



Technologies: Digital Technologies

Teaching, learning and assessment exemplar

Year 8

Python text adventure game



Acknowledgement of Country

Kaya. The School Curriculum and Standards Authority (the Authority) acknowledges that our offices are on Whadjuk Noongar boodjar and that we deliver our services on the country of many traditional custodians and language groups throughout Western Australia. The Authority acknowledges the traditional custodians throughout Western Australia and their continuing connection to land, waters and community. We offer our respect to Elders past and present.

Background

This teaching, learning and assessment exemplar (the exemplar) has been developed by the School Curriculum and Standards Authority (the Authority) as part of the *School Education Act Employees (Teachers and Administrators) General Agreement 2017* (Clause 61.1–61.3).

Copyright

© School Curriculum and Standards Authority, 2026

This document – apart from any third-party copyright material contained in it – may be freely copied, or communicated on an intranet, for non-commercial purposes in educational institutions, provided that the School Curriculum and Standards Authority (the Authority) is acknowledged as the copyright owner, and that the Authority’s moral rights are not infringed.

Copying or communication for any other purpose can be done only within the terms of the *Copyright Act 1968* or with prior written permission of the Authority. Copying or communication of any third-party copyright material can be done only within the terms of the *Copyright Act 1968* or with permission of the copyright owners.

Any content in this document that has been derived from the Australian Curriculum may be used under the terms of the [Creative Commons Attribution 4.0 International licence](#).

Disclaimer

Any resources such as texts, websites and so on that may be referred to in this document are provided as examples of resources that teachers can use to support their learning programs. Their inclusion does not imply that they are mandated or that they are the only resources relevant to the course. Teachers must exercise their professional judgement as to the appropriateness of any they may wish to use.

This resource utilises electronic web-based resources, such as videos and image galleries. Teachers should be present while an electronic resource is in use and close links immediately after a resource, such as a video has played to prevent default ‘auto play’ of additional videos. Where resources are referred for home study, they should be uploaded through Connect, or an equivalent system, that filters advertising content.

Contents

The Western Australian Curriculum	1
The Technologies curriculum	1
This exemplar.....	2
Catering for diversity.....	2
Using this exemplar.....	3
Links to electronic resources.....	3
Best practice	4
Teaching and learning	4
Assessing	4
Reflecting.....	4
Python text adventure game	5
Year level description	6
Achievement standard	7
Lessons 1–12.....	9
Appendix A	29
Appendix B	31
Appendix C.....	37

The Western Australian Curriculum

The *Western Australian Curriculum and Assessment Outline* (the *Outline* – <https://k10outline.scsa.wa.edu.au/>) sets out the mandated curriculum, guiding principles for teaching, learning and assessment, and support for teachers in their assessment and reporting of student achievement. The *Outline* recognises that all students in Australian schools, or international schools implementing the Western Australian Curriculum, are entitled to be given access to the eight learning areas described in the *Alice Springs (Mparntwe) Education Declaration*, December 2019.

The Technologies curriculum

The mandated curriculum is presented in the year level syllabus documents.

The Technologies curriculum delivers a sequential and age-appropriate progression of learning with the following key elements:

- a year level description that provides an overview of the context for teaching and learning in the year
- a series of content descriptions, populated through strands and sub-strands, that sets out the knowledge, understanding and skills that teachers are expected to teach and students are expected to learn
- an achievement standard that describes an expected level that the majority of students are achieving by the end of a given year of schooling. An achievement standard describes the quality of learning (e.g. the depth of conceptual understanding and the sophistication of skills) that would indicate the student is well placed to commence the learning required in the next year.

This exemplar

This Technologies exemplar articulates the content in the *Outline* and approaches to teaching, learning and assessment reflective of the Principles of Teaching, Learning and Assessment. This exemplar demonstrates a sequence of teaching and learning, including suggested assessment points, for 12 lessons.

Catering for diversity

This exemplar provides a suggested approach for the delivery of the curriculum and reflects the rationale, aims and content structure of the learning area. When planning the learning experiences, consideration has been given to ensuring that they are inclusive and can be used in, or adapted for, individual circumstances. It is the classroom teacher who is best placed to consider and respond to (accommodate) the diversity of their students. Reflecting on the learning experiences offered in this exemplar will enable teachers to make appropriate adjustments (where applicable) to better cater for students' gender, personal interests, achievement levels, socio-economic, cultural and language backgrounds, experiences and local area contexts.

