac toe and allows players to take turns until there is a winner. |

**Run** the code and confirm or challenge their prediction.

**Investigate** the code by:

* Describing how it works.

|  |
| --- |
| Students identify the data and control structures including if (branching) and while (loop). |

* Identifying all the functions in the code and what they do.

|  |
| --- |
| In the **non OOP** tictactoe.py code version:  The main() function contains the code that creates a new board data structure (stored in the gameBoard variable) and calls other functions in the program.  Students should recognise that some functions can be identified as verbs followed by () e.g.: getBlankBoard(). |

* Identifying the parameters.

|  |
| --- |
| Many of the functions accept the variable **board** as their first parameter. These functions are related to each other in that they all operate on a common data structure. |

Add these descriptions to the code using # internal documentation.

Provide a brief description of how this code works.

|  |
| --- |
| **Sample answer:**  This code represents the 9 spaces on a tic-tac-toe board with strings 1 through 9, and values are the strings 'X','O', or ' '.  The numbered spaces are in the same arrangement as a phone’s keypad. |

Identify the functions and parameters in the tictactoe.py code and explain what each one does.

|  |
| --- |
| **Sample answers:**  **main ()** function: contains the code that creates a new board data structure (stored in the gameboard variable) and calls other functions in the program.  **getBlankBoard()** function: returns a dictionary with the nine spaces set to ‘ ‘ for a blank board.  **getBoardStr()** function: accepts a dictionary representing the board and returns a multiline string representation of the board that can be printed to the screen. This is what renders the tic-tac-toe board’s text that the game displays.  **isValidSpace()** function: returns ‘True’ if it’s passed a valid space number and that space is blank.  **isWinner()** function: sets parameters to accept a board dictionary and either: 'X' or 'O' to determine whether that player has 3 marks in a row on the board.  **isBoardFull()** function: determines whether the board has no blank spaces, meaning the game has ended.  **updateBoard()** function: sets parameters to accept a board dictionary, a space and a player’s ‘X’ or ‘O’ mark, and updates the dictionary.  Many of the functions accept the variable board as their first parameter. |

Since many of the functions accept the variable board as their first parameterthese functions are related to each other in that they all operate on a common data structure.

When several functions in the code operate on the same data structure, it’s often best to group them together as the **methods** and **attributes** of a **class**.

The tictactoe.py program can be redesigned to use a TTTBoard **class** that will store the board dictionary in an **attribute** named spaces.

The functions that had ‘board’ as a parameter will become methods of the TTTBoard class and use the self parameter instead of a board parameter.

**Activity 61: PRIMM**

Read and predict using Al Sweigart’s ‘Tic-Tac-Toe\_oop.py' on the '[Programming and Classes](https://inventwithpython.com/beyond/chapter15.html)' webpage.

**Teacher note:** because the code is linked to the site, students will simply copy and paste the answers to these questions. Teachers should consider providing the Python code only and have a reference to the source, but not the direct link.

**Predict** what the code will do.

|  |
| --- |
| **Sample answer:**  The same as the non oop version |

**Run** the code and confirm or challenge their prediction.

**Investigate** the code by:

Describing how it works.

|  |
| --- |
| The code that was in getBlankBoard() is instead used in the TTTBoard class’s \_init\_method. The other functions from the non oop version have been made methods |

Identifying all the classes, methods and attributes.

|  |
| --- |
| Students should recognise the class TicTacToeBoard and the methods identified by the (self) references. |

Add these descriptions to the code using # internal documentation.

Open this [code comparison of Tic Tac Toe written in Python: Non-OOP and OOP versions.](https://autbor.com/compareoop/#diff)

****In the space below compare the non-OOP Python code with the OOP Python code.

|  |
| --- |
| **Sample answer:**  The non-OOP Python code and the OOP Python code have output that looks identical. The code that was used to be in getBlankBoard() has been moved to the TTTBoard class’s\_\_init\_\_()method, because they perform the same task of preparing the board data structure.  The other functions have been converted into methods, with the self-parameter replacing the old board parameter because they have similar purposes – they’re both blocks of code that operate on a tic-tac-toe board data structure.  When the code in these methods needs to change the dictionary stored in the \_spaces attribute, the code uses self \_spaces. When the code in these methods need to call other methods, the calls would also begin with ‘self’ and a period.  The \_spaces attribute begins with an underscore, which means that only the code inside the methods of TTTBoard should access or modify it.  Code outside the class should only be able to modify \_spaces indirectly by calling methods that modify it. |

****Tic-tac-toe is a small program. What if this program were tens of thousands of lines long, with hundreds of different functions?

|  |
| --- |
| **Sample answer:**  A program with a few dozen classes would be easier to understand than a program with several hundred different functions.  OOP breaks down a complicated program into easier to understand modules. |

## Data for software engineering

### Investigate the use of number systems for computing purposes, including binary, decimal and hexadecimal

Computers don’t understand words or numbers the way humans do. Modern software allows the end user to ignore this, but at the lowest levels of computers, everything is represented by a binary electrical signal that registers in one of 2 states: on or off.

To make sense of complicated data, your computer must encode it in binary.

This representation at different levels is a good example of abstraction.

Binary is a base 2 number system. Base 2 means there are only 2 digits: 1 and 0 which correspond to the on and off states your computer can understand.

The words ’binary digits’ can be abbreviated to read ‘bits’.

All students will be familiar with base 10 —, the decimal system.

Decimal makes use of 10 digits that range from 0 to 9, and then wraps around to form two-digit numbers, with each digit being worth ten times more than the last (1, 10, 100, and so on).

**Activity 62:** teacher displays a board with columns that represent the values of the decimal system and provides an example: 59312 with extended working. This will provide insight into the algorithm beneath all number systems including binary, octal and hexadecimal.

Table 13 – values of the decimal system

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 104 | 103 | 102 | 101 | 100 |
| 10000s | 1000s | 100s | 10s | 1s |
| 5 | 9 | 3 | 1 | 2 |
| 5 × 10000 | 9 × 1000 | 3 × 100 | 1 × 10 | 2 × 1 |

This shows that the number 59312 can be expressed as:

(5 × 10000) + (9 × 1000) + (3 × 100) +(1 × 10) + (2 × 1)

**Activity 63:** scenario

****Imagine you are describing this counting process to an alien unfamiliar with our number system.

Can you describe the pattern used to count past 10, past 100 and past 1000 in the decimal system?

|  |
| --- |
| **Sample answer:**  Some explanation of place values will be needed. Starting from the right hand (units column) counting from 0 to 9 occurs until the numeral 9 is reached. A 0 is then put into place and a 1 is carried into the next column to the left, the tens column. This process is repeated in each column. |

Can this pattern be described using an algorithm?

|  |
| --- |
| **Sample answer:**  Yes. |

What control structures (sequence, selection, iteration) are used?

|  |
| --- |
| **Sample answer:**  All control structures can be used: sequence of process steps to add, selection of the condition when column number > 9 and iteration to loop through each count and each column. |

Write this algorithm using pseudocode in the space below.

Swap your algorithm with a classmate and desk check each other’s to see if they work.

|  |
| --- |
| **Sample answer:**  The exercise here is less to do with formulating an exact algorithm and more an introduction to algorithmic thinking and its application to a familiar routine. Students that adhere strictly to the scenario of ‘explaining the counting process to an alien’ will soon confront issues of abstraction and assumed knowledge. They may gain an appreciation of coding and the digitisation process. |

**Activity 64:** counting in binary

**Teacher note:** physical activities invigorate students. Role-play, modelling and metaphor can draw students out of abstracted and theoretical study from behind their screens into a more concrete understanding of the concepts and material reality behind a study of computing.

****The teacher chooses 4 students to demonstrate counting in binary.

Students stand side by side facing the class.

The first student is the ‘least significant bit’ and represents the 1’s column (20s)

The next student represents the 2’s (21s)

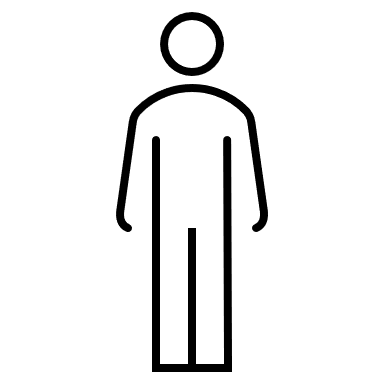
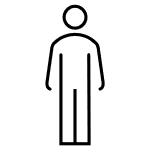
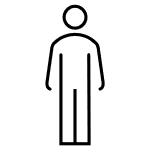
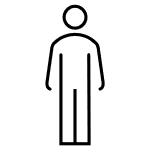
The third student represents the 4’s column (22s)

The final (fourth) student represents the 8’s column (23s)

Table 14 – counting in binary

|  |  |  |  |
| --- | --- | --- | --- |
| 23s | 22s | 21s | 20s |
| 8s | 4s | 2s | 1s |

Figure 5 – student positioning



To represent counting up using binary numbers, students are instructed that when they are standing up straight, they are ‘switched on’ and represent the value of their column.

All standing represents 15, for example (8 + 4 + 2 +1 = 15)

Accounting for 0, this gives us 16 possible values for 4 binary bits.

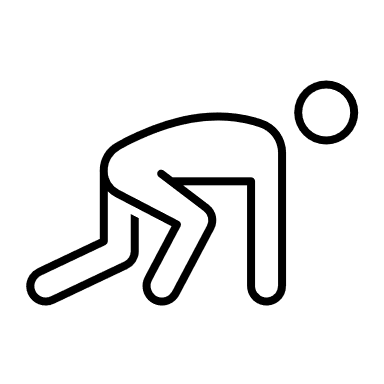
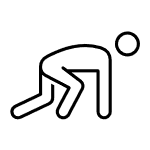
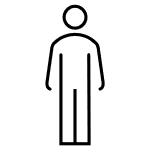
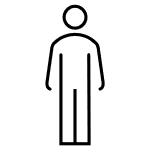
When a student crouches (or bobs down) they are **off** and represent **zero**.

