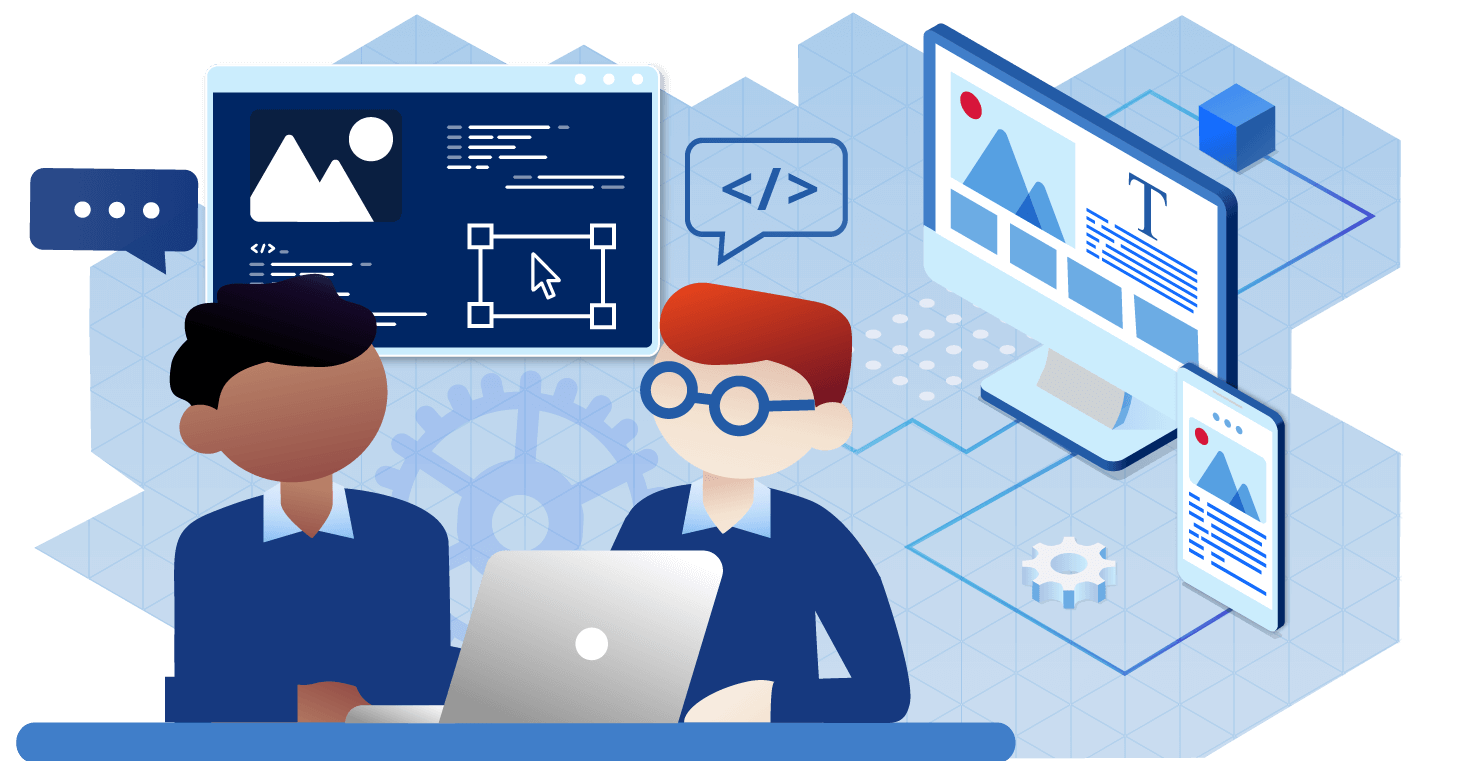
# Software Engineering Stage 6 (11–12) – sample program of learning

**programming fundamentals**



Contents

[Rationale 3](#_Toc120548202)

[Overview 4](#_Toc120548203)

[Outcomes 5](#_Toc120548204)

[Lesson sequence and details 7](#_Toc120548205)

[Week 1 7](#_Toc120548206)

[Week 2 8](#_Toc120548207)

[Week 3 9](#_Toc120548208)

[Week 4 10](#_Toc120548209)

[Week 5 11](#_Toc120548210)

[Week 6 12](#_Toc120548211)

[Week 7 13](#_Toc120548212)

[Week 8 14](#_Toc120548213)

[Week 9 15](#_Toc120548214)

[Week 10 16](#_Toc120548215)

[Additional information 17](#_Toc120548216)

[Further implementation support 17](#_Toc120548217)

[Assessment for learning 17](#_Toc120548218)

[Differentiation 18](#_Toc120548219)

[Support and alignment 20](#_Toc120548220)

[Evidence base 22](#_Toc120548221)

[References 24](#_Toc120548222)

## Rationale

The NSW Department of Education publishes a range of curriculum support materials, including samples of lesson sequences, scope and sequences, assessment tasks, examinations, student and teacher resource booklets, and curriculum planning and curriculum evaluation templates. The samples are not exhaustive and do not represent the only way to complete or engage in each of these processes. Curriculum design and implementation is a dynamic and contextually-specific process. While the mandatory components of syllabus implementation must be met by all schools, it is important that the approach taken by teachers is reflective of their needs and faculty/school processes.