Using this exemplar

This teaching, learning and assessment exemplar provides suggestions to support the delivery of the mandated curriculum content. The exemplar provides:

- a teaching and learning sequence
- the mandated curriculum content to be taught at each point of the teaching and learning sequence, suggested resources, sample assessment tasks and marking keys
- the number of lessons to deliver the teaching and learning experiences
- learning intentions and support notes that may provide focus questions and additional information and/or examples to assist with the interpretation of curriculum content
- support notes to assist teachers to unpack the content and support teaching and learning experiences
- teaching and learning experiences that outline the structure of the lesson. These explicitly state each activity that the lesson will progress through and the key focus area for that activity.

Links to electronic resources

This sequence of lessons may utilise electronic web-based resources, such as videos and image galleries. Teachers should be present while an electronic resource is in use and close links immediately after a resource, such as a video, has played to prevent default 'auto play' of additional videos. Where resources are referred for home study, they should be uploaded through Connect, or an equivalent system, that filters advertising content.

Best practice

Teaching and learning

The teaching and learning opportunities offered in this exemplar are not exhaustive. Thus, teachers are encouraged to make professional decisions about which learning experiences, and the sequence in which they are delivered, are best suited to their classroom context, taking into account the availability of resources and student ability.

This sample may prove a useful starting point for amplifying creativity in the classroom, while presenting the embedded expectations of the Western Australian Curriculum: Technologies.

Teachers may find opportunities to incorporate the General Capabilities and the Cross-curriculum Priorities into the teaching and learning program.

Ways of teaching – teachers can locate additional information on the Ways of teaching from the School Curriculum and Standards Authority (the Authority) website

<https://k10outline.scsa.wa.edu.au/home/wa-curriculum/learning-areas/technologies/digital-technologies/p-10-digital-technologies-teaching/digital-technologies-ways-of-teaching>.

Assessing

Assessment, both formative and summative, is an integral part of teaching and learning. Assessment should arise naturally out of the learning experiences provided to students. In addition, assessment should provide regular opportunities for teachers to reflect on student achievement and progress. As part of the support it provides for teachers, this exemplar includes suggested assessment points. It is the teacher's role to consider the contexts of their classroom and students, the range of assessments required, and the sampling of content descriptions selected to allow their students the opportunity to demonstrate achievement in relation to the year level achievement standard. Teachers are best placed to make decisions about whether the suggested assessment/s are used as formative or summative assessment and/or for moderation purposes.

Ways of assessing – a range of assessment strategies that can enable teachers to understand where students are in their learning is available on the Authority website

<https://k10outline.scsa.wa.edu.au/home/wa-curriculum/learning-areas/technologies/digital-technologies/p-10-digital-technologies-assessing/digital-technologies-ways-of-assessing>.

Reflecting

Reflective practice involves a cyclic process during which teachers continually review the effects of their teaching and make appropriate adjustments to their planning. The cycle involves planning, teaching, observing, reflecting and replanning.

This exemplar supports reflective practice and provides flexibility for teachers in their planning. The exemplar shows how content can be combined and revisited throughout the year. Teachers will choose to expand or contract the amount of time spent on developing the required understandings and skills according to their reflective processes and professional judgements about their students' evolving learning needs.

Python text adventure game

This exemplar can be used to further develop students' understanding and skills in computational thinking. Throughout the teaching and learning sequence, teachers will explicitly teach the process and production skills students require to complete the summative assessment task at the end of the sequence.

Students design and create a text adventure game based on ethical issues relating to the collection and ownership of data. The content of the game can include questions to prompt the following ideas:

- the importance of verifying the source and origin of data when conducting research, to assess the credibility and trustworthiness of data providers
- critical thinking and scepticism when encountering data from unfamiliar or dubious sources
- fact-checking methodologies and tools to cross-reference sources
- evaluating the authenticity of data collected.

Year level description

In the early adolescence phase of schooling, students align with their peer group and begin to question established conventions, practices and values. Learning and teaching programs assist students to develop a broader and more comprehensive understanding of the contexts of their lives and the world in which they live.

Digital Technologies further develops student understanding and skills in computational and design thinking, such as decomposing problems, and engages with a wider range of information systems as they broaden their experiences and involvement in local, regional, national and global activities.