For example, 5 would be represented as a one in the 4’s column + a one in the 1’s

Table 15 – counting in binary

|  |  |  |  |
| --- | --- | --- | --- |
| 23s | 22s | 21s | 20s |
| 8s | 4s | 2s | 1s |

Figure 6 – student positioning



**Activity 65:** How do I count to 16 using binary?

|  |
| --- |
| **Sample answer:**  Students should request another student join the 4, for example:  Add another bit that represents 24s (16) |

Move to 8 bits, and you have 256 possible values. This takes up a lot more space to represent, as 4 digits in decimal give us 10,000 possible values. (9999 along with zero)

Computers store bits in blocks of 8. Each block of 8 bits is called a byte.

4 bits (half a byte) is called a nibble, seriously!

Binary uses the same ‘wrap around’ algorithm with each digit being worth 2 times more than the last.

Revision of the binary counting system can be [found here](https://education.nsw.gov.au/teaching-and-learning/curriculum/tas/tas-curriculum-resources-7-12/tas-7-10-curriculum-resources/crack-the-code).

Table 16 – revision of the binary counting system

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 27s | 26s | 25s | 24s | 23s | 22s | 21s | 20s | In Base 10 |
| 128s | 64s | 32s | 16s | 8s | 4s | 2s | 1s | Calculations |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 2 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | (2+1) = 3 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 4 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | (32+8+2) = 42 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | (128+64+32+16+8+4+2+1) = 255 |

Here are some larger numbers.

**Activity 66:** counting

Each byte represents a number between 0 and 255. There are 256 numbers in total but computers consider zero to be the first counting number so the highest number is 255.

What’s the fastest way to do this calculation without adding up the columns?

|  |
| --- |
| **Sample answer:**  128 × 2 = 256  256-1= 255 |

**Activity 67:** counting binary using hands

****Using their thumb as the ‘least significant bit’ students can use the same approach to count in [binary using their hands (2:39)](https://youtu.be/Bke95oWWZII). (Beware of binary 4).

What number can they represent up to using the 10 digits on their hands?

|  |
| --- |
| **Sample answer:**  210 -1 = 512  512-1 = 511. |

**Activity 68:** binary digits

****While students are standing in their 8 bits (one byte) arrangement the teacher explains that they should all now know how decimal numbers are represented in binary: on and off (high and low voltage).

**Teacher note:** the number line represented by these 8 standing students extends, theoretically infinitely in both directions.

Computers memory is not infinite so decisions around standards of representing numbers need to be made.

What other things do computers represent using binary digits?

|  |
| --- |
| **Sample answers:**  Negative numbers  Fractions  Text  Colours |

**Activity 69:** negative numbers

[Think Pair Share](https://app.education.nsw.gov.au/digital-learning-selector/LearningActivity/Card/645#.Y_Kr3SagGrY.link). Students are asked to consider how they would represent negative numbers using only binary numbers.

Describe your solution in the space below:

|  |
| --- |
| **Sample answer:**  Creative responses may be discussed here. Students are likely to land upon the simplest method: Use the leftmost digit of the number as a special value to represent the sign of the number: 0 = positive, 1 = negative. For example, a value of positive 12 (decimal) would be written as 01100 in binary, but negative 12 (decimal) would be written as 11100. Notice that in this system, it is important to show the leading 0 (to indicate a positive value). |

**Teacher note:** students benefit from being given the opportunity to problem solve before learning the solution by rote.

For technical reasons, addressed [later in this resource](#_Represent_integers_using) ‘two's complement’ is more often used for representing negative numbers.

In this system, a positive 12 is still 01100, but -12 would be written as 10100.

Notice that there is nothing intrinsically correct about one system over another.

Either 11100 or 10100 can be used to represent -12, it just depends on what system of interpretation is used. That is, a human programmer chooses the meaning of the bits.

**Fractions**

Just as we can use the decimal point to represent fractional amounts (such as 14.72), binary values can also represent fractional amounts.

In base 10, the digit immediately to the right of the decimal point (7 in our example) represents 10-1 (that is, 7 x one-tenth).

In base- 2 binary, a number to the right of the decimal point is 2-1, or one-half.

The next digit to the right would be multiplied by one-fourth, and the next by one-eighth, and so forth.

Table 17 – binary numbers as fractions

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 2-1s | 2-2s | 2-3s |
| 0 | One half | One quarter | One eighth |

Most modern computers use a format called ‘IEEE floating-point’ to represent very large numbers and fractional numbers in a manner similar to scientific notation, using 2 rather than 10 as the base to represent the exponent and mantissa.

Floating Point numbers will be looked at closely in datatypes.

Again, a human programmer chooses the meaning of the bits.

**Activity 70: computers and numbers summary**

**Read the following paragraphs:**

A computer cannot represent every number.

It does not have infinite memory.

There will always be a number so large that we do not have enough bits to represent it. However, we can reserve certain patterns of 1s and 0s to represent certain values that have special use.

For example, even though we cannot represent very large numbers that take more bits than we have available, we can reserve one pattern of 1s and 0s to represent infinity.

This is similar to you writing the mathematical symbol for infinity ∞ on a piece of paper. Even though you might not be able to write enough digits on the page to represent a very, very large number, you can write a single, special symbol to represent infinity.

We can do the same thing with a special pattern of 1s and 0s.

That is exactly what is done in the IEEE floating-point standard used in most modern computers.

Notice that the computer treats every string of bits according to the program. For example, according to the human programmer's intention.

In the space, below summarise these into key points:

|  |
| --- |
|  |

#### How does the computer represent text?

**Activity 71:** [Think-Pair-Share](https://app.education.nsw.gov.au/digital-learning-selector/LearningActivity/Card/645#.Y_Kr3SagGrY.link)

Students are asked to consider how they would represent text using only binary numbers.

Describe your solution in the space below:

|  |
| --- |
| **Sample answer:**  Creative responses may be discussed here, Students are likely to land upon the simplest method:  Use a digit to represent ‘A’ another number for ‘B’  Students should be asked to consider how then would they represent lower case ‘a’, ‘b’ and so on.  Different students may propose different solutions encouraging a discussion on how one computer would then ‘talk’ to another and the need for standards. |

**Teacher note:** students benefit from being given the opportunity to problem solve before learning the solution by rote.

**Activity 72:** ASCII

****Examine [ASCII code](https://www.ascii-code.com/), the American Standard Code for Information Interchange to write their names in binary and check their answers with a [binary to ascii translator.](https://www.binaryhexconverter.com/binary-to-ascii-text-converter)

Write your name in ASCII code below:

|  |
| --- |
|  |

**Activity 73:** [Think-Pair-Share](https://app.education.nsw.gov.au/digital-learning-selector/LearningActivity/Card/645#.Y_Kr3SagGrY.link)

What are the limitations of ASCII? Discuss

Describe the limitations of ASCII in the space below:

|  |
| --- |
| **Sample answer:**  ASCII only works for the characters in the English language which has 26 values and fits into a byte. |

**Activity 74:**

Research languages other than English and the numbers of characters used in their choice of one of these languages. For example: Mandarin, Arabic, Cantonese, Vietnamese, Tagalog, Hindi, Greek, Spanish.

**Activity 75:**

Research Unicode and the possible number of characters it can represent.

**Hexadecimal**

There’s another base system that’s also used in programming: hexadecimal. Although computers don’t run on hexadecimal, programmers use it to represent binary addresses in a human-readable format when writing code.

This is because 2 digits of hexadecimal can represent a whole byte, 8 digits in binary.

Hexadecimal uses 0-9 like decimal, and also the letters A through F to represent the additional 6 digits. A =10, B=11, C=12, D=13, E=14, F=15

Hexadecimal uses base 16. It can be demonstrated using the same ‘wrap around’ algorithm with each digit being worth 16 times more than the last.

See the example below:

Table 18 – hexadecimal base system

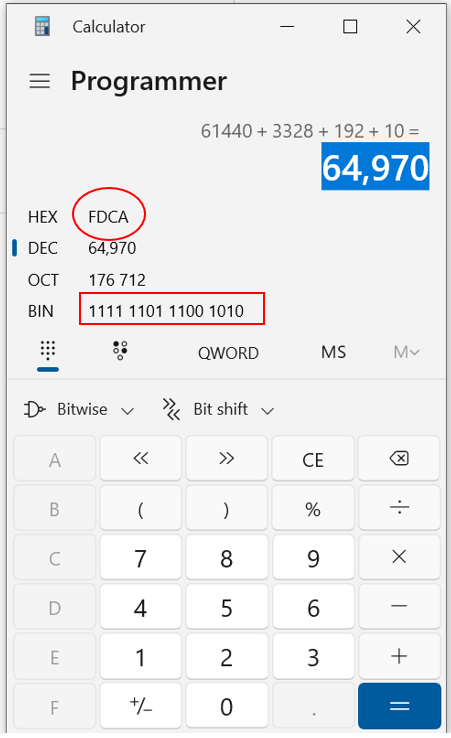
|  |  |  |  |
| --- | --- | --- | --- |
| 163s | 162s | 161s | 160s |
| 16×16×16 | 16 ×16 | 16 | 1s |
| F | D | C | A |
| 15×16×16×16 | 13×16×16 | 12×16 | 10 ×1 |

(15×16×16×16) + (13×16×16) + (12×16) + (10 ×1) =

61,440 + 3,328 + 192 + 10 = 64,970.