NESA defines [programming](https://educationstandards.nsw.edu.au/wps/portal/nesa/k-10/understanding-the-curriculum/programming) as the process of ‘selecting and sequencing learning experiences which enable students to engage with syllabus outcomes and develop subject specific skills and knowledge’ ([NESA](https://educationstandards.nsw.edu.au/wps/portal/nesa/k-10/understanding-the-curriculum/programming) 2022). A program is developed collaboratively within a faculty. It differs from a unit in important ways, as outlined by NESA on their [advice on units](https://educationstandards.nsw.edu.au/wps/portal/nesa/k-10/understanding-the-curriculum/programming/advice-on-units) page. A unit is a contextually-specific plan for the intended teaching and learning for a particular class for a particular period. The organisation of the content in a unit is flexible and it may vary according to the school, the teacher, the class, and the learning space. They should be working documents that reflect the thoughtful planning and reflection that takes place during the teaching and learning cycle. There are mandatory components of programming and unit development, and this template provides one option for the delivery of these requirements. The NESA and department guidelines that have influenced this template are elaborated upon at the end of the document.

This resource has been developed to assist teachers in NSW Department of Education schools to create learning that is contextualised to their classroom. It can be used as a basis for the teacher’s own program, assessment, or scope and sequence, or be used as an example of how the new curriculum could be implemented. The resource has suggested timeframes that may need to be adjusted by the teacher to meet the needs of their students.

## Overview

**Description**: this program of learning addresses the focus area of programming fundamentals. The lessons and sequences in this program of learning are designed to allow students develop the knowledge and skills to design and create algorithms and software solutions to meet a user’s needs. This program is designed to be used in conjunction with the Programming Fundamentals Teacher Support Resource (TSR). The TSR is a comprehensive and detailed list of activities that addresses the Software Engineering syllabus content.

Teachers are advised to discern and select activities from the TSR for delivery in their contexts. The TSR includes a walkthrough of an educational software project that could be modified for alignment to the assessment task for this focus area.

Weeks 1–3 see 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 client and their class. This software enables input of three 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 during weeks 4–10.

During 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 final assessment task for this unit. Teachers may opt to extend the triangle recognition program as the assessment task for less experienced students.

**Duration**: this program of learning is designed to be completed over a period of approximately 10 weeks in 60-minute lesson sequences but can be adapted to suit the school context.

**Explicit teaching**: suggested learning intentions and success criteria are available for some lessons provided. Learning intentions and success criteria are most effective when they are contextualised to meet the needs of students in the class. The examples provided in this document are generalised to demonstrate how learning intentions and success criteria could be created.

## 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**
* describes how current hardware, software and emerging technologies influence the development of software engineering solutions **SE-11-03**
* applies safe and secure practices to collect, use and store data **SE-11-04**
* 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) © NSW Education Standards Authority (NESA) for and on behalf of the Crown in right of the State of New South Wales, 2022.

**Prior to planning for teaching and learning, please consider the following.**

**Engagement**

* How will I provide authentic, relevant learning opportunities for students to personally connect with lesson content?
* How will I support every student to grow in independence, confidence, and self-regulation?
* How will I facilitate every student to have high expectations for themselves?
* How will I identify and provide the support each student needs to sustain their learning efforts?

**Representation**

* What are some different ways I can present content to enable every student to access and understand it?
* How will I identify and address language and/or cultural considerations that may limit access to content for students?
* How will I make lesson content and learning materials more accessible?
* How will I plan learning experiences that are relevant and challenging for the full range of students in the classroom?

**Expression**

* How will I provide multiple ways for students to respond and express what they know?
* What tools and resources can students use to demonstrate their understanding?
* How will I know every student has understood the concepts and language presented in each lesson?
* How will I monitor if every student has achieved the learning outcomes and learning growth?

