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

The object-oriented paradigm

Contents

[Rationale 3](#_Toc150844797)

[Overview 4](#_Toc150844798)

[Outcomes 6](#_Toc150844799)

[Lesson sequence and details 8](#_Toc150844800)

[Week 1 8](#_Toc150844801)

[Week 2 13](#_Toc150844802)

[Week 3 16](#_Toc150844803)

[Week 4 19](#_Toc150844804)

[Weeks 5–8 22](#_Toc150844805)

[Week 9 27](#_Toc150844806)

[Week 10 30](#_Toc150844807)

[Additional information 33](#_Toc150844808)

[Further implementation support 33](#_Toc150844809)

[Assessment for learning 33](#_Toc150844810)

[Differentiation 34](#_Toc150844811)

[Support and alignment 35](#_Toc150844812)

[Evidence base 37](#_Toc150844813)

[References 39](#_Toc150844814)

# 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 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**: In this unit students will develop a fundamental understanding of the object-oriented paradigm (OOP). Students will investigate the key concepts of this paradigm. They will differentiate between OOP and other paradigms investigated in the fundamentals of programming focus area. Students will be guided through programming and coding activities that reinforce their understanding.

During Weeks 1 and 2 of the learning sequence, students will be introduced to objects, methods and attributes by identifying their use in Python's Turtle module. They will then use starting code provided in the Appendix of the teacher support resource (TSR) to write their own classes, objects, attributes and methods to draw shapes.

During Weeks 3 and 4 of the learning sequence, students complete a series of workbook activities to consolidate their understanding and learn about modelling tools used in object-oriented programming (OOP).

During Weeks 5 to 8 of the learning sequence, students are guided step-by-step through the creation of a text-based role-playing game (RPG) using key OOP concepts.

During Weeks 9 and 10 of the learning sequence, students test, evaluate and submit their projects. Students present their work to the class and are peer assessed via a Q&A session.

The TSR is mapped directly to the content from the object-oriented paradigm focus area of the Software Engineering 11–12 syllabus, assumes no prior knowledge of OOP and introduces the unit through a series of simple coding activities. The skills required for success in this unit include [computational thinking [PDF 41.6 KB]](https://www.csiro.au/-/media/Digital-Careers/Files/Resources/CTIA-Worksheets/DigitalCareers_CTIAWorksheets_CTdefinitions.pdf), design thinking and systems thinking skills.

**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**
* applies tools and resources to design, develop, manage and evaluate software **SE-11-06**
* implements safe and secure programming solutions **SE-11-07**
* applies language structures to refine code **SE-11-08**
* manages and documents the development of a software project **SE-11-09**