Using a Calculator in Programmer view, check your calculations

Figure 7 – calculator functions



Notice how 16 binary digits have been reduced to 5 decimal digits and only 4 Hexadecimal digits! on the screen that each Hexadecimal digit (F D C A) corresponds to their binary equivalent represented in 4 bits (a nibble).

F = 1111, D = 1101, C = 1100, A = 1010

This makes converting between Hexadecimal and Binary easy! Students could arrange themselves into 4 bit (half bytes or nibbles) to represent the hexadecimal digits.

Another example with a hexadecimal number that combines both digits and letters will help reinforce that the letters are an extension on the range of values that can be used in each place.

For example: 6F3B = 28475

**Activity 76:**

****Take turns providing a 4-digit Hexadecimal number on the whiteboard for their colleague to solve by hand while the class checks the answers using the calculator.

**Activity 77:** number convertor

****Design and create a number converter using either spreadsheet software or a programming language they are familiar with to convert:

* Binary to decimal
* Decimal to binary
* Decimal to Hexadecimal
* Hexadecimal to Decimal
* Binary to Hexadecimal
* Hexadecimal to Binary
* Extension challenge is to create a number system convertor that will convert between any base number system that the user required. The user should be permitted to choose any base and convert to any other base. For example, base 3 to base 7

**Teacher note:** Activity 75 could assist here.

**Teacher note:** in the topic Programming for the web, students will use CSS and HTML

These activities introduce colour on the screen and representations with hexadecimal numbers.

A monitor or TV screen generates 3 colours of light (red, green, and blue) and the different colours we see are due to different combinations and intensities of these 3 primary colours. RGB (Red, Green, Blue)

Each of these is represented using a byte which has 256 values.

These values (255, 255, 255) can also be represented in Hex and seen in webpages in this format: #FFFFFF ( FF =255, FF = 255, FF = 255)

**Activity 78:**

* Research [data representation for images and colours.](https://www.csfieldguide.org.nz/en/chapters/data-representation/images-and-colours/)
* Use [HTML color codes](https://htmlcolorcodes.com/) to add colour to HTML pages and confirm mathematically that the #hex representation is the same number as the RGB
* Experiment with colouring webpages using these values.

**Activity 79:** representing Numbers and Letters with Binary

****As a class watch [Representing Numbers and Letters with Binary (10:45)](https://youtu.be/1GSjbWt0c9M) and answer the following questions:

1. How is adding in binary like adding in decimal?

|  |
| --- |
| **Sample answer:**  They use the same algorithm. For example, carry to the next column. |

1. How many colours can be represented by 8 bit graphics?

|  |
| --- |
| **Sample answers:**  256 |

1. How many colours can be represented by 24 bit graphics?

|  |
| --- |
| **Sample answer:**  16,777,216 |

1. What is the largest number you can represent using 32 bits and 64 bits?

|  |
| --- |
| **Sample answer:**  32 bits: ‘Just under 4.3 billion’. 64 bits: 9.2 quintillion |

1. What is the standard used to represent floating point numbers?

|  |
| --- |
| **Sample answer:**  IEEE 754 standard |

1. Why was Unicode developed?

|  |
| --- |
| **Sample answer:**  To do away with different international schemes and replace them with one universal encoding scheme so computers worldwide could ‘talk to and understand each other’. |

### Represent integers using two’s complement

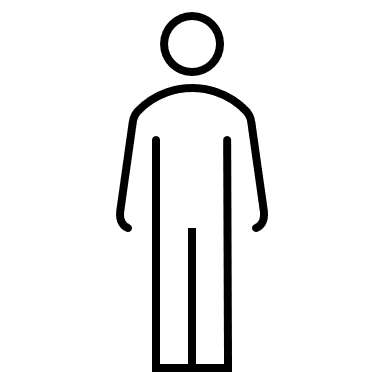
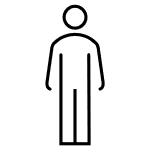
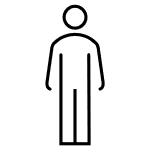
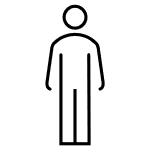
**Activity 80:** students arrange themselves as 4 bits and count off by bobbing up and down up to 15.

Students are asked how they would represent negative integers.

Table 19 – counting in binary

|  |  |  |  |
| --- | --- | --- | --- |
| 23s | 22s | 21s | 20s |
| 8s | 4s | 2s | 1s |

Figure 8 – student positioning



Students may offer the solution that one of the bits represents a minus sign.

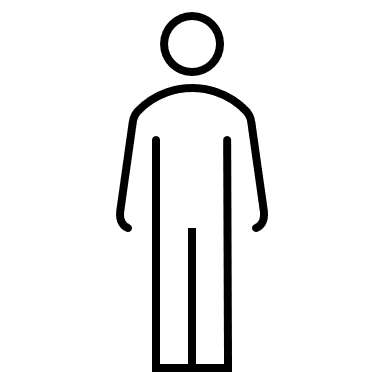
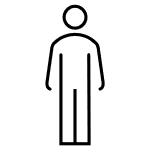
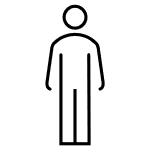
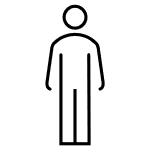
Computers only work in zero’s and ones so the discussion may turn to the need for the one of the bits to represent a sign bit.

Given computers add very quickly a better way is to represent the negative number using the twos complement method.

Table 20 – representing negative integers

|  |  |  |  |
| --- | --- | --- | --- |
| -23s | 22s | 21s | 20s |
| -8s | 4s | 2s | 1s |

Figure 9 – student positioning



(1 × (-8) + 1 × 4 +1 × 2 +1 × 1) = -1

**Activity 81:** students are given the following table with the first rows filled and complete the pattern.

Table 21 – two's complement and decimal equivalent

|  |  |
| --- | --- |
| Two’s complement | Decimal equivalent |
| **1000** | **-8** |
| **1001** | **-7** |
| **1010** | **-6** |
| 1011 | -5 |
| 1100 | -4 |
| 1101 | -3 |
| 1110 | -2 |
| 1111 | -1 |
| 0000 | 0 |
| 0001 | 1 |
| 0010 | 2 |
| 0011 | 3 |
| 0100 | 4 |
| 0101 | 5 |
| 0110 | 6 |
| 0111 | 7 |

**Activity 82:** four more students can join the 4 bit (nibble) to make a 8-bits (byte).

Using 8 bit two’s complement what is the range of numbers that can be represented?

|  |
| --- |
| **Sample answer:**  −16 to 16 |

Two's complement is the way computers represent integers.

**Activity 83:** follow the steps to find the two's complement negative notation of an integer:

For example, working with 8 bits to represent −28 expressed in two's complement notation.

* Step 1: Write out the number in binary form: 28 in binary form is 00011100
* Step 2: Invert the digits so that all 0’s becomes 1’s, and 1’s become 0’s.
  + 11100011
* Step 3: \*Add 1.
  + 11100100

That is how one to write −28 in 8 bit binary.

\*The algorithm to add in binary is the same as in decimal: each column is added commencing with the right hand side of the result is 10 or 11 then carry the I to the next column.

01+ 01+ 10, 10 + 01 = 11, 10 + 10 = 100.

Set challenges on the whiteboard for each other in using two’s complement and in binary arithmetic.

**Homework**: watch [Why We Use Two's Complement (16:15)](https://youtu.be/lKTsv6iVxV4).

### Investigate standard data types

A key concept that students need to know is that computers do not have infinite memory even though we have number systems with infinitely big and small numbers. How then, do we represent these numbers?

Other questions which could be added to the whiteboard:

**Question 1**: What is this?

0299857427

**Answer**: a series of 10 digits.

**Question 1a**: What could it be?

* An identification number
* A medicare number
* A password
* A telephone number

**Question 2**: What is this?

1000001

**Answer**: A series of 7 digits.

**Question 2a**: What could it be?

* One million and one (base 10)
* A binary number (equal to 65 decimal)

#### Visualising data types

Memory storage, like variables, can be represented using paper cups, letter boxes, lockers or cylinders.

Imagine each cylinder has a lid with an opening that accepts only a certain type of data.

Figure 10 – representation of memory storage

a

false

423

Figure 11 – representations of data types

Representations of data types

**Activity 84:** design cylinders for other data types including:

* real
* single
* precision
* floating point
* date and time

and provide examples of each.

Data types can vary based on size, length and use depending on the coding language

For example, in an Arduino sketch an int (integer data type) stores a 16-bit (2-byte) value.

This a range of: −32,768 to 32,767

Long variables are extended size variables for number storage, and store 32 bits (4 bytes), from −2,147,483,648 to 2,147,483,647.

Floating-point numbers can be:

* as large as 3.4028235E+38
* as low as −3.4028235E+38.

They are stored as 32 bits (4 bytes) of information.

(minimum value of −2^15 and a maximum value of (2^15) - 1).

In an Arduino sketch the parameter 78 produces the following output depending upon the conversion used:

Serial.print(78, BYTE) // gives "N"

Serial.print(78, BIN) // gives "1001110"

Serial.print(78, OCT) // gives "116"

Serial.print(78, DEC) // gives "78"

Serial.print(78, HEX) // gives "4E"

**Activity 85:**

For each of the following standard data types, research its use in the language of your choice [compared to its use in Python](https://www.w3schools.com/python/python_datatypes.asp).