## Lesson sequence and details

### Week 1

Table 1 – lesson sequence and details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Outcomes and content | Teaching and learning activities | Evidence of learning | Differentiation or adjustments | Registration and evaluation notes |
| **Outcome:**  **SE-11-01**  **Content:**  Students:   * Explore fundamental software development steps used by programmers when designing software   Including:   * requirements definition * determining specifications * design * development * integration * testing and debugging * installation * maintenance | **Learning intention**  Understand the software development steps used by programmers.  **Success criteria**   * Determine the requirements for a new software solution. * Identify key specifications in the development of a software solution. * Explain the role of integration in the development of software solutions. * Apply appropriate design techniques. * Identify key features of the development step.   **Teaching and learning activity**  Teacher introduces the course and gives an overview of the learning in the topic and the assessment students will complete.  Students are introduced to glossary terms and complete a glossary for understanding the topic.  In groups, students explore scenarios to determine the requirements for the software solution.  Students articulate their understanding of their given problem with the class. | Students complete Activities 1 to 4 of the Teacher Support Resource (TSR).  Students participate in classroom discussion and contribute ideas on the Triangle recognition educational software requested from the Primary Teacher client.  Students complete the IPO chart for the project from the TSR.  Students can identify user requirements for the given system.  Students complete Activities 5 to 10 of the TSR.  Students create the pseudocode solution for the triangle recognition educational software.  Students participate in classroom discussion and contribute ideas.  Students select and implement appropriate data structures, algorithms and UI design in their project.  Students articulate the processes involved in the development step. | Suggested adjusted activities. This section is also for use in school when making adjustments to support all students to achieve in their learning.  There are no prerequisites for enrolment in Software Engineering.  Students will arrive with diverse experiences and abilities in programming.  Students complete a pre-test of fundamental concepts to inform the teacher of the depth required to address each content point of the syllabus. (Appendix 1 of the TSR).  Pre-teach key vocabulary and concepts prior to viewing videos, provide a transcript, and use closed captions when viewing. Provide a glossary and allow the use of bilingual dictionaries for uncommon terms.  Computational thinking skills pre-test using [Bebras resources](https://digitalcareers.csiro.au/en/Resources/Bebras-Unplugged)will assist to identify differences in ability and understanding. |  |

### Week 2

Table 2 – lesson sequence and details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Outcomes and content | Teaching and learning activities | Evidence of learning | Differentiation or adjustments | Registration and evaluation notes |
| **Outcomes:**  **SE-11-01, SE-11-03**  **Content:**  Students:   * Explore fundamental software development steps used by programmers when designing software * Research and evaluate the prevalence and use of online code collaboration tools | **Learning intention**  Understand the software development steps used by programmers.  **Success criteria**   * Identify and explain the role testing and debugging tools play in the development of software * Distinguish between debugging and testing tools * Modify existing program to meet new requirements * Discuss and model [debugging and testing tools](https://youtu.be/b7VbiZBg-dA) within the preferred development tools * Investigate a wide range of testing and debugging tools used within software engineering.   **Teaching and learning activity**  Teacher demonstrates the installation of a software application.  Students install software such as Addons in Google Chrome and/or project.  Students investigate and assess the prevalence of online code collaboration tools such as [Replit](https://replit.com) and [Codecollab](https://codecollab.io). | Students interpret the steps needed for success in the assessment task.  Students complete Activities 11 to 22 from the Teacher Support Resource (TSR).  Students commence entering bugs into their ‘Bug Book’.  Students create a list of debugging and testing tools.  Students distinguish between a range of debugging and testing tools.  Students successfully install applications.  Students complete Activities 23 to 25 from the TSR to analyse and evaluate online collaboration tools. | Suggested adjusted activities. This section is also for use in school when making adjustments to support all students to achieve in their learning.  The department’s [Digital Selector](https://app.education.nsw.gov.au/digital-learning-selector/) should be considered for Learning activities that assist students with:   * Remembering * Understanding * Applying * Analysing * Evaluating * Designing |  |

### Week 3

Table 3 – lesson sequence and details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Outcomes and content | Teaching and learning activities | Evidence of learning | Differentiation or adjustments | Registration and evaluation notes |
| **Outcomes:**  **SE-11-02, SE-11-04**  **Content:**  Students:   * 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 * Apply divide and conquer and backtracking as algorithmic design strategies * Develop structured algorithms using pseudocode and flowcharts, including the use of subprograms | **Learning intention**  Understand algorithms as they relate to computational thinking.  **Success criteria**   * Define the key features used in pseudocode and flowcharts. * Design and develop structured algorithms using pseudocode and flowcharts to solve a problem. * Correctly use control structures including sequence, selection, iteration, and subprograms.   **Teaching and learning activity**  Teacher:   * demonstrates [KWLH models](https://app.education.nsw.gov.au/digital-learning-selector/LearningActivity/Card/562). * identifies and displays a variety of definitions for an algorithm. * explains the key features of algorithms. * demonstrates the key features of pseudocode and flowcharts as they relate to algorithms * displays several solutions created in algorithmic form. * demonstrates how to construct a correct algorithm used to solve a given problem in both pseudocode and flowchart form. * demonstrates divide and conquer and backtracking as algorithmic design strategies.   Students:   * define an algorithm based on a range of previously created definitions from a range of sources. * watch [Santa's Dirty Socks (Divide and Conquer Algorithms) (3:40)](https://youtu.be/wVPCT1VjySA) * discuss [draw.io](https://app.diagrams.net/) as a CASE tool. algorithmic design strategies. | Assessment Task 1 (continued) with formative assessment and feedback.  Students complete Activities 26 to 29 from the Teacher Support Resource (TSR).  Students clearly articulate the meaning and purpose of an algorithm.  Students participate in classroom discussion and contribute ideas.  Students document and recall the key features and differences between pseudocode and flowcharts.  Students use [draw.io](https://app.diagrams.net/) to create flowcharts.  Students correctly create simple flowcharts and pseudocode solutions to a range of problems.  Students complete Activities 30 to 31 from the TSR.  Students complete Activities 39 to 44 from the TSR. | Suggested adjusted activities. This section is also for use in school when making adjustments to support all students to achieve in their learning.  The [Bebras Mini Challenge: Coding and Decoding](https://challenge.bebras.edu.au/index.php?action=user_competition&end=false&grp_id=100) provide activities to differentiate and adjust.  Cloze passage example should be rewritten for school context.  The following strategies to teach coding should be investigated:   * [Parsons puzzles](https://computerscienced.co.uk/site/parsons-puzzles/).   [PRIMM](https://primmportal.com/). |  |