In Year 8, students investigate the properties of networked systems and their suitability, based on performance specifications. They consider methods of data transmission and security in wired, wireless and mobile networks. They analyse and validate data using spreadsheets and make predictions by identifying trends. Students continue to investigate patterns, processes and phenomena, and explore forms of data representation and technology.

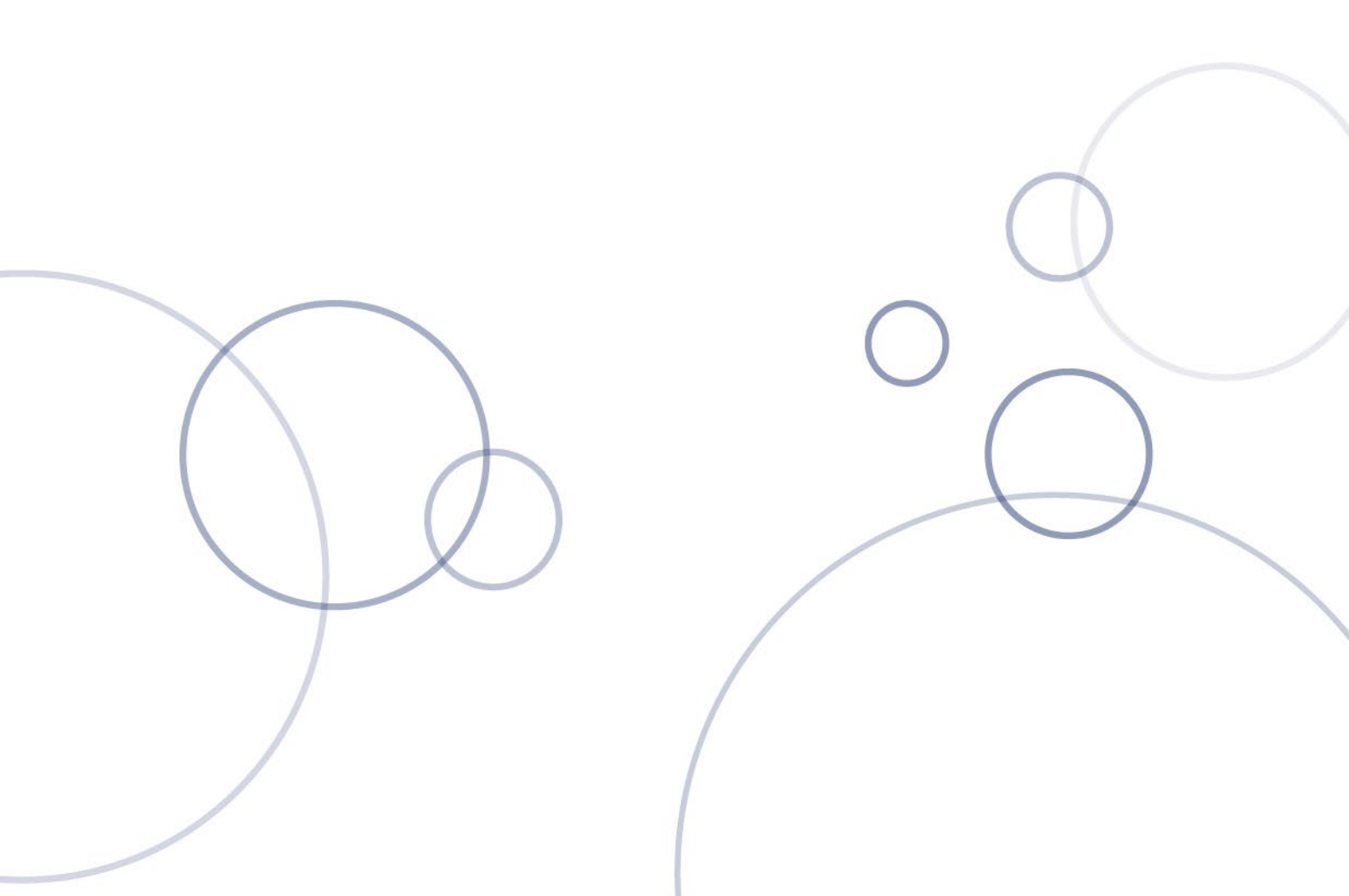
When defining problems, students identify the key elements of the problems and the factors and constraints at play. They design increasingly complex algorithms. Students develop an understanding of ethical issues regarding the collection and ownership of data.