Table 22 – data types and their use in Python and other languages

|  |  |  |
| --- | --- | --- |
| Data Type | Python | Other Language |
| char and string |  |  |
| Boolean |  |  |
| real |  |  |
| single precision floating point |  |  |
| date and time | not a data type of its own.  a module named datetime can be imported to work with dates as date objects. |  |

PYnative: Python programming (2022) [Python Data Types](https://pynative.com/python-data-types/#h-str-data-type)

**Activity 86**:

[Explore and investigate Python basics](https://pynative.com/python-exercises-with-solutions/) by completing the [’Practice Python Exercises and Challenges with Solutions](https://pynative.com/python-exercises-with-solutions/)’ in the online code editor.

**Activity 87:**

Identify where you have used the standard data types in a program or code snippet you have written.

Table 23 – data types and code samples

|  |  |
| --- | --- |
| Data Type | Code sample |
| char and string |  |
| Boolean |  |
| real |  |
| single precision floating point |  |
| date and time |  |

#### Real

Integers are a type of real number.

**Activity 88**: design a diagram that provides a simple explanation of real [numbers](https://www.mathsisfun.com/numbers/real-numbers.html%2026).

|  |
| --- |
| **Sample answer:** real numbers  A diagram that provides a simple explanation of real numbers. |

Decimals are also real numbers but contain a decimal point, (for example you can’t write them without a decimal point).

**Activity 89:**

Describe how computers use data types to represent numbers.

|  |
| --- |
| **Sample answer:**  Computers use binary numbers to represent numbers.  In many programming languages there are different types for integers and decimals.  If you know a variable is only going to contain an integer then you would declare the variable using an integer type:int i = 10;  But if you know that the variable needed to hold decimal numbers then you would declare it with a type that can store decimals: float d = 3.5; (the float type in C/C++ stores decimals).  Integers are whole numbers that include negative whole numbers, zero, and positive whole numbers.  Python uses a variable number of bits to represent integers.  The largest integer number that Python can represent depends on the available memory of the computer.  In Python, all integers are instances of the class int.  Python integers support all standard operations including addition, subtraction, multiplication, division, and exponent. |

#### Date and time

**Teacher note**: Excel has 4 main datatypes:

1. Logic
2. Number including dates
3. Text (called Strings in programming)
4. Error

All of these are fundamentally represented as zeros and ones.

Excel can be used for many engaging activities to discover datatypes including finding day zero.

**Activity 90:**

****Open an excel spreadsheet.

Choose a cell and type into the function =TODAY()

This will give you today’s date and time.

By subtracting your birthdate from TODAY you should work out how many days old you are. With a simple calculation (× 24) you can work out how many hours old you are.

You may wish to put in an estimated time of departure from school to calculate how long they have left of school.

While most of the world uses the Gregorian calendar, Microsoft uses a calendar that starts at a different date.

Can you work out the date that Microsoft’s calendar begins?

**Hint:** right click on the cell where you have added TODAY and change the data type to a number. Subtract this value from the TODAY cell and convert the result into a data datatype.

|  |
| --- |
| **Sample answer:**  0/01/1900 |

|  |
| --- |
| #Python simple data types  import datetime  #String  name = "Craig"  lastname = input("Enter your last name " + name + ": ")  fullname = name + " " + lastname  print(fullname)  #Integer  # notice the need for casting between types with constructor functions str and int  age = 0  age = int(input("Now enter your age " + fullname + ": "))  #Date / Time with datetime constructor  x = datetime.datetime.now()  year = x.year  print(fullname + " you are " + str(age) + " in " + str(year))  #Real numbers with Floating Point  # not need to declare the type but based on the initialisation  score = 0.0  score = 23/30\*100  print("Your score is: " + str(score))  #Boolean  if (score < 50):  passed = False  else:  passed = True    if passed:  print("Yay, you passed")    #Date / Time  print(datetime.datetime.now()) |

### Create data dictionaries as a tool to describe data and data types, structure data and record relationships

'Data dictionary:

A data dictionary provides a comprehensive description of each variable stored or referred to in a system. This commonly includes variable name, data type, format, size in bytes, number of characters to display the item including number of decimal places (if applicable), the purpose of each variable and a relevant example. Any validation rules applicable to the data item can also be included. Details of records or arrays of records can be included in data dictionaries.’

Source: [NESA Course Specifications](https://library.curriculum.nsw.edu.au/341419dc-8ec2-0289-7225-6db7f2d751ef/94e1eb0a-0df7-4dbe-9b72-5d5e0d17143a/software-engineering-11-12-higher-school-certificate-course-specifications.PDF)

A data dictionary contains metadata about data elements.

(For example, data about data or the rules of the data).

The metadata included in a data dictionary can assist in defining the characteristics of the data elements and the rules for how they are used and what they are used for.

**Activity 91:** consider [20 Famous Software Disasters and their causes](https://www.devtopics.com/20-famous-software-disasters/).

Which of these could potentially have been avoided by adhering to the use of a detailed data dictionary? Answer in the space below:

|  |
| --- |
|  |

**Activity 92:** choose from one of the programming exercises attempted (triangle recognition, simple calculator, number system converter, tic-tac-toe, sudoku) and create a data dictionary that describes the data structures used: variables, lists single and 2D, objects etc.

Table 24 – data dictionary

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Variable** | **Data type** | **Format for display** | **Size in bytes** | **Size for display** | **Description** | **Example** | **Validation** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

### Use data structures of arrays, records, trees and sequential files

**Activity 93:** describe the use of arrays as a data structure.

|  |
| --- |
| **Sample answer:**  An array is a fundamental data structure available in most programming languages.  Arrays store information in adjoining blocks of memory and so are contiguous data structures. An array is a collection of values of the same type saved under the same name.  An analogy would be that they are like parking spots for vehicles.  Python does not have a built-in array type, but you can use lists for all of the same tasks. |

**Activity 94:** complete and run the code for the activities in [Python Lists](https://pynative.com/python-lists/).

**Activity 95:** describe the use of records as a data structure.

|  |
| --- |
| **Sample answer:**  Compared to arrays, record data structures provide a fixed number of fields.  Each field can have a name and may also have a different type.  An analogy would be that the car park has different vehicles: cars, scooters, bikes (data types) parked.  Python uses tuples. |

**Activity 96:** complete the activities in [Tuples in Python](https://pynative.com/python-tuples/) and [Dictionaries in Python](https://pynative.com/python-dictionaries/).

**Activity 97:** describe the use of trees as a data structure.

|  |
| --- |
| **Sample answer:**  Trees are a relation-based data structure, which specialise in representing hierarchical structures.  They are like family trees you would find in genealogy.  They provide very fast ways of finding data. |

**Activity 98:** [trace the tree diagram](https://www.educative.io/blog/8-python-data-structures#tree) and run the binary tree code to explore how this works.

**Teacher note**: students to complete a tree on the whiteboard or on paper with a given set of numbers first.

**Activity 98:** describe the use of sequential files.

**Teacher note**: students have completed a reading/writing grades to file activity.

|  |
| --- |
| **Sample answer:**  A sequential file is one that contains and stores data in chronological order. Teachers may acquire a cassette tape and demonstrate how a favourite song was selected to play. For example, sequential files must be read from the beginning, up to the location of the desired data.  A text file is a sequential file. |

**Activity 99:** read through and apply the built-in functions discussed in: [Reading and Writing to test files in Python](https://www.geeksforgeeks.org/reading-writing-text-files-python/).

## Developing solutions with code

**Teacher note**: the programming activities in this teacher resource have provided experiences with and understanding of the syllabus content below The assessment task for this unit is a culmination of these activities – a solution with code project. This project determines how well students have applied skills in computational thinking and programming to develop a software solution. Teachers may consider extending upon these projects into other focus areas or as foundations for students’ major projects in the HSC course.

### Apply skills in computational thinking and programming to develop a software solution.

**Activity 100:**

The following is a checklist of knowledge and skills you should have demonstrated during this course. Provide an example of where you have used each of these in a solution you have coded.

#### Converting an algorithm into code

|  |
| --- |
|  |

#### Using control structures

|  |
| --- |
|  |

#### Using data structures

|  |
| --- |
|  |

#### Using standard modules

|  |
| --- |
|  |

#### Creating relevant subprograms that incorporate parameter passing

|  |
| --- |
|  |

### Implement data structures that support data storage

The choice of appropriate control structure (sequence, selection and iteration) is an essential part of developing an algorithm. It cannot be done properly without considering the information (or data) that the algorithm manipulates. Data that is organised to logically reflect the relationship between their elements is called ‘structured’. The data elements and their interrelationships together form a data structure.

If a single variable can be represented as a container with a different lid to accept different data types, an array can be represented as a series of the same types of containers that are accessed via their index (address). These containers have data written into or data read out of them using ‘for’ or ‘while’ iterations that loop through each container.

An example familiar to students would be a deck of cards with 52 integers.

Other examples can be found [here.](https://minecraft.makecode.com/courses/csintro/arrays/unplugged-1)

Two-dimensional arrays are best exemplified for students by using a spreadsheet or chessboard. Loops inside other loops are called nested. Nested for loops can be used to inspect and process each cell or square.