### Week 4

Table 4 – lesson sequence and details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Outcomes and content | Teaching and learning activities | Evidence of learning | Differentiation or adjustments | Registration and evaluation notes |
| **Outcomes:**  **SE-11-01, SE-11-06, SE-11-07**  **Content:**  Students:   * Use modelling tools including structure charts, abstraction and refinement diagrams to support top-down and bottom-up design * Analyse the logic and structure of written algorithms Including: * determining inputs and outputs * determining the purpose of the algorithm * desk checking and peer checking * determining connections of written algorithms to other subroutines or functions * Identify procedures and functions in an algorithm | **Learning intention**  Understand modelling tools as they relate to developing software.  Experiment with coding in a range of different paradigms.  **Success criteria**   * Correctly use modelling tools to support top-down and bottom-up design. * Analyse the logic and structure of written algorithms.   **Teaching and learning activity**  Teacher:   * demonstrates a range of modelling tools including structure charts, abstraction, and refinement diagrams. * reviews and analyses several correctly produced algorithms and determines the following features * the purpose of the algorithm * inputs and outputs * desk checking and peer checking * connections of algorithms to subroutines or functions. * demonstrates how to identify procedures and functions in an algorithm. | Students interpret Assessment task 1 part 2 (continued formative assessment).  Students complete Activities 32 to 35 from the Teacher Support Resource (TSR).  Students create structure charts and abstraction and refinement diagrams to document several software solutions.  Students complete Activity 45 from the TSR.  Students analyse a range of algorithms determining the purpose and the inputs and outputs relevant to the solution.  Students complete Activity 46 from the TSR.  Students complete Activity 47 from the TSR.  Students complete Activity 48 from the TSR.  Students complete Activity 49 from the TSR.  Students identify connections related to algorithms including subroutines and functions.  Students complete Activities 50–51 from the TSR.  Students define and demonstrate an understanding of procedures and functions in an algorithm.  Students complete Activities 52–58 from the TSR. | Suggested adjusted activities. This section is also for use in school when making adjustments to support all students to achieve in their learning.  Modelling tools should be introduced as ways to inform clients and colleagues via graphic and readily understood diagrams. Course specifications for Software Engineering are essential here.  Students should be required to model different scenarios and solutions. |  |

### Week 5

Table 5 – lesson sequence and details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Outcomes and content | Teaching and learning activities | Evidence of learning | Differentiation or adjustments | Registration and evaluation notes |
| **Outcomes:**  **SE-11-02, SE-11-07**  **Content:**  Students:   * Experiment with object-oriented programming, imperative, logic and functional programming paradigms * Investigate the use of number systems for computing purposes, including binary, decimal and hexadecimal * Represent integers using two’s complement | **Learning intention**  Explore other programming paradigms and select the most appropriate.  **Success criteria**   * Correctly identify syntax from a range of programming paradigms including, imperative, logic, object orientated and functional paradigms.   **Learning intention**  Understand the purpose of representing base 10 numbers in Binary and Hexadecimal.  Represent numbers in Binary and Hexadecimal.  **Success criteria**   * Convert integers to Binary and Hexadecimal. * Convert negative integers into Binary using two’s complement.   **Teaching and learning activity**  Teacher presents to the class a video explaining the [purpose (12:49)](https://youtu.be/5xavJzo9EiY) and process of representing numbers in Binary.  As a class watch [how numbers are represented using the Hexadecimal number system (7:15)](https://youtu.be/GyTzzZYTwmQ).  Students:   * use the online [binary game](https://learningcontent.cisco.com/games/binary/index.html) to convert numbers into Binary. * perform simple conversions into Hexadecimal using online [tools](https://flippybitandtheattackofthehexadecimalsfrombase16.com/). * participate in the [online activity](https://wordwall.net/resource/9589924/computing/hexadecimal-to-decimal-match-up-game) for converting Hexadecimal numbers into Decimal. * explore the [two’s complement](https://www.wisc-online.com/learn/computer-science/it-networking/tmh9514/subtracting-binary-numbers-by-adding-the-2s-c) process.   Teacher:   * scaffolds the process of [converting negative numbers](https://youtu.be/lKTsv6iVxV4) into binary using two’s complement. | Students complete Activities 59–61 from the Teacher Support Resource (TSR).  Students complete a range of tutorials to an introductory level, experimenting in a range of programming paradigms including:   * Imperative * Logic * Functional * Object-oriented   Students complete Activities 62–83 from the TSR.  Students competently convert whole numbers into both Binary and Hexadecimal.  Students successfully convert Hexadecimal numbers into Decimal numbers.  Students demonstrate their understanding by applying the twos complement process to represent negative numbers. | Suggested adjusted activities. This section is also for use in school when making adjustments to support all students to achieve in their learning.  Sample code from other paradigms could be issued to students with keywords redacted for students to complete. Similar to a cloze passage exercise though using code.  Further explanation of [two’s complement (16:15)](https://youtu.be/lKTsv6iVxV4).  Differentiated [activity](https://www.csunplugged.org/en/topics/binary-numbers/how-binary-digits-work-junior/) for converting numbers into Binary.  The use of role play with students behaving like bits (on and off) is a very useful exercise in differentiation for all students. |  |