Achievement standard

By the end of the year:

Students describe methods of data transmission and security in wired, wireless and mobile networks, identify specifications of hardware components and outline potential impact on particular tasks. They identify how digital systems represent image and audio data using binary. Students analyse and validate data, using spreadsheets, to draw conclusions and make predictions based on identified trends. They design algorithms involving nested control structures, represented by flow charts and pseudocode, in plain English. Students implement, modify and debug programs involving control structures in a general-purpose programming language. They trace algorithms to predict output for a given input and identify errors. Students identify ethical issues regarding the collection and ownership of data and evaluate the authenticity, accuracy and timeliness of acquired data. They describe cybersecurity threats, including phishing.

In Digital Technologies, students investigate a given need or opportunity for a specific purpose and develop a design brief. Students consider and select components and resources to develop solutions, identifying constraints. They use appropriate technical terms and technologies to design, develop, evaluate and communicate alternative digital solutions. Students use project management skills to plan, develop and communicate, whilst considering time, resources and costs to achieve solutions. They develop contextual criteria to evaluate design processes and solutions.



Lessons 1–12

Python text adventure game

Lesson 1: Fake news

The Western Australian Curriculum content addressed in this lesson is below.

Acquiring, managing and analysing data

- Evaluate the authenticity, accuracy and timeliness of acquired data

Privacy and security

- Ethical issues relating to the collection and ownership of data
-

Lesson outline

Learning intentions

Students will:

- understand and explain the importance of verifying the source and origin of data when conducting research, to assess the credibility and trustworthiness of data providers
- use critical thinking and scepticism when encountering data from unfamiliar or dubious sources
- learn and apply fact-checking methodologies and tools to cross-reference sources
- evaluate the authenticity of data collected.

Preparation for the lesson

- Select a newspaper or magazine article that clearly shows the use of fake news strategies, such as hyperbole or unproven gossip.
- Source several online news articles that range from authentic to suspicious.
- Prepare a video to show students that explains the concept of fake news. For example:
ABC Behind the News – How to Spot Fake News – BTN Media Literacy
https://www.youtube.com/watch?v=X_yj1UMQEac.

Support notes

- Fake news is information created to mislead its audience.
- Fake news stories sometimes use false headlines to get your attention.
- Fake news can be in a variety of forms, like propaganda or gossip.
- Fact-checking websites, such as Snopes and RMIT ABC Fact Check, can be used as tools for checking the credibility and trustworthiness of information.

Teaching and learning experience

Introduction

- Explain to students that fake news as a concept existed before the invention of computers. Newspapers in the 1890s used the technique of sensationalism in news stories to sell more papers. This gained the attention of news readers without always being completely truthful.
- Show the class an example of fake news and ask them to identify the headline and subheadings that they find interesting or exciting quotes or information from the article.
- With the introduction of the internet and social media, the availability of news – including fake news – became widespread. The wealth of information online makes it difficult to pick fact from fiction.

- Play the prepared ABC Behind the News video.
- Ask students to consider their own habits when seeking the news:
 - What kinds of news do you look at?
 - Do you use the TV, radio, print or online media to seek the news?
 - Where do you get your news?

Main activity

- Display the prepared online news articles that range from authentic to suspicious.
- Students move into groups and are assigned one article to evaluate. Each group completes the table below.

Authorship Is it from a trustworthy source that is known for being honest about information?	
Timeliness Is the information up to date?	
Content Does the headline match the content? Are different views included, to give the content balance? Does the content seem to be unbelievable or 'too good to be true'? Is there enough evidence and reasoning in the story to justify the claims or conclusion?	

- Groups report on their findings at the conclusion of the group discussions.

Lesson conclusion

- Students compose a headline or short story that could be real or fake.
- Students add their headlines or short stories to an online class discussion board.
- Other students assess whether the headline is true or false, adding structured comments to the post and demonstrating good digital citizenship. One commenting structure to consider is the 'feedback sandwich', which involves including a point of negative feedback between two positive aspects.

Lesson 2: Algorithms and programs in Python

The Western Australian Curriculum content addressed in this lesson is below.

Digital implementation

- Design algorithms involving nested control structures and represent them using flow charts and pseudocode

Lesson outline

Learning intentions

Students will:

- understand algorithms and programs, and how they are executed
- make the transition from visual/block code to Python® syntax
- write and execute a program using Python.