#### Single and multidimensional arrays

**Activity 101:** Predict, Run, Investigate, Modify and Make using these code snippets.

|  |
| --- |
| #Using lists as arrays  #######  #single dimension arrays  #######  #empty list  print("This is an array or list of fibonacci integers")  fib= []  #add a new element with the append function  fib.append(0)  fib.append(1)  x = 2  #populate array with fibonacci numbers  while (x <=20):  fib.append(fib[x-1] + fib[x - 2])  x +=1  print(fib)  print() |

|  |
| --- |
| #########  #Two dimension arrays are created using a list of lists  #########  twoD = [[3,2,1],[6,5,4],[9,8,7]]  print("This is a list of lists with the inside sets of brackets each a set of lists")  print(twoD)  #Each individual number can be accessed by the index of the outside list then the index of the inside list  for x in range(3):  for y in range(3):  print (twoD[x][y])  print() |

#### Lists

|  |
| --- |
| ########  #Lists that contain multiple data types  ########  mylist = ["Craig", 19, 34.5]  print(mylist[0])  print(mylist[1])  print(mylist[2]) |

#### Trees

**Activity 102:** ****watch: [Introduction to Trees (Binary Tree) in Python – A Simplified Tutorial (11:26)](https://youtu.be/fUkrQD9nw0Y).

Students follow along by drawing the Tree presented.

Students refer to Predict, Run, Investigate, Modify and Make using the following code:

|  |
| --- |
| class Node:    def \_\_init\_\_(self, data):    self.left = None  self.right = None  self.data = data    def insert(self, data):  # Compare the new value with the parent node  if self.data:  if data < self.data:  if self.left is None:  self.left = Node(data)  else:  self.left.insert(data)  elif data > self.data:  if self.right is None:  self.right = Node(data)  else:  self.right.insert(data)  else:  self.data = data    # Print the tree  def PrintTree(self):  if self.left:  self.left.PrintTree()  print( self.data),  if self.right:  self.right.PrintTree()    # Use the insert method to add nodes  root = Node(12)  root.insert(6)  root.insert(14)  root.insert(3)    root.PrintTree() |

#### Stacks

**Activity 103:**

****Watch: [Python Stacks – Python Tutorial for Absolute Beginners (4:50)](https://youtu.be/NKmasqr_Xkw).

Students refer to Predict, Run, Investigate, Modify and Make using the following code:

|  |
| --- |
| # Python program to  # demonstrate stack implementation  # using list  stack **=** []  # append() function to push  # element in the stack  stack.append('a')  stack.append('b')  stack.append('c')  **print**('Initial stack')  **print**(stack)  # pop() function to pop  # element from stack in  # LIFO order  print('\nElements popped from stack:')  print(stack.pop())  print(stack.pop())  **print**(stack.pop())  print('\nStack after elements are popped:')  print(stack)  # uncommenting print(stack.pop())  # will cause an IndexError  # as the stack is now empty |

#### Hash tables

**Activity 104:** ****as a class watch [What is a HashTable Data Structure – Introduction to Hash Tables, Part 0 (7:36)](https://youtu.be/MfhjkfocRR0).

Homework: watch [The Hash Table Data Structure: A Historical and Technical Overview](https://youtu.be/jt8mjox6vaU) (duration 17:30)

Students refer to Predict, Run, Investigate, Modify and Make using the following code:

|  |
| --- |
| ###################  # This sample Hash Table code is from  # https://stephenagrice.medium.com/how-to-implement-a-hash-table-in-python-1eb6c55019fd  ###################  # Capacity for internal array  INITIAL\_CAPACITY = 50  # Node data structure - essentially a LinkedList node  class Node:  def \_\_init\_\_(self, key, value):  self.key = key  self.value = value  self.next = None  def \_\_str\_\_(self):  return "<Node: (%s, %s), next: %s>" % (self.key, self.value, self.next != None)  def \_\_repr\_\_(self):  return str(self)  # Hash table with separate chaining  class HashTable:  # Initialize hash table  def \_\_init\_\_(self):  self.capacity = INITIAL\_CAPACITY  self.size = 0  self.buckets = [None]\*self.capacity  # Generate a hash for a given key  # Input: key - string  # Output: Index from 0 to self.capacity  def hash(self, key):  hashsum = 0  # For each character in the key  for idx, c in enumerate(key):  # Add (index + length of key) ^ (current char code)  hashsum += (idx + len(key)) \*\* ord(c)  # Perform modulus to keep hashsum in range [0, self.capacity - 1]  hashsum = hashsum % self.capacity  print(hashsum)  return hashsum  # Insert a key,value pair to the hashtable  # Input: key - string  # value - anything  # Output: void  def insert(self, key, value):  # 1. Increment size  self.size += 1  # 2. Compute index of key  index = self.hash(key)  # Go to the node corresponding to the hash  node = self.buckets[index]  # 3. If bucket is empty:  if node is None:  # Create node, add it, return  self.buckets[index] = Node(key, value)  return  # 4. Iterate to the end of the linked list at provided index  prev = node  while node is not None:  prev = node  node = node.next  # Add a new node at the end of the list with provided key/value  prev.next = Node(key, value)  # Find a data value based on key  # Input: key - string  # Output: value stored under "key" or None if not found  def find(self, key):  # 1. Compute hash  index = self.hash(key)  # 2. Go to first node in list at bucket  node = self.buckets[index]  # 3. Traverse the linked list at this node  while node is not None and node.key != key:  node = node.next  # 4. Now, node is the requested key/value pair or None  if node is None:  # Not found  return None  else:  # Found - return the data value  return node.value  # Remove node stored at key  # Input: key - string  # Output: removed data value or None if not found  def remove(self, key):  # 1. Compute hash  index = self.hash(key)  node = self.buckets[index]  prev = None  # 2. Iterate to the requested node  while node is not None and node.key != key:  prev = node  node = node.next  # Now, node is either the requested node or none  if node is None:  # 3. Key not found  return None  else:  # 4. The key was found.  self.size -= 1  print(node.value)  #result = node.value  # Delete this element in linked list  if prev is None:  self.buckets[index] = node.next # May be None, or the next match  else:  prev.next = prev.next.next # LinkedList delete by skipping over  # Return the deleted result  return result |

### Compare the execution of the Waterfall and Agile project management models as applied to software development

**Activity 105**: class debate

****Students are divided into 2 teams: Waterfall and Agile.

Each team has smaller teams to complete the following:

* Researchers
* Description
* History
* Where its used/clients
* Advantages
* Disadvantages
* Case study of success and failure
* Speakers
* First
* Second
* Third

Complete the following table after the debate.

Teacher may introduce other approaches including Wagile which combines Waterfall and Agile

Justify which approach is best suited to their Educational Software Project

Table 25 – Agile versus Waterfall comparison chart

|  |  |  |
| --- | --- | --- |
| Comparison | Waterfall | Agile |
| Beginnings | 1950 | 2001 |
| Background | Infrastructure and engineering | Software development |
| Client interaction | **Minimal** | **Encouraged** |
| Papers | *Managing the Development of Large Software System* by Winston Royce | The Agile Manifesto |
| Preferred by | Banks, governments, insurance companies, large teams | Startups, small teams, SaaS products, small companies |
| Highest priority | Deliver an end product that matches initial requirements | Continuously deliver working software to the client |
| Advantages | Enables organisations to do extensive, upfront estimation and planning | Enables teams to rapidly respond to changing requirements |
| Disadvantages | Lack of customer involvement and an overwhelming amount of upfront documentation | Software delivery timelines can be difficult to estimate if requirements frequently change |

### Test and evaluate solutions, considering key aspects including functionality, performance, readability of code and quality of documentation

**Activity 106**:

Discuss the key aspects of testing and evaluating software solutions to agree upon what each of these would look like in a student solution.

Each student presents their project to the class and are peer assessed on the following criteria.

Table 26 – project peer review

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Criteria | 1 | 2 | 3 | 4 | 5 |
| FUNCTIONALITY |  |  |  |  |  |
| PERFORMANCE |  |  |  |  |  |
| READABILITY OF CODE |  |  |  |  |  |
| QUALITY OF DOCUMENTATION |  |  |  |  |  |

Students redesign this table into a data structure that will store each student’s evaluation of other students’ solutions.

This will include student name, project name and total and average functions.

This could be done in a shared Google sheet OR as a coding problem and software solution to extend students.

### Use debugging tools

**Activity 107**:

Place your code into an online debugging tool, like ‘[Online Python Debugger’](https://www.onlinegdb.com/online_python_debugger), and press **debug** on the menu.

What is a debugging tool?

|  |
| --- |
| **Sample answer:**  It is a utility that runs a target program in an environment where you can control the execution of a program and see the state of the code when the program is paused. |

#### Breakpoints

What are breakpoints?

|  |
| --- |
| **Sample answer:**  These are the places the programmer sets to tell the debugger where to pause the program. When you start the debug mode it will pause execution when the program reaches the line where breakpoint is set. |

#### Single line stepping

What is single line stepping?

|  |
| --- |
| **Sample answer:**  Single-stepping is a powerful feature of a debugger as it allows the programmer to inspect a single instruction at a time before returning control to the debugger. |

#### Watches

What are watches in debugging?

|  |
| --- |
| **Sample answer:**  Watches are windows that allow the programmer to see the value held by all variables at a particular time. The values are updated with each step through the program and become visible every time the program is paused. |

#### Interfaces between functions

**Activity 108:**

Use the [Python Debugger](https://www.geeksforgeeks.org/python-debugger-python-pdb/) to test their code.

#### Debugging output statements

How do we use debugging output statements?

|  |
| --- |
| **Sample answer:**  This debugging technique is based on inserting statements (in suitable positions of the source code) that print the content of variables that could contain wrong values causing an error. |

**Activity 109:**

As a class watch: [Debugging Practice: Using Print Statements (7:58)](https://youtu.be/r0JvqH6OWKQ).

#### Debugging software available in an integrated development environment (IDE)

**Activity 110:**

As a class watch [Debugging Python | Python tricks (15:10)](https://youtu.be/I6yEW9DCPMA).