### Week 6

Table 6 – identifying and defining lesson sequence and details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Outcomes and content | Teaching and learning activities | Evidence of learning | Differentiation or adjustments | Registration and evaluation notes |
| **Outcomes:**  **SE-11-01, SE-11-02**  **Content:**  Students:   * Investigate standard data types * Create data dictionaries as a tool to describe data and data types, structure data, and record relationships. | **Learning intentions**  Know a range of data types used in software engineering including:   * char (character) and string * Boolean * real * single precision floating point * integer * date and time   Understand how data dictionaries describes the data, data types and record relationships used in an algorithm.  **Success criteria**   * Describe and uses appropriate data types in the development of software solutions. * Develop a data dictionary and decide on appropriate data types and relationships. | Assessment Task 1 (continued).  Students complete Activities 84-90 from the Teacher Support Resource (TSR).  Students complete Activities 91–92 from the TSR.  Students create data dictionaries which describe the data, data types and record relationships used in an algorithm.  Students independently create a data dictionary. | Suggested adjusted activities. This section is also for use in school when making adjustments to support all students to achieve in their learning.  Diagrammatic representation of cylinders (as variables) with lids that have different openings to represent various data types is a useful technique to introduce abstract topics.  Construct cards (or use sticky notes) and organise the examples based on the standard data types. |  |

### Week 7

Table 7 – lesson sequence and details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Outcomes and content | Teaching and learning activities | Evidence of learning | Differentiation or adjustments | Registration and evaluation notes |
| **Outcome:**  **SE-11-02**  **Content:**  Students:   * Use data structures of arrays, records, trees and sequential files. | **Learning intention**  Implement and use arrays, records, trees and sequential files in an appropriate programming language.  **Success criteria**   * Developed programs that include the use of arrays, records, trees and sequential files.   **Teaching and learning activity**  Define the standard data structures used in software engineering.  Using an appropriate IDE, demonstrate how to code the data structures:   * arrays * records * trees * sequential files.   Students:   * learn and use fundamental knowledge of programming and data structures to create a range of software solutions. know the purpose and function of each data structure including arrays, records, trees and sequential files. | Students complete Activities 93–99 from the Teacher Support Resource (TSR).  Students competently use an IDE.  Students successfully implement the data structures using the chosen language to solve problems.  Students continue work on Assessment task 1 including: Software Development, Designing Algorithms, Data for Software Engineering and Developing Solutions with Code. | Suggested adjusted activities. This section is also for use in school when making adjustments to support all students to achieve in their learning.  Scaffolding using diagrams and tracing flow can be useful techniques to introduce these abstract concepts. |  |

### Week 8

Table 8 – lesson sequence and details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Outcomes and content | Teaching and learning activities | Evidence of learning | Differentiation or adjustments | Registration and evaluation notes |
| **Outcomes:**  **SE-11-02, SE-11-06, SE-11-07**  **Content:**  Students:   * apply skills in computational thinking and programming to develop a software solution   Including:   * 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   Including:   * single and multidimensional arrays * lists * trees * stacks * hash tables | **Learning intention**  Describe the key concepts involved in developing a software solution.  **Success criteria**   * Identify and describe the key concepts in developing a software solution that the students have worked on.   **Teaching and learning activity**  Teacher led discussion on the coding solutions developed in class to this point including:   * triangle recognition * simple calculator * tic-tac-toe.   Students:   * Predict Run Investigate, Modify code for: * single and multidimensional arrays * lists * trees * stacks * hash tables   As a class watch [Introduction to Trees (Binary Tree) in Python - A Simplified Tutorial (11:26)](https://youtu.be/fUkrQD9nw0Y).  As a class watch Python Stacks.  As a class watch [What is a HashTable? (7:36)](https://youtu.be/MfhjkfocRR0)  As a class watch [The Hash Table Data Structure: A Historical and Technical Overview (17:30)](https://youtu.be/jt8mjox6vaU). | Students complete Activity 100 from the Teacher Support Resource (TSR).  Students complete Activities 101–104 from the TSR. | Suggested adjusted activities. This section is also for use in school when making adjustments to support all students to achieve in their learning.  Students should be required to present their findings to class as a Q&A response to their software solution. |  |