[Software Engineering 11–12 Syllabus](https://curriculum.nsw.edu.au/learning-areas/tas/software-engineering-11-12-2022/overview) © 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/ adjustments | Registration and evaluation notes |
| **SE-11-01**  **SE-11-02**  **Understanding OOP**  Students:   * apply the key features of an object-oriented programming (OOP) language * compare procedural programming with OOP. | **Learning intention**  Identify and define the fundamental concepts of the object-oriented paradigm.  Modify sample code that uses key concepts of the object-oriented paradigm.  Distinguish between samples of procedural programming and OOP code.  Apply the key features of an object-oriented programming (OOP) language.  **Success criteria**   * I can identify and define class, object, instance, attribute and methods. * I can modify code that uses the key concepts of the object-oriented paradigm. * I can use specialist terminology. * I can distinguish between procedural and OOP sample code. * I can apply the key features of an object-oriented programming (OOP) language.   **Teaching and learning activity**  Students complete teacher support resource (TSR) **Activities 1–4**.  Teacher-led discussion an historical overview of how programming was first developed.  Students   * review the programming generations and discuss first gen and second gen code fragments * review the ‘Tic-Tac-Toe’ project from the Programming fundamentals unit (beginning from **Activity 32** of TSR).   Teacher-led discussion on the imperative paradigm and review simple Python commands in a console development environment.  Teacher introduces the concept of object-oriented paradigms as the one most familiar to students.  Teacher-led discussion on the differences between imperative/procedural and object-oriented paradigm including:   * discussion of the strengths of different paradigms * difficulty of solving specific requirements. * emerging technologies.   Students compare and contrast the key differences between the imperative paradigm and object-oriented paradigm.  Teacher explains purpose of private and public classes.  Students watch the introduction [video on object-oriented language](https://www.youtube.com/watch?v=SS-9y0H3Si8).  Teacher-led discussion on the key terms: object, class, encapsulation, abstraction, attribute, inheritance, instantiation, generalisation, method and polymorphism.  Students provided with project solutions in Mu or Visual Studio based on the object-oriented paradigm. Students identify all the OOP language in the code project.  Students complete **Activities 5–15** of the TSR. | Students discuss and add to their glossary throughout the learning sequence to assist in the correct use of specialist terminology.  Students identify and define class, object, instance, attribute and methods by recognising their use in code samples.  Students modify code in a simple exercise using Python Turtle to demonstrate the use of key concepts and discuss their understanding.  Students share, discuss and debate most suitable analogies for OOP.  Students participate in classroom discussion and contribute ideas.  Students can identify user attributes and methods of various objects in code.  Students can identify and explain the key terms of object-oriented paradigm.  Students participate in classroom discussion and contribute ideas.  Students select and implement appropriate data structures, algorithms and user interface (UI) design in their project.  Students articulate the key terms in OOP.  Students create, analyse and modify object-oriented code. | This section is also for use in school when making adjustments to support all students to achieve in their learning.  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 and use visuals where appropriate.  The flipped classroom has students learning in their own time to dedicate class time to hands-on learning and interactive discussion. For homework, students watch a detailed description of key concepts:  [Fundamental Concepts of Object- Oriented Programming (9:15)](https://youtu.be/m_MQYyJpIjg?si=S60zXfZIGu3LHx1X).  Homework: students visit the ‘[Tower of Hanoi’](https://www.mathsisfun.com/games/towerofhanoi.html) website and interact with the [game](https://www.mathsisfun.com/games/towerofhanoi.html). In class they discuss how the various discs and poles could interact, requesting permission from each other (object-oriented). They outline how they could also be instructed to follow a direct sequence (imperative). |  |

## Week 2

Table 2 – lesson sequence and details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Outcomes and content | Teaching and learning activities | Evidence of learning | Differentiation/ adjustments | Registration and evaluation notes |
| **SE-11-01**  **SE-11-02**  **SE-11-06**  **SE-11-08**  Students:   * apply the key features of an object-oriented programming (OOP) language * compare procedural programming with OOP. | **Learning intention**  Understand and apply the concepts of the object-oriented paradigm.  **Success criteria**   * I can write OOP code to create a class, object, instance, attribute and method. * I can identify and define: encapsulation, abstraction, inheritance, instantiation, generalisation and polymorphism.   **Teaching and learning activity**  Teacher-led discussion on the imperative paradigm and review of simple Python commands in a console development environment.  Students revise:   * the concept of object-oriented paradigms * the key terms: object, class , attribute , inheritance , method and polymorphism.   Students build project solutions in Mu, Python or Visual Studio based on the object-oriented paradigm. Students describe and demonstrate the use of all the OOP language in the code projects.  Students complete **Activities 16–27** of the TSR. | Students answer class quiz questions on terminology.  Students participate in classroom discussion and contribute ideas.  Students can recite and explain the key terms of object-oriented paradigm.  Students can develop code with specific reference to the attributes and methods of various objects in code.  Students can interpret, modify or adapt code examples with explicit reference to properties, attributes and polymorphism. | This section is also for use in school when making adjustments to support all students to achieve in their learning.  Students modify sample code: classes, objects, attributes and methods to draw other shapes. |  |