List the tools in pdb and pdup.

|  |
| --- |
| **Sample answer:**  Windows for variables, stacks, breakpoints, step by step tracing through code, post mortem mode. |

### Determine sets of suitable test data

#### Boundary values

The control structures, ‘selection’ (branching) and ‘repetition’ (loops) use the results of conditions to determine their actions.

These conditions if incorrect can cause major errors.

To avoid this, all conditions need close inspection during testing. Every condition should be checked for test data above, below and equal to any values upon which decisions are based.

For example, the condition Counter < 3 should be tested using a value greater than 3, less than 3 and equal to 3.

**Activity 111:**

Students identify all conditions in one of their programs:

* triangle recognition
* simple calculator
* number system convertor
* tic-tac-toe.

****Students determine suitable test data to check the boundaries of these conditions.

**Activity 112:**

Write an algorithm to determine the ticket type for a customer using the NSW trains ‘[Ticket eligibility and concessions](https://transportnsw.info/tickets-opal/ticket-eligibility-concessions)’ webpage.

****Students identify the conditions and determine suitable test data to check the boundaries of these conditions.

#### Path coverage

Path coverage is used to test complex code snippets that include loop statements or a combination of loops and decision statements.

More complex and complicated algorithms occur when other conditional operators or logical operators (and or or) are added that have values based on previous processes. In these algorithms each possible path of execution is tested to ensure path coverage.

More information can be found on the ‘[White Box Testing: A Complete Guide With Techniques, Examples, & Tools’](https://www.softwaretestinghelp.com/white-box-testing-techniques-with-example/) webpage.

**Activity 113:** refer to the Course Specifications to convert the pseudocode below into a flowchart and complete the following questions.

INPUT A & B

C = A + B

IF C>100

PRINT “ITS DONE”

END IF

IF A>50

PRINT “ITS NOT DONE ”

END IF

|  |
| --- |
| Flowchart of the ‘done and not done’ pseudocode |

**Activity 114:** how many test cases are needed for path coverage?

|  |
| --- |
| **Sample answer:**  There are 2 decision statements.  For each decision statement, 2 branches need to be tested.  One for true and the other for the false condition.  Two decision statements require 2 test cases to test the true side and 2 test cases to test the false side, which makes a total of 4 test cases. |

**Activity 115:** determine 4 test cases?

|  |
| --- |
| **Sample answers:**   * TestCase\_01: A=50, B=60 * TestCase\_02: A=55, B=40 * TestCase\_03: A=40, B=65 * TestCase\_04: A=30, B=30 |

**Activity 116:** studentschoose 4 coloured highlighters and trace the path each test case takes onto their flowchart algorithm.

This could be done on a whiteboard or using graphic software.

#### Faulty and abnormal data

**Activity 117:** complete the following table of types of test data.

Table 27 – types of test data

|  |  |  |
| --- | --- | --- |
| Test criteria | Description | Example: (only integers between 1 and 10 allowed as input) |
| Normal data | Typical and expected data that should be accepted by the program | 5 |
| Extreme data | Data at the upper or lower limits of expectations that should be accepted by the program | 1, 10 |
| Boundary data | A pair of values at each end of a range:   * the data at the upper or lower limits of expectations that should be accepted * the immediate values before or beyond the limits of expectation that should be rejected. | 1,10 (to be accepted)  0,11 (to be rejected) |
| Faulty and Abnormal | Data that falls outside of what is acceptable and should be rejected by the system | Fourteen, 5.7, 13 |

**Activity 118:**

Describe types of test data appropriate for use in one of the programs they have worked upon:

* triangle recognition
* simple calculator
* number system convertor
* tic-tac-toe.

**Activity 119:** extension

Explain the importance of [testing in the Software Engineering](https://cs.ccsu.edu/~stan/classes/CS410/Notes16/08-SoftwareTesting.html) industry with reference to all types.

More information can be found on the ‘[CS 410/510 – Software Engineering: Software testing](https://cs.ccsu.edu/~stan/classes/CS410/Notes16/08-SoftwareTesting.html)’ webpage.

### Determine typical errors experienced when developing code, including syntax, logic, and runtime, and explain their likely causes

**Activity 120:** bug book

Issue each student with a bug book (either an exercise book or digital) where they document every error they experience and how they solved it.

Teachers should make it classroom practise to advise students to ‘ask 3 and then me’.

This will help students move beyond learned helplessness, encourage sharing and collaboration and provide students with time and “headspace” to solve the problems and enjoy the inherent satisfaction of doing so.

Students use the debugging tools of the IDE they are using and are encouraged to browse solutions to error messages and document these solutions.

Students document their errors in categories: Syntax, Runtime and Logic.

Learning a new language will result in Syntax errors being experienced most often.

Some of the most common examples of runtime errors in Python are:

* division by zero
* using an undefined variable or function name
* performing an operation on incompatible types
* accessing a list element, dictionary key or object attribute that does not exist
* accessing a file that does not exist.

Logic may be the hardest to find as the program may still run though the output is incorrect. The example of entering 3 sides with length ‘0’ into a triangle recognition program and receiving ‘Equilateral’ is an example of a logic error since triangles cannot be made of zero length sides.

Table 28 – bug book including syntax, runtime and logic errors

|  |  |  |
| --- | --- | --- |
|  | Bug book  My favourite errors |  |
| Syntax | **Runtime** | **Logic** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## Appendix 1 – basic programming concepts pre-test

This pre-test has been developed to gauge students' current ability to read and understand simple snippets of code.

Students should be assured that their responses provide a guide to the teacher on where class time needs to be spent.

This pre-test will also provide students with an indication of the key concepts they need to understand and master.

Students should be encouraged to desk check every algorithm.

1. The variables hobbit and elf contain 1.4 and 2.0 respectively, which represent their height. What is the output of the program?

IF hobbit > elf THEN

Display "Elf is taller"

ELSE

Display "Hobbit is taller"

END IF

1. Elf is taller
2. **Hobbit is taller**
3. Elf is taller Hobbit is taller
4. What will the computer display after it has executed the following instructions?

a = 7

b = 9

a = b

Display a

1. True
2. False
3. 7
4. **9**
5. What value is stored in y after the following code executes?

x = 10

y = 4

WHILE x >= y

y = y + 2

END WHILE

1. 4
2. 10
3. **12**
4. What will the program display after an input of –1 is entered?

Get x

IF x > 3 THEN

Display “Boo”

ELSE

Display “-hoo”

ENDIF

Display "!"

1. !
2. Boo
3. **-hoo!**
4. Boo-hoo!
5. What will the computer display after executing the following program if the user enters the number 3 for the first INPUT and 4 for the second?

Get a

Get b

b = a

a = b

Display a

Display b

1. 3 4
2. 4 3
3. **3 3**
4. 4 4
5. 3 0
6. 0 4
7. What will the following code output?

count = 0

output = ""

WHILE count < 3

count = count + 1

output = output & "C"

END WHILE

Display output

1. The program produces no output
2. C
3. CC
4. **CCC**
5. What will the program display after it’s executed if the user types in 7 for x?

y = 0

INPUT x

IF x > 3 THEN

y = 1

ELSE

y = 4

END IF

Display y

1. 0
2. **1**
3. 4
4. 5
5. 14
6. What will the following code segment output?

count = 0

x = 9

WHILE x > 7

count = count + 1

x = x – 1

END WHILE

Display count

1. 0
2. 1
3. **2**
4. 3
5. How many times is Hello displayed after the execution of the program below?

i = 0

FOR i = 0 TO i = 10 STEP 2

Display "Hello"

NEXT i

1. 10
2. 6
3. **5**
4. 4
5. Which code fragment, inserted in the position indicated between the WHILE and the DO keywords would ensure that the following loop body was only executed 3 times?

x = 0

WHILE <Code fragment inserted here>

x = x + 1

END WHILE

1. x > 3
2. x >= 3
3. **x < 3**
4. x <= 3
5. Consider the following pseudocode.

IF final THEN

display winner

ENDIF

Which data type is final most likely to be?

1. **Boolean**
2. Integer
3. Real
4. String
5. Which of the following is the closest description of the purpose of the algorithm?

x = 0

WHILE x != 3

Get v

IF v = 3 THEN

x = 3

END IF

END WHILE

1. Read in the value 3 five times
2. Read in five numbers
3. Read in and add up five numbers
4. **Read in numbers until a 3 is received**
5. We need to determine whether someone is of working age.

For this purpose, we can start working when we are 16 and stop working when we are 65.

We have the following if statement:

IF <Boolean expression to test age> THEN

<Code if the person is of working age>

ELSE

<Code if the person is not of working age>

END IF

Assuming that the person's age is held in the variable personsAge, which is the best Boolean expression for this if statement:

1. personsAge < 16 OR personsAge >= 65
2. personsAge > 16 OR personsAge < 66
3. personsAge < 16 OR > 65
4. **personsAge >= 16 AND personsAge < 65**
5. personsAge >= 16 AND <=65
6. What will the following code print out to the screen?

a = 0

b = 0

REPEAT

b = b + 1

UNTIL a = 0

Display b

1. 0
2. **1**
3. 3
4. Infinite loop
5. What is printed out as a result of this code?

answer = 0

FOR startNumber = 1 TO startNumber = 5

startNumber = startNumber + 1

answer = answer + startNumber

END FOR

Display answer

1. **12**
2. 15
3. 20
4. Other
5. The following program has been designed to print each person’s name, one after the other. Which code fragment is required to achieve this purpose?

names = [“Jane”, “Henry”, “Kirsty”, “Robert”]

person = 0

WHILE person < length of names list

Display <Code fragment inserted here>

person = person + 1

END WHILE

1. **names[ person ]**
2. names[ 1 ]
3. names
4. names[ “henry” ]
5. Jane
6. The following section of code is meant to determine whether someone is old enough to have a full driving licence.