### Week 9

Table 9 – identifying and defining lesson sequence and details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Outcomes and content | Teaching and learning activities | Evidence of learning | Differentiation or adjustments | Registration and evaluation notes |
| **Outcomes:**  **SE-11-01, SE-11-03, SE-11-06, SE-11-07**  **Content:**  Students:   * Compare the execution of the Waterfall and Agile project management models as applied to software development * Test and evaluate solutions, considering key aspects including functionality, performance, readability of code, quality of documentation * Use debugging tools   Including:   * breakpoints * single line stepping * watches * interfaces between functions * debugging output statements * debugging software available in an integrated development environment (IDE) | **Learning intention**  Know the differences between Waterfall and Agile project management models and justify their use in a range of scenarios.  **Success criteria**   * Contribute to class debate and research examples of Waterfall and Agile Project management models.   **Teaching and learning activity**  Research:   * description * history * where its used or clients * advantages * disadvantages * case study of success and failure.   **Learning intention**   * To evaluate a software solution based on key aspects of: functionality, performance, readability of code, quality of documentation.   **Success criteria**   * Contribute to the peer evaluation of software solutions based on functionality, performance, readability of code, quality of documentation.   Each student presents their developed solution to class for evaluation. | Students complete Activity 105 from the Teacher Support Resource (TSR).  Students participate in a class debate.  The outcome of the debate should be judged by an independent colleague and/or class representatives designated to the role.  Students complete Activity 106 from the TSR.  Complete Activities 107–110 from the TSR. | Suggested adjusted activities. This section is also for use in school when making adjustments to support all students to achieve in their learning.  A collaborative jigsaw activity around research into Waterfall and Agile will engage all students in the activity. Providing role descriptions will assist in aligning topics with confidence.  Collaborative discussion while desk checking algorithms written on the whiteboard can assist to engage students. |  |

### Week 10

Table 10 – identifying and defining lesson sequence and details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Outcomes and content | Teaching and learning activities | Evidence of learning | Differentiation or adjustments | Registration and evaluation notes |
| **Outcomes:**  **SE-11-04, SE-11-06**  **Content:**  Students:   * Determine sets of suitable test data   Including:   * boundary values * path coverage * faulty and abnormal data * Determine typical errors experienced when developing code, including syntax, logic and runtime, and explain their likely causes | **Learning intention**  To understand the importance of test data in developing a software solution  **Success criteria**   * Determine sets of suitable test data   **Teaching and learning activity**  During class presentations students present the data they have used to test their software solution. | Formative assessment opportunities in class peer evaluation of code solutions.  Students complete Activities 107–110 from the Teacher Support Resource (TSR).  Submission of Bug Book. | Suggested adjusted activities. This section is also for use in school when making adjustments to support all students to achieve in their learning.  Use real world scenarios familiar to students like drivers license age or ride height entry can assist students with the logic for applying test data |  |

## Additional information

For additional support or advice, contact the TAS curriculum team by emailing [TAS@det.nsw.edu.au](mailto:TAS@det.nsw.edu.au).

### Further implementation support

Curriculum design and implementation is a dynamic and contextually-specific process. The department is committed to supporting teachers to meet the needs of all students. The advice below on assessment and planning for the needs of every student may be useful when considering the material presented in this sample program of learning.

### Assessment for learning

Possible formative assessment strategies that could be included:

* Learning intentions and success criteria assist educators to articulate the purpose of a learning task to make judgements about the quality of student learning. These help students focus on the task or activity taking place and what they are learning and provide a framework for reflection and feedback. [Online tools](https://app.education.nsw.gov.au/digital-learning-selector/LearningActivity/Card/622) can assist implementation of this formative assessment strategy.
* Eliciting evidence strategies allow teachers to determine the next steps in learning and assist teachers in evaluating the impact of teaching and learning activities. Strategies that may be added to a learning sequence to elicit evidence include all student response systems, [exit tickets](https://app.education.nsw.gov.au/digital-learning-selector/LearningActivity/Card/543), mini whiteboards (actual or [digital](https://app.education.nsw.gov.au/digital-learning-selector/LearningActivity/Card/575)), [hinge questions](https://app.education.nsw.gov.au/digital-learning-selector/LearningActivity/Card/557), [Kahoot](https://app.education.nsw.gov.au/digital-learning-selector/LearningTool/Card/621), [Socrative](https://app.education.nsw.gov.au/digital-learning-selector/LearningTool/Card/587), or quick quizzes to ensure that individual student progress can be monitored and the lesson sequence adjusted based on formative data collected.
* Feedback is designed to close the gap between current and desired performance by informing teacher and student behaviour (AITSL 2017). AITSL provides a [factsheet to support evidence-based feedback](https://www.aitsl.edu.au/teach/improve-practice/feedback#:~:text=FEEDBACK-,Factsheet,-A%20quick%20guide).
* [Peer feedback](https://app.education.nsw.gov.au/digital-learning-selector/LearningActivity/Card/549) is a structured process where students evaluate the work of their peers by providing valuable feedback in relation to learning intentions and success criteria. It can be supported by [online tools](https://app.education.nsw.gov.au/digital-learning-selector/LearningActivity/Browser?cache_id=1d29b).
* Self-regulated learning opportunities assist students in taking ownership of their own learning. A variety of strategies can be employed and some examples include reflection tasks, [Think-Pair-Share](https://app.education.nsw.gov.au/digital-learning-selector/LearningActivity/Card/645), [KWLH charts](https://app.education.nsw.gov.au/digital-learning-selector/LearningActivity/Card/562), [learning portfolios](https://app.education.nsw.gov.au/digital-learning-selector/LearningActivity/Card/583) and [learning logs](https://app.education.nsw.gov.au/digital-learning-selector/LearningActivity/Card/564).