## Week 3

Table 3 – lesson sequence and details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Outcomes and content | Teaching and learning activities | Evidence of learning | Differentiation/ adjustments | Registration and evaluation notes |
| **SE-11-01**  **SE-11-02**  **SE-11-06**  **SE-11-08**  Students:   * use data flow diagrams, structure charts and class diagrams to represent a system. | **Learning intention**  Develop an understanding of system modelling as it relates to computational thinking in the object -oriented programming (OOP) context.  **Success criteria**   * I can identify and define the key features of a system using modelling tools in OOP.   **Teaching and learning activity**  Students review and revise the IPO diagram, abstraction and refinement diagrams from the previous Programming Fundamentals unit.  Teacher outlines the primary tools needed and introduces students to the various rationales and needs for each type:   * data flow diagram * structure chart * class diagram * abstraction and refinement * IPO diagram * Gantt chart.   Teacher demonstrates the use of data flow diagrams with examples. Teacher explicitly describes each shape and its purpose including outlining the key concepts:   * external entity * data flow * data storage * processes.   Students design and develop data flow diagrams to model the solution to a problem.  Students correct and modify existing data flow diagrams.  Students complete **Activities 28–49** of the TSR. | Students refer to the [Software Engineering Course Specifications](https://curriculum.nsw.edu.au/learning-areas/tas/software-engineering-11-12-2022/overview#software-engineering-course-specifications-software_engineering_11_12_2022) to demonstrate the following:   * data flow diagram * structure chart * class diagram * abstraction and refinement * IPO diagram * Gantt chart.   Students contribute to a class quiz reviewing IPO diagram and abstraction and refinement diagrams from the previous Programming fundamentals unit.  Students can clearly articulate the meaning and purpose of a system modelling tool.  Students recall and describe the purposes of IPO diagrams and abstraction diagrams.  Students participate in classroom discussion and contribute ideas.  Students attempt system modelling scenarios and questions from past HSC Software Design and Development examination papers.  Students make use of a Software Development Drawing tool to construct data flow diagrams.  Students can read, interpret, build, correct and modify data flow diagrams.  Students recount the steps to construct a structure chart.  Students design and create informative class diagrams for the turtle and draw shapes coding activities. | This section is also for use in school when making adjustments to support all students to achieve in their learning.  Students model the turtle and shapes code they have written and consider system modelling of a game they are familiar with. |  |

## Week 4

Table 4 – lesson sequence and details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Outcomes and content | Teaching and learning activities | Evidence of learning | Differentiation/ adjustments | Registration and evaluation notes |
| **SE-11-01**  **SE-11-02**  **SE-11-06**  **SE-11-08**  Students:   * describe the process of design used to develop code in an OOP language * assess the effectiveness of programming code developed to implement an algorithm * investigate how OOP languages handle message-passing between objects * explain code optimisation in software engineering * outline the features of OOP that support collaborative code development. | **Learning intention**  Describe the process of design used to develop code in an object-oriented programming (OOP) language.  **Success criteria**   * I can describe the process of design used to develop code in an OOP language including. task definition, top-down and bottom-up, facade pattern and agility. * I can assess the effectiveness of programming code developed to implement an algorithm. * I understand how OOP languages handle message-passing between objects. * I can explain code optimisation in software engineering. * I can outline the features of OOP that support collaborative code development including: * consistency, code commenting, version control and feedback.   **Teaching and learning activity**  Teacher models [debugging and testing tools](https://youtu.be/b7VbiZBg-dA) within the preferred development tools.  Students investigate a wide range of testing and debugging tools used within software engineering.  Teacher-led discussion on the process of design used to develop code in an OOP language:   * task definition, top-down and bottom-up, facade pattern and agility.   Teacher outlines the features of OOP that support collaborative code development:   * consistency, code commenting, version control and feedback.   Students complete **Activities 50–66** of the TSR. | Students participate in classroom discussion and contribute ideas on the process of design used to develop code in an OOP language.  Students create a list of debugging and testing tools.  Students can distinguish between a range of debugging and testing tools.  Students can describe a range of design tools used to develop code.  Student can describe and demonstrate the advantages of coding in a collaborative environment.  Students analyse scenarios to answer questions on inheritance, polymorphism and message-passing.  Students recognise the process of design used to develop code in the OOP activities (turtle and draw shapes) they have completed.  Students recognise the features of OOP that support collaborative code development in the OOP activities (turtle and draw shapes) they have completed. | This section is also for use in school when making adjustments to support all students to achieve in their learning. |  |