Get age

IF age >= 17

Display “Eligible for licence”

ELSE

Display “Not eligible for licence”

END IF

What is displayed if an age of 15 is entered?

1. Eligible for licence
2. **Not eligible for licence**
3. nothing to display
4. Consider the following code segment.

names = [ “Alan”, “Bert”, “Cora”, “Dani”]

Display names [2]

What is output of the program?

1. Alan
2. Bert
3. **Cora**
4. Dani
5. Which code fragment needs to be added in order for the following program to print each person’s name one after the other. Assume that the array index starts at 1.

names = [“Jane”, “Henry”, “Kirsty”, “Robert”]

person = 1

WHILE person <= length of names

Display <Code fragment inserted here>

person = person + 1

END WHILE

1. **names[ person ]**
2. names[ 1 ]
3. names
4. names[ “henry” ]
5. Jane
6. What is displayed on the screen after this section of code is executed?

a = 5

IF a = 3 THEN

Display “True”

END IF

Display a

1. 3
2. **5**
3. True
4. False
5. Consider the segment of code below.

apples = < answer 1 >

oranges = < answer 2 >

IF apples > 5 OR oranges > 10 THEN

Display “Your fruit order requires two boxes!”

END IF

Which of the following set of values does not produce the message “Your fruit order requires two boxes!”

1. **5, 10**
2. 10, 5
3. 1, 20
4. 20, 30
5. What is the result of this section of code if 10 is entered as the mark?

Get mark

IF mark > 0 AND < 100 THEN

Display “Mark in range”

ELSE

Display “Out of range”

END IF

1. **Mark in range**
2. Mark out of range
3. Syntax error / will not run
4. True
5. What will be the output of this program?

passcode = “456”

passcode2 = 456

IF passcode = passcode2 THEN

Display “Passcode matches”

ELSE

Display “Passcode does not match”

END IF

1. Passcode matches
2. Passcode does not match
3. **Syntax error**
4. **No output will be displayed**
5. What will be the output on the screen when the following code is run with a value of 13 input:

BEGIN Main

Get number

processed = Process(number)

Display “This is a “ + processed + “ number.”

END Main

BEGIN Process(var1)

IF var1 = 42 OR var1 = 1337 THEN

message = “great”

ELSE

message = “boring”

ENDIF

RETURN message

END Process

1. great
2. boring
3. This is a great number
4. **This is a boring number**

## References

This resource contains NSW Curriculum and syllabus content. The NSW Curriculum is developed by the NSW Education Standards Authority. This content is prepared by NESA for and on behalf of the Crown in right of the State of New South Wales. The material is protected by Crown copyright.

Please refer to the NESA Copyright Disclaimer for more information <https://educationstandards.nsw.edu.au/wps/portal/nesa/mini-footer/copyright>.

NESA holds the only official and up-to-date versions of the NSW Curriculum and syllabus documents. Please visit the NSW Education Standards Authority (NESA) website [https://educationstandards.nsw.edu.au/](https://aus01.safelinks.protection.outlook.com/?url=https%3A%2F%2Feducationstandards.nsw.edu.au%2F&data=05%7C01%7CCaitlin.Pace1%40det.nsw.edu.au%7C9c2c1a9f59c94d2df30708dafa7edb23%7C05a0e69a418a47c19c259387261bf991%7C0%7C0%7C638097720042599463%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=NzFc%2Bg75dh42X2SUZ1JxGsPBapsGv7KoRkSbJ1MjiTE%3D&reserved=0) and the NSW Curriculum website [https://curriculum.nsw.edu.au/home](https://aus01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fcurriculum.nsw.edu.au%2Fhome&data=05%7C01%7CCaitlin.Pace1%40det.nsw.edu.au%7C9c2c1a9f59c94d2df30708dafa7edb23%7C05a0e69a418a47c19c259387261bf991%7C0%7C0%7C638097720042599463%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=SYVPECiogUlm2Ck2OkCJ8LGVJ3ZUXn%2Bm5%2F%2FbO4ocGOM%3D&reserved=0).

[Software Engineering 11–12 Course Specifications](https://curriculum.nsw.edu.au/syllabuses/software-engineering-11-12-2022?tab=teaching-and-learning) © NSW Education Standards Authority (NESA) for and on behalf of the Crown in the State of New South Wales, 2022.

[Software Engineering 11–12 Syllabus](https://curriculum.nsw.edu.au/syllabuses/software-engineering-11-12-2022?tab=course-overview) © NSW Education Standards Authority (NESA) for and on behalf of the Crown in the State of New South Wales, 2022.

Bader D (26 August 2020) ‘[Common Python Data Structures (Guide)](https://realpython.com/python-data-structures/#array-data-structures)’, *Real Python*, accessed 25 May 2023.

BBC (2023) [What are the types of triangle?](https://realpython.com/python-data-structures/#array-data-structures), BBC Bitesize website, accessed 16 May 2023.

BinaryHexConvertor (2009–2023) [Binary to Ascii Text Converter](https://www.binaryhexconverter.com/binary-to-ascii-text-converter), BinaryHexConvertor website, accessed 25 May 2023.

Bourne Grammar UK (2023) [3.2.7 Subroutines (Procedures and Functions)](https://bournetocode.com/projects/GCSE_Computing_Fundamentals/pages/3-2-7-subroutines.html), Bournetocode website, accessed 16 May 2023.

Code.org (2023) ‘[Anyone can learn computer science](https://studio.code.org/courses)’, *Courses*, Code.org website, accessed 25 May 2023.

Codesansar (2023) [Python Program to Check Validity of Triangle Given Three Sides](https://www.codesansar.com/python-programming-examples/check-validity-triangle-given-sides.htm), Codesansar website, accessed 16 May 2023.

Computer Science (20 March 2022) ['Python Programming 16. Sub Procedures' [video]](https://youtu.be/K05_CozFPeE), *Computer Science*, YouTube, accessed 25 May 2023.

ComputerScienced (2023) [Parson's Puzzles](https://computerscienced.co.uk/site/parsons-puzzles/), Computer Scienced website, accessed 25 May 2023.

Crash Course (16 March 2017) [‘Representing Numbers and Letters with Binary: Crash Course Computer Science #4’ [video]](https://www.youtube.com/watch?v=1GSjbWt0c9M), *CrashCourse*, YouTube, accessed 25 May 2023.

Crash Course (18 May 2017) [‘Programming Basics: Statements & Functions’ [video]](https://www.youtube.com/watch?v=l26oaHV7D40), *CrashCourse*, YouTube, accessed 25 May 2023.

CSIRO Australia (2023) [Bebras](https://www.csiro.au/en/education/Programs/Digital-Careers/Bebras), CSIRO website, accessed 25 May 2023.

Dixon & Moe (2015-2023) [HTML Color Picker](https://htmlcolorcodes.com/), HTML Color Codes website, accessed 25 May 2023.

Educba (2023) [Object oriented programming paradigm](https://www.educba.com/object-oriented-programming-paradigm/), Educba website, accessed 25 May 2023.

Excel Macro Mania (3 Dec 2021) ['Excel Sudoku Generator (Part 1/3)' [video]](https://youtu.be/zp4J0qGgN_w), *Excel Macro Mania*, YouTube, accessed 25 May 2023.

GBD Online (2016-2023) [Online Python Debugger](https://www.onlinegdb.com/online_python_debugger), GBD Online website, accessed 25 May 2023.

Hamedani M (14 November 2018) ['Python Stacks - Python Tutorial for Absolute Beginners' [video]](https://youtu.be/NKmasqr_Xkw), *Programming with Mosh*, YouTube, accessed 25 May 2023.

Harvey B and Wright M (1991) ‘[Chapter 10: Example: Tic-Tac-Toe](http://people.eecs.berkeley.edu/~bh/ssch10/ttt.html)’, in MIT Press (eds) *Simply Scheme: Introducing Computer Science,* 2nd edn, The *MIT Press*, accessed 25 May 2023.

Hod S (MIT Licensed) (2023) [Parsons Puzzles](https://parsons.problemsolving.io/) [website], accessed 16 May 2023.

Injosoft AB (2005-2023) [ASCII Table](https://www.ascii-code.com/), ASCII-CODE.com, accessed 25 May 2023.

IONOS Inc. (21 May 2021) ‘[Imperative programming: Overview of the oldest programming paradigm](https://www.ionos.com/digitalguide/websites/web-development/imperative-programming/)’, IONOS *Digital Guide*, accessed 25 May 2023.

ISO Training Institute (3 November 2017) [‘COMPILER| INTERPRETER |Difference between Interpreter and Compiler| Interpreter vs Compiler Animated’ [video]](https://www.youtube.com/watch?v=e4ax90XmUBc), *ISO Training Institute*, YouTube, accessed 25 May 2023.

Jones D (2021-2023) [Reading and Writing Files in Python](https://realpython.com/courses/reading-and-writing-files-python/), Real Python website, accessed 16 May 2023.

Khan Academy (2023) [Divide and Conquer Algorithms](https://www.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/divide-and-conquer-algorithms), Khan Academy website, accessed 29 May 2023.

Kindson The Genius (3 September 2021) [‘Introduction to Trees (Binary Tree) in Python - A Simplified Tutorial’ [video]](https://www.youtube.com/watch?v=fUkrQD9nw0Y), *Kindson The Genius*, YouTube, accessed 25 May 2023.