Preparation for the lesson

- Ensure the students and you have access to a Python integrated development environment (IDE).
- Create a slide show of worked examples of visual/block code versus Python code.
- Create a work sheet with practice tasks that cover the following concepts: variables, values and an assignment. The work sheet should include examples with syntax errors to be identified and corrected.
- Develop a Python cheat sheet of common commands, to assist students to work through the activities.

Support notes

- An algorithm is a sequence of iterative steps that are used to solve a problem. It includes a set of precise instructions expressed in a programming language.
- A program is a set of precise instructions expressed in a programming language. Translating the programming language is necessary for a computer to execute the instructions.
- The lesson is a transition from visual/block code to Python, which introduces the concept of syntax and errors. The transition from visual/block code to written code can cause frustration in novice users.
- It is important for you to:
 - be familiar with a Python IDE
 - understand input, output and assignment in Python
 - be comfortable with identifying and fixing common syntax errors.

Teaching and learning experience

Introduction

- Using the *Think-pair-share* strategy, ask students to identify what they already know about algorithms. Ask them to recall examples from prior learning. Examples may include the sequence of steps used for making a hot chocolate, building with connecting blocks or deciding what method of transport to take to school based on the weather.

Main activities

Activity 1

- Use the prepared slide show with examples to explain the difference between algorithms and programs. Provide a definition for 'algorithm' and 'program'. Focus on the main difference between algorithms and programs, which is that programs are algorithms expressed following the strict rules of a programming language. This is essential, because programs will be executed by machines.
- Explain that a Python program needs to be run through a Python interpreter that will read, translate and execute the program.
- Introduce students to syntax errors, explaining that just like natural languages have syntax rules, so does coding. Briefly discuss the rules of syntax.
- Explain that visual/block code is a good way to learn coding since it allows students to build programs using graphical blocks rather than syntax, which contains syntax errors. In text-based programming, syntax errors are common and easy to make.
- Use the two prepared slides, one with a worked example of Python code with a syntax error and one without, to conduct a spot-the-difference activity as a class.
- Remind students to adopt a growth mindset and to not be discouraged by syntax errors.
- Use a Python IDE to demonstrate the traditional 'Hello world' program in Python and explain that this is what students will complete first.
- Explain the IDE features, then display a few common syntax errors and demonstrate how to find the errors, read error messages and correct them. Students work in pairs to follow the teacher's instructions.

Activity 2

- Students work in pairs using driver/passenger roles to work through the tasks on their work sheet; they should have the Python cheat sheet available for reference.
- Have the students swap roles after each task.
- After the activity, use the slide show to provide solutions to each task. Discuss their solutions as a class or in small groups.

Lesson conclusion

- Ask students to identify code snippets responsible for output, input and assignment. Provide an example on the slide show.

Lesson 3: Making predictions using a sequence

The Western Australian Curriculum content addressed in this lesson is below.

Digital implementation

- Design algorithms involving nested control structures and represent them using flow charts and pseudocode

Lesson outline

Learning intentions

Students will:

- use arithmetic expressions to calculate values
- describe assignment statements
- use variables to store and reference values
- receive input from the keyboard and convert it to a numerical value.

Preparation for the lesson

Note: An extra lesson may be required if you are not confident in the students' flowcharting skills.

- Ensure the students and you have access to a Python integrated development environment (IDE).
- Create a slide show of worked examples of visual/block code versus Python code. Also include an explanation of how an assignment works, using an arithmetic example.
- Familiarise yourself with flow charts and structured English. Place examples of these alongside code examples.
- Create a work sheet with practice tasks that cover the following concepts: assignment statements, assignments with expressions and using numbers as input.
- Develop a Python cheat sheet on arithmetic operators to help students work through the activities.
- Develop a quick, formative multiple-choice quiz for the lesson conclusion about the programming concepts covered in the lesson.

Support notes

- A sequence is the simplest type of control structure. It is just one instruction followed by the next, in logical order.
- Assignments are instructions to be executed; they are not equations. The assignments are read from right to left. A subsequent assignment in a program can assign a new value, replacing the previous value.
- An expression can be used to assign a value to a variable.
- A string is a piece of text or more than one character placed together.
- The `int()` function needs to be used to convert a string to a number that can be used for arithmetic.
- You will need to be comfortable identifying and fixing common syntax errors.