## Weeks 5–8

Table 5 – lesson sequence and details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Outcomes and content | Teaching and learning activities | Evidence of learning | Differentiation/ adjustments | Registration and evaluation notes |
| **SE-11-06**  **SE-11-08**  **Programming in OOP**  Students:   * design and implement computer programs involving branching, iteration and functions in an OOP language for an identified need or opportunity * implement and modify OOP programming code. | **Learning intention**  Design and implement computer programs in an OOP language for an identified need or opportunity.  **Success criteria**   * I can design computer programs involving branching, iteration and functions in an object-oriented programming (OOP) language for an identified need or opportunity. * I can read, interpret and modify OOP programming code.   **Teaching and learning activity**  Teachers and students negotiate on the criteria for success of a text-based adventure game written in OOP.  Teacher leads a discussion on classes, objects and characters within a computer game.  Teacher guides the students through the creation of the text-based adventure game using OOP in the TSR.  Teacher explicitly stops, discusses and demonstrates the use of key OOP concepts as the game is being written. Students complete the accompanying activities and discuss findings.  Teacher uses the PRIMM strategy at any point where sample code is issued to students:   * Predict what the code will do * Run the code * Investigate how the code works * Modify the code and see how it works * Make your own code by rewriting it.   Throughout the game development, the teacher explains and demonstrates various techniques for developing and executing OO code, including:   * clear and uncluttered mainline, one logical task per subroutine, use of stubs * use of control structures and data structures * ease of maintenance * version control and regular backup.   Throughout the development of the game the teacher explicitly describes:   * each routine, its purpose and relationship to other modules, including passing of parameters * connection between mainline programming and subroutines to structure charts.   Throughout the development of the game the teacher demonstrates industry standard coding practices, such as camel code and explains the connection of code maintenance.  Throughout the development of the game the teacher articulates importance of version control, backups and links to good assessment practice, and submission of drafts.  Students complete **Challenges 1–26** and all steps in the game coding in the TSR including: Part 1 – setting the scene, Part 2 – adding the characters, Part 3 – interacting with the game, Part 4 – extending the game. | Students participate in classroom discussion and contribute ideas.  Students explain why object-oriented programming (OOP) languages are particularly suited to the development of computer games.  Students create a text-based adventure game using OOP and successfully complete the coding activities in the TSR including:   * setting the scene * adding the characters * interacting with the game * extending the game.   Students explain how their text-based adventure game works using OOP by maintaining a journal of their work.  Students demonstrate the use of the PRIMM strategy including discussing their predictions for what the code will do and their investigations after the code has been run.  During the game development students demonstrate:   * clear and uncluttered mainline, one logical task per subroutine, use of stubs, use of control structures and data structures, ease of maintenance, version control, regular backup, passing variables between modules. | Teachers may introduce authentic and real-world software engineering like collaborating on the development of code via modules and groupwork within GitLive or Replit.  Students should be encouraged to adapt the site and scenario for the adventure game from:   * caves to space, forests, islands, classrooms * characters from monsters to astronauts, elves, pirates or teachers.   Students should be encouraged with a creative approach to the gameplay for originality.  A list of possible extensions is provided in the TSR.  Teacher demonstrates Python using different IDEs, for example Visual Studio, and the use of stubs, modules with examples.  Students could be issued with a [differentiated task](https://education.nsw.gov.au/teaching-and-learning/high-potential-and-gifted-education/supporting-educators/implement/differentiation-adjustment-strategies) where they aspire to a project just beyond their own immediate comfort and experience with programming. Students should aspire to a personal best with experienced programmers expected to produce a fully functional graphic or asset-rich OOP game, and inexperienced programmers required to at least complete the given task according to the TSR. (This can be challenging to equitably assess, though can be of enormous motivational value to students intimidated by the ability of their peers and lifts the ceiling of attainment for most capable students.) |  |

## Week 9

Table 6 – lesson sequence and details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Outcomes and content | Teaching and learning activities | Evidence of learning | Differentiation/ adjustments | Registration and evaluation notes |
| **SE-11-01**  **SE-11-02**  **SE-11-09**  Students:   * apply methodologies to test code. | **Learning intention**  Apply methodologies to test and evaluate code.  **Success criteria**   * I can apply methodologies to test and evaluate code including: unit, subsystem and system testing; black, white and grey box testing; quality assurance.   **Teaching and learning activity**  Teacher explains and demonstrates various techniques for testing and evaluating OOP code, including: unit, subsystem and system testing; black, white and grey box testing; quality assurance.  Teacher explicitly describes each of the various techniques, their purpose and relationship to other forms of testing. Teacher connects the concept to industry standards such as how major gaming or operating system companies would perform various types of testing.  Teacher describes connection between mainline programming and subroutines to structure charts.  Teacher demonstrates industry standard coding practices, such as camel code and explains the connection of code maintenance.  Teacher articulates importance of version control, backups and links to good assessment practice, and submission of drafts.  Students complete **Activities 67–79** of the TSR. | Students participate in classroom discussion and contribute ideas.  Students can articulate and describe the testing techniques, the differences between each, and their purposes. | This section is also for use in school when making adjustments to support all students to achieve in their learning. |  |