The primary role of assessment is to establish where individuals are in their learning so that teaching can be differentiated and further learning progress can be monitored over time.

Feedback that focuses on improving tasks, processes and student self-regulation is the most effective. Students engaging with feedback can take many forms including formal, informal, formative, summative, interactive, demonstrable, visual, written, verbal and non-verbal.

[What works best update 2020](https://education.nsw.gov.au/about-us/educational-data/cese/publications/research-reports/what-works-best-2020-update) (CESE 2020a)

### Differentiation

Differentiated learning can be enabled by differentiating the teaching approach to content, process, product and the learning environment. For more information on differentiation go to [Differentiating learning](https://education.nsw.gov.au/teaching-and-learning/professional-learning/teacher-quality-and-accreditation/strong-start-great-teachers/refining-practice/differentiating-learning) and [Differentiation](https://education.nsw.gov.au/campaigns/inclusive-practice-hub/primary-school/teaching-strategies/differentiation).

When using these resources in the classroom, it is important for teachers to consider the needs of all students in their class, including:

* **Aboriginal and Torres Strait Islander students**. Targeted [strategies](https://education.nsw.gov.au/teaching-and-learning/aec/aboriginal-education-in-nsw-public-schools) can be used to achieve outcomes for Aboriginal students in K-12 and increase knowledge and understanding of Aboriginal histories and cultures. Teachers should utilise students’ Personalised Learning Pathways to support individual student needs and goals.
* **EAL/D learners**. EAL/D learners will require explicit English language support and scaffolding, informed by the [EAL/D enhanced teaching and learning cycle](https://education.nsw.gov.au/teaching-and-learning/curriculum/literacy-and-numeracy/resources-for-schools/eald/enhanced-teaching-and-learning-cycle) and the student’s phase on the [EAL/D Learning Progression](https://education.nsw.gov.au/teaching-and-learning/curriculum/multicultural-education/english-as-an-additional-language-or-dialect/planning-eald-support/english-language-proficiency). In addition, teachers can access information about [supporting EAL/D learners](https://education.nsw.gov.au/teaching-and-learning/curriculum/multicultural-education/english-as-an-additional-language-or-dialect/planning-eald-support/english-language-proficiency) and [literacy and numeracy support specific to EAL/D learners](https://education.nsw.gov.au/teaching-and-learning/curriculum/literacy-and-numeracy/resources-for-schools/eald).
* **Students with additional learning needs**. Learning adjustments enable students with disability and additional learning and support needs to access syllabus outcomes and content on the same basis as their peers. Teachers can use a range of [adjustments](https://education.nsw.gov.au/teaching-and-learning/disability-learning-and-support/personalised-support-for-learning/adjustments-to-teaching-and-learning) to ensure a personalised approach to student learning. Subject specific curriculum considerations can be found on the [Inclusive Practice hub](https://education.nsw.gov.au/campaigns/inclusive-practice-hub/primary-school/teaching-strategies/differentiation).
* **High potential and gifted learners**. [Assessing and identifying high potential and gifted learners](https://education.nsw.gov.au/teaching-and-learning/high-potential-and-gifted-education/supporting-educators/assess-and-identify#Assessment1) will help teachers decide which students may benefit from extension and additional challenge. [Effective strategies and contributors to achievement](https://education.nsw.gov.au/teaching-and-learning/high-potential-and-gifted-education/supporting-educators/evaluate) for high potential and gifted learners help teachers to identify and target areas for growth and improvement. In addition, the [Differentiation Adjustment Tool](https://education.nsw.gov.au/teaching-and-learning/high-potential-and-gifted-education/supporting-educators/implement/differentiation-adjustment-strategies) can be used to support the specific learning needs of high potential and gifted students. The [High Potential and Gifted Education Professional Learning and Resource Hub](https://schoolsnsw.sharepoint.com/sites/HPGEHub/SitePages/Home.aspx) supports school leaders and teachers to effectively implement the High Potential and Gifted Education Policy in their unique contexts.

All students need to be challenged and engaged to develop their potential fully. A culture of high expectations needs to be supported by strategies that both challenge and support student learning needs, such as through appropriate curriculum differentiation. (CESE 2020a:6).

### Support and alignment

**Resource evaluation and support**: all curriculum resources are prepared through a rigorous process. Resources are periodically reviewed as part of our ongoing evaluation plan to ensure currency, relevance, and effectiveness. For additional support or advice contact the TAS curriculum team by emailing [TAS@det.nsw.edu.au](mailto:TAS@det.nsw.edu.au).