Mathôt S (2018–2023) [Python Tutorials](https://pythontutorials.eu/), Sebastiaan Mathôt website, accessed 16 May 2023.

Mathôt S (5 March 2018) ['Debugging Python | Python tricks' [video]](https://youtu.be/I6yEW9DCPMA), *Sebastiaan Mathôt*, YouTube, accessed 25 May 2023.

Mirza M and GitHub, Inc. (2019) [Tic tac toe from chapter 11 of Programming in Haskell – Hutton 2nd edition](https://gist.github.com/momirza/aec1ee5e86f315e85c5cd84867d2ff86), GitHub Gist website, accessed 16 May 2023.

NESA (NSW Education Standards Authority) (2023) [Familiarisation questions](https://fam.hsconline.nesa.nsw.edu.au/)*,* HSC Online website, accessed 25 May 2023.

Numberphile (22 August 2015) ['The 8 Queen Problem - Numberphile' [video]](https://youtu.be/jPcBU0Z2Hj8), *Numberphile*, YouTube, accessed 25 May 2023.

Paul Programming (21 May 2013) ['What is a HashTable Data Structure - Introduction to Hash Tables , Part 0' [video]](https://youtu.be/MfhjkfocRR0), *Paul Programming*, YouTube, accessed 25 May 2023.

pktparticle (n.d.) ‘[Computer Organization | Von Neumann architecture](https://www.geeksforgeeks.org/computer-organization-von-neumann-architecture/)’, *Geeks for Geeks*, accessed 16 May 2023.

PRIMM (2018) [PRIMM materials 2018](https://primmportal.com/), PRIMM portal website, accessed 25 May 2023.

PRIMM (2018) [What Is PRIMM?](https://primmportal.com/), PRIMM portal website, accessed 25 May 2023.

Pykes K (August 2022) ‘[Two Simple Methods To Convert A Python File To An Exe File](https://www.datacamp.com/tutorial/two-simple-methods-to-convert-a-python-file-to-an-exe-file)’, *datacamp*, accessed 25 May 2023.

Python Software Foundation (2001–2023) [CSV File Reading and Writing](https://docs.python.org/3/library/csv.html), Python website, accessed 25 May 2023.

Python Software Foundation (2001–2023) [Python](https://www.python.org/) [website], Python website, accessed 25 May 2023.

Python Software Foundation (2023) [Python for Non-Programmers](https://wiki.python.org/moin/BeginnersGuide/NonProgrammers), Python Wikipedia website accessed 25 May 2023.

Ramos LP (27 June 2022) ‘[Build a Tic-Tac-Toe Game With Python and Tkinter](https://realpython.com/tic-tac-toe-python/)’, *Real Python*, accessed 25 May 2023.

Refsnes Data (1999-2023) [Python File Open](https://www.w3schools.com/python/python_file_open.asp), W3schools website, accessed 16 May 2023.

Replit, Inc. (2023) [Teams for Education](https://replit.com/site/teams-for-education), Replit website, accessed 25 May 2023.

Ruscia T (2023) ‘[Learn how to create a Sudoku Solver using python and backtracking.](https://www.techwithtim.net/tutorials/python-programming/sudoku-solver-backtracking)’, *Tech with Tim*, accessed 25 May 2023.

Slim R (24 February 2023) [‘The Hash Table Data Structure: A Historical and Technical Overview’ [video]](https://www.youtube.com/watch?v=jt8mjox6vaU), *Rayan Slim*, YouTube, accessed 25 May 2023.

Software Testing Help (5 May 2023) ‘[White Box Testing: A Complete Guide With Techniques, Examples, & Tools](https://www.softwaretestinghelp.com/white-box-testing-techniques-with-example/)’, *Software Testing Help*, accessed 25 May 2023.

Sommerville (n.d.) [Software Testing](https://cs.ccsu.edu/~stan/classes/CS410/Notes16/08-SoftwareTesting.html), CS 410/510 - Software Engineering website, accessed 25 May 2023.

Spraul AV (17 January 2018) [‘Backtracking (Think Like a Programmer)’ [video]](https://www.youtube.com/watch?v=gBC_Fd8EE8A), *V. Anton Spraul*, YouTube, accessed 25 May 2023.

Stack Exchange Inc. (2023) [Tic Tac Toe in Python with OOP](https://codereview.stackexchange.com/questions/274308/tic-tac-toe-in-python-with-oop), Code Review Stack Exchange website, accessed 25 May 2023.

Sweigart A (2023) [#73 SUDOKU PUZZLE](https://inventwithpython.com/bigbookpython/project73.html), Invent with Python website, accessed 25 May 2023.

Sweigart A (2023) [15 OBJECT-ORIENTED PROGRAMMING AND CLASSES](https://inventwithpython.com/beyond/chapter15.html), Invent with Python website, accessed 25 May 2023.

Sweigart A (2023) [Tic-Tac-Toe OOP and Non OOP Comparison](https://autbor.com/compareoop/#diff), Autbor website, accessed 25 May 2023.

Tanimoto S (11 May 2003) [Prolog Tic-Tac-Toe](https://courses.cs.washington.edu/courses/cse341/03sp/slides/PrologEx/tictactoe.pl.txt) [website], accessed 25 May 2023.

Tanimoto S (11 May 2003) [SWISH Prolog Tic- Tac- Toe](https://swish.swi-prolog.org/p/Tic-Tac-Toe.swinb), Swish website, accessed 25 May 2023.

Tejalk A (4 November 2022) ‘[Python Debugger – Python pdb](https://www.geeksforgeeks.org/python-debugger-python-pdb/)’, *Geeks for Geeks*, accessed 25 May 2023.

Thenlin R (16 February 2021) ‘[8 data structures every Python programmer needs to know](https://www.educative.io/blog/8-python-data-structures#tree)’, *Educative*, accessed 25 May 2023.

Toady T (2007-2008) ‘[20 Famous Software Disasters](https://www.devtopics.com/20-famous-software-disasters/)’, *DevTopics*, accessed 25 May 2023.

TokyoEdTech (May 16 2018) ['Debugging Practice: Using Print Statements' [video]](https://youtu.be/r0JvqH6OWKQ), *TokyoEdTech*, YouTube, accessed 25 May 2023.

Transformers And Power Rangers Toy Reviews (23 January 2012) ['How to Count to 31 on One Hand by Using the Binary Counting System' [video]](https://youtu.be/Bke95oWWZII), *Transformers And Power Ranges Toy Reviews,* YouTube, accessed 25 May 2023.

Transport for NSW (n.d.) ‘[Ticket eligibility and concessions](https://transportnsw.info/tickets-opal/ticket-eligibility-concessions)’, *Tickets and Opal,* Transport for NSW website, accessed 16 May 2023.

Trejo CG (2023) [How and when to use any() and all() in Python](https://www.educative.io/answers/how-and-when-to-use-any-and-all-in-python), Educative.io website, accessed 29 May 2023.

Trinket (2015-2023) [Code is Your Canvas](https://trinket.io/), Trinket website, accessed 25 May 2023.

UC Computer Science Education (30 October 2010) ['Santa's Dirty Socks (Divide and Conquer Algorithms)' [video]](https://youtu.be/wVPCT1VjySA), *UC Computer Science Education*, YouTube, accessed 25 May 2023.

University of Canterbury, New Zealand (Computer Science Education Research Group) (n.d.) ‘[Data Representation: 5.5. Images and Colours](https://www.csfieldguide.org.nz/en/chapters/data-representation/images-and-colours/)’, in *Computer Science Field Guides*, CSFG website, accessed 16 May 2023.

University of Canterbury, New Zealand (Computer Science Education Research Group) (n.d.) [Algorithms](https://www.csunplugged.org/en/at-a-distance/algorithms/), CS Unplugged website, accessed 25 May 2023.

Vishalxviii (2023) ‘[Functional Programming Paradigm](https://www.geeksforgeeks.org/functional-programming-paradigm/)’, *Geeks for Geeks*, accessed 25 May 2023.

Wild C (29 June 2017) ‘[Relational and Logical Operators](https://www.cs.odu.edu/~zeil/cs250PreTest/latest/Public/operators/index.html)’, Old Dominion University website, accessed 25 May 2023.

**© State of New South Wales (Department of Education), 2023**

The copyright material published in this resource is subject to the *Copyright Act 1968* (Cth) and is owned by the NSW Department of Education or, where indicated, by a party other than the NSW Department of Education (third-party material).

Copyright material available in this resource and owned by the NSW Department of Education is licensed under a [Creative Commons Attribution 4.0 International (CC BY 4.0) licence](https://creativecommons.org/licenses/by/4.0/).

[](https://creativecommons.org/licenses/by/4.0/)

This licence allows you to share and adapt the material for any purpose, even commercially.

Attribution should be given to © State of New South Wales (Department of Education), 2023.

Material in this resource not available under a Creative Commons licence:

* the NSW Department of Education logo, other logos and trademark-protected material
* material owned by a third party that has been reproduced with permission. You will need to obtain permission from the third party to reuse its material.

**Links to third-party material and websites**

Please note that the provided (reading/viewing material/list/links/texts) are a suggestion only and implies no endorsement, by the New South Wales Department of Education, of any author, publisher, or book title. School principals and teachers are best placed to assess the suitability of resources that would complement the curriculum and reflect the needs and interests of their students.

If you use the links provided in this document to access a third-party's website, you acknowledge that the terms of use, including licence terms set out on the third-party's website apply to the use which may be made of the materials on that third-party website or where permitted by the *Copyright Act 1968* (Cth). The department accepts no responsibility for content on third-party websites