## Week 10

Table 7 – lesson sequence and details

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Outcomes and content | Teaching and learning activities | Evidence of learning | Differentiation/ adjustments | Registration and evaluation notes |
| **SE-11-01**  **SE-11-02**  **SE-11-09**  Students:   * apply methodologies to test and evaluate code. | **Learning intention**  Describe methods used to plan, develop and engineer software solutions.  Explain how structural elements are used to develop programming code.  Manage and document the development of a software project.  **Success criteria**   * I can describe methods used to plan, develop and engineer software solutions. * I can explain how structural elements are used to develop programming code. * I can manage and document the development of a software project.   **Teaching and learning activity**  Teachers model questions to be asked of students during their class presentations and the Q&A session.  Teachers apply [the 5 Whys](https://app.education.nsw.gov.au/digital-learning-selector/LearningActivity/Card/638) to provide depth and critical thinking to these questions.  Teachers and students apply the negotiated criteria to peer assess the presentations in accordance with the success criteria established prior to the commencement of the project.  Students present to the class, respond to Q&A session and participate in peer assessment. | Students submit:   * workbook activities from the TSR * a role-playing text-based adventure game written (using object-oriented programming (OOP) languages) in Python * documentation digitally including: * a journal documenting the progress and pitfalls (see [Gregory Yobs account of the original version of Hunt the Wumpus](https://www.atariarchives.org/bcc1/showpage.php?page=247)) * a structure chart of the game * a data dictionary * a class diagram * testing strategies * sets of test data, that test the most common inputs and the appropriate expected results * a complete copy of the code (using internal and intrinsic documentation) * evaluation of the final game * class presentation with Q&A session. | This section is also for use in school when making adjustments to support all students to achieve in their learning.  High-performing students should be permitted time to provide thorough explanations of their work, with presentations used ‘for’ and ‘as’ learning and not simply ‘of’ learning. |  |

# 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/560), [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/583).

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 (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).
* **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/policy-library/policies/pd-2016-0468) 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, SE-11-06, SE-11-07, SE-11-08, SE-11-09

**Author**: TAS, 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 in the TAS statewide staffroom.

**Creation date**: 2023

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

# Evidence base

[Software Engineering 11–12 Syllabus](https://curriculum.nsw.edu.au/learning-areas/tas/software-engineering-11-12-2022/overview) © 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 (2017) ‘[Feedback Factsheet](https://www.aitsl.edu.au/teach/improve-practice/feedback#:~:text=FEEDBACK-,Factsheet,-A%20quick%20guide)’, AITSL, accessed 25 October 2023.

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 25 October 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 25 October 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 25 October 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 25 October 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 25 October 2023.

NESA (2022) ‘[Programming](https://educationstandards.nsw.edu.au/wps/portal/nesa/k-10/understanding-the-curriculum/programming)’, *Understanding the curriculum*, NESA website, accessed 25 October 2023.

Rosenshine B (2012) ‘[Principles of Instruction: Research-Based Strategies That All Teachers Should Know’ [PDF 192KB]](https://www.aft.org/sites/default/files/Rosenshine.pdf), *American Educator*, 36(1):12–19, ISSN-0148-432X, accessed 25 October 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 25 October 2023.

Wiliam D (2017) *Embedded Formative Assessment*, 2nd ed, 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.

# 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](https://aus01.safelinks.protection.outlook.com/?url=https%3A%2F%2Feducationstandards.nsw.edu.au%2Fwps%2Fportal%2Fnesa%2Fmini-footer%2Fcopyright&data=05%7C01%7CCaitlin.Pace1%40det.nsw.edu.au%7C9c2c1a9f59c94d2df30708dafa7edb23%7C05a0e69a418a47c19c259387261bf991%7C0%7C0%7C638097720042599463%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=BzQh0UsffVZE3eO22b2Xba3p0VMOBZSHfS21FGHXtZM%3D&reserved=0).

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/> and the NSW Curriculum website [https://curriculum.nsw.edu.au/home](https://curriculum.nsw.edu.au/).

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

Amos D (2023) Real Python [*Object-Oriented Programming (OOP) in Python 3*](https://realpython.com/python3-object-oriented-programming/)*,* Real Python website, accessed 6 September 2023.

Computer Science (2 November 2020) ['Fundamental Concepts of Object Oriented Programming' [video]](https://www.youtube.com/watch?v=m_MQYyJpIjg), *Computer Science*, YouTube, accessed 10 October 2023.

Coursera (2023) ‘[Object-Oriented Design](https://www.coursera.org/lecture/object-oriented-design/1-2-6-generalization-xjMaQ)’ *1.2.6 – Generalization*, Coursera website, accessed 9 October 2023.

Digital Learning and Teaching Victoria (2021) [*Object-oriented programming: getting started for Australian teachers*](https://sites.google.com/dltv.vic.edu.au/object-oriented), Digital Learning and Teaching Victoria website, accessed 9October 2023.

GitHub,Inc (2023) ['Pygame Zero'](https://github.com/RPi-Distro/pgzero), *pgzero*, GitHub website, accessed 6 September 2023.

Invent With Python (n.d.) [*Object-Oriented Programming and Classes*](https://inventwithpython.com/beyond/chapter15.html), *Chapter 15*, Invent with Python website, accessed 6 September 2023.

Microsoft (2023) [*Visual Studio Code*](https://code.visualstudio.com/Download), Microsoft website, accessed 6 September 2023.

MIT OpenCourseWare (16 February 2017) ['8. Object Oriented Programming' [video]](https://www.youtube.com/watch?v=-DP1i2ZU9gk), *MIT OpenCourseWare*, YouTube, accessed 10 October 2023.

OOP workshop (12 May 2021) [Replit site](https://replit.com/@OOPWorkshop/1petvariablessol#main.py) [website], accessed 6 September 2023.

Poylas A (28 December 2018) [‘Design Patterns — A quick guide to Facade pattern](https://medium.com/@andreaspoyias/design-patterns-a-quick-guide-to-facade-pattern-16e3d2f1bfb6)’, *Medium*, accessed 6 September 2023.

Python Software Foundation (2001–2023) [*Python*](https://www.python.org/) [website], accessed 6 September 2023.

Raspberry Pi Foundation (n.d.) [Turtle Race!](https://projects.raspberrypi.org/en/projects/turtle-race) [website], accessed 1 November 2023.

Schafer C (21 June 2016) ['Python OOP Tutorial 1: Classes and Instances' [video]](https://youtu.be/ZDa-Z5JzLYM?si=UPX6p5o3ICXyMG7p), *Corey Schafer*, YouTube, accessed 10 October 2023.

Sweigart A [Tic-Tac-Toe Non-OOP and Tic-Tac-Toe OOP](https://autbor.com/compareoop/#diff) [website], accessed 6 September 2023.

Tollervey NH (2023) [Code with Mu: a simple Python editor for beginner programmers](https://codewith.mu/) [website], accessed 6 September 2023.

Wikipedia contributors (2022) [*Hunt the Wumpus*](https://en.wikipedia.org/wiki/Hunt_the_Wumpus), Wikipedia website, accessed 6 September 2023.

Wingware (1999–2023) [Wing Python IDE](https://wingware.com/downloads) [website], accessed 6 September 2023.

W3Schools (1999–2023) [What Is OOP?](https://www.w3schools.com/cs/cs_oop.php) [website], accessed 6 September 2023.

Ying K (22 September 2020) ['Inheritance/Polymorphism in Object Oriented Programming | Python for Beginners | Code with Kylie #10' [video]](https://youtu.be/C2QfkDcQ5MU?si=k6_v0tIc50WXJWKA), *Kylie Ying*, YouTube, accessed 10 October 2023.

Yob G (n.d.) [*Hunt the Wumpus*](https://www.atariarchives.org/bcc1/showpage.php?page=247)*,* atariarchives.org, accessed 9 October 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) license](https://creativecommons.org/licenses/by/4.0/).

[](https://creativecommons.org/licenses/by/4.0/)

This license 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 license:

* 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.