Teaching and learning experience

Introduction

- Use multiple-choice questions on three coding examples to recall concepts covered in Lesson 2. Ask students to make predictions using a *Think-pair-share* strategy. Examples could involve finding the syntax error, assigning a variable or guessing the output of print statements. Correct students' misunderstandings.
- Use the slide show to explain how an assignment works, showing an arithmetic example. Explain that the assignment is not an equation; it is an instruction to be executed. A new assignment can assign a new value that replaces the previous value.

For example:

```
pin = 3651
print(pin, 'is my secret pin number')
```

- Provide another example of assignment, this time with an expression.

For example:

```
pin = 3000 + 600 + 50 + 1
print(pin, 'is my secret pin number')
```

- Provide students with a Python cheat sheet that lists arithmetic operators.
- Explain that during program execution, a variable must have been assigned a value before that value is referred to. Walk students through a live example.

Main activity

- Provide students with the code for a program to do a simple conversion. For example, convert bits to bytes in the wrong order.
- Students work in pair programming. Ask students to rearrange the order of the sequence to run without errors.
- Provide instructions for writing a program to calculate students' age from their year of birth.
- The students will experience a type error, because the number entered for their year of birth will be treated as a string (i.e. a piece of text).

For example:

```
print('What year were you born?')
birth_year = input()
age = 2026 - birth_year
print('You are', age, 'years old')
```

- The expression needs to be passed through the `int()` function to convert it to a number.

For example:

```
print('What year were you born?')
birth_year = int(input())
age = 2026 - birth_year
print('You are', age, 'years old')
```

- Students continue writing programs that receive and process numerical inputs from the keyboard. For example, converting miles to kilometres.

Lesson conclusion

- At the end of the lesson, use a multiple-choice quiz on the concepts covered to check students' understanding. You may choose to facilitate the quiz online to get instantaneous feedback.

Lesson 4: Selection/decisions

The Western Australian Curriculum content addressed in this lesson is below.

Digital implementation

- Design algorithms involving nested control structures and represent them using flow charts and pseudocode
- Implement, modify and debug programs involving control structures in a general-purpose programming language

Lesson outline

Learning intentions

Students will:

- use selection (if statements) to control the flow of a program execution between decisions
- use relational operators in Python.

Preparation for the lesson

- Ensure the students and you have access to a Python integrated development environment (IDE).
- Create a slide show of worked examples of visual/block code versus Python code. Include a photograph in the slide show that shows crossroads or a matrix style of formatting. Place an example of visual programming in the form of block code that uses decision (if/else). Place the flow chart and structured English next to the example.
- Familiarise yourself with flow charts and structured English. Place examples of these alongside code examples.
- Create a work sheet with practice tasks to cover the following concepts: if statements and relational operators to compare values of expressions. Both strings and numbers should be used in the work sheet questions.
- Develop a checklist to help students spot common syntax errors.
- Develop a Python cheat sheet on relational operators, to help students work through the activities.

Support notes

- Selection is a control structure that involves checking conditions and selecting which path (program) to follow. Selection is necessary in a program whenever the program must execute different actions in different circumstances to achieve the intended output.
- You will need to be comfortable with:
 - identifying and fixing common syntax errors
 - the use of conditions
 - the use of selection.

Teaching and learning experience

Introduction

- Play the Python code slide show.
- Explain to students that tracing a program is the process of understanding the sequence line by line, placing different inputs and identifying the program's process-intended outputs.
- Ask the whole class to trace through the program, and explain that they will learn about selection in Python this lesson.
- Trace the flow chart with your finger to show the possible paths the program may take during execution.
- Explain to students that an 'if' statement must be used when there is more than one possible path for a program to follow. Demonstrate this through the flow chart, tracing the flow of data in a selection example, referring to the structured English to the side.
- Through a teacher-led live coding session, show the 'if' statement in Python and explain when it is necessary. Explain the syntax of the 'if' statement in Python (no capitalisation, the colon, indentation, the = operator).
- Print a checklist to help students spot common syntax errors.

Main activity

- Students use pair programming to work through a series of programs that use selection. Examples include printing a statement that asks for a favourite colour or a number guessing program. Remind students to swap driver/passenger roles after each task.
- Walk through the number guessing game