**Alignment to system priorities and/or needs**: [School Excellence Policy](https://education.nsw.gov.au/policy-library/policies/pd-2016-0468), [School Success Model](https://education.nsw.gov.au/public-schools/school-success-model/school-success-model-explained).

**Alignment to the School Excellence Framework**: this resource supports the [School Excellence Framework](https://education.nsw.gov.au/teaching-and-learning/school-excellence-and-accountability/sef-evidence-guide/resources/about-sef) elements of curriculum (curriculum provision) and effective classroom practice (lesson planning, explicit teaching).

**Alignment to Australian Professional Teaching Standards**: this resource supports teachers to address [Australian Professional Teaching Standards](https://educationstandards.nsw.edu.au/wps/portal/nesa/teacher-accreditation/meeting-requirements/the-standards/proficient-teacher) 3.2.2, 3.3.2.

**Consulted with**: Curriculum and Reform and subject matter experts.

**NSW syllabus**: Software Engineering 11-12

**Syllabus outcomes**: SE-11-01, SE-11-02, SE11-03, SE11-04, SE-11-06 and SE-11-07.

**Author**: TAS Team, Curriculum Secondary Learners, Curriculum Reform

**Publisher**: State of NSW, Department of Education

**Resource**: Program of learning

**Related resources**: further resources to support Software Engineering 11-12 can be found on the [TAS curriculum page](https://education.nsw.gov.au/teaching-and-learning/curriculum/tas).

**Professional learning**: relevant professional learning is available through [HSC Professional Learning](https://education.nsw.gov.au/teaching-and-learning/professional-learning/hsc-pl) or on the [TAS curriculum page](https://education.nsw.gov.au/teaching-and-learning/curriculum/tas).

**Creation date**: 2023

**Rights**: © State of New South Wales, Department of Education.

## Evidence base

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

AITSL (Australian Institute for Teaching and School Leadership Limited) (n.d) [*Learning intentions and success criteria* [PDF 251KB]](https://www.aitsl.edu.au/docs/default-source/feedback/aitsl-learning-intentions-and-success-criteria-strategy.pdf?sfvrsn=382dec3c_2#:~:text=Learning%20Intentions%20are%20descriptions%20of,providing%20feedback%20and%20assessing%20achievement.), AITSL, accessed 31 May 2023.

AITSL (2017) ‘[Feedback Factsheet](https://www.aitsl.edu.au/teach/improve-practice/feedback#:~:text=FEEDBACK-,Factsheet,-A%20quick%20guide)’, AITSL, accessed 31 May 2023.

Brookhart S (2011) *How to Assess Higher-Order Thinking Skills in Your Classroom*, Hawker Brownlow Education, Victoria.

CESE (Centre for Education Statistics and Evaluation) (2020a) ‘[What works best 2020 update](https://education.nsw.gov.au/about-us/educational-data/cese/publications/research-reports/what-works-best-2020-update)’, NSW Department of Education, accessed 31 May 2023.

CESE (Centre for Education Statistics and Evaluation) (2020b) ‘[What works best in practice](https://education.nsw.gov.au/about-us/educational-data/cese/publications/practical-guides-for-educators-/what-works-best-in-practice)’, NSW Department of Education, accessed 31 May 2023.

NESA (NSW Education Standards Authority) (2022) ‘[Advice on units](https://educationstandards.nsw.edu.au/wps/portal/nesa/k-10/understanding-the-curriculum/programming/advice-on-units)’, *Understanding the curriculum,* NESA website, accessed 31 May 2023

NESA (2022) ‘[Proficient Teacher: Standard descriptors](https://educationstandards.nsw.edu.au/wps/portal/nesa/teacher-accreditation/meeting-requirements/the-standards/proficient-teacher)’, The Standards, NESA website, accessed 31 May 2023.

NESA (2022) ‘[Programming](https://educationstandards.nsw.edu.au/wps/portal/nesa/k-10/understanding-the-curriculum/programming)’, *Understanding the curriculum*, NESA website, accessed 31 May 2023.

Rosenshine B (2012) ‘[Principles of Instruction: Research-Based Strategies That All Teachers Should Know’](https://www.aft.org/periodical/american-educator/spring-2012), *American Educator*, 36(1):12-19, 39, ISSN-0148-432X, accessed 31 May 2023.

Wiliam D (2013) [‘Assessment: The bridge between teaching and learning’](https://www.researchgate.net/publication/258423377_Assessment_The_bridge_between_teaching_and_learning), *Voices from the Middle*, 21(2):15–20, accessed 31 May 2023.

Wiliam D (2017) *Embedded Formative Assessment*, 2nd edn, Solution Tree Press, Bloomington, IN.

Wisniewski B, Zierer K and Hattie J (2020) [‘The Power of Feedback Revisited: A Meta-Analysis of Educational Feedback Research’](https://doi.org/10.3389/fpsyg.2019.03087), *Frontiers In Psychology*, 10:3087, doi:10.3389/fpsyg.2019.03087, accessed 31 May 2023.

## 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.

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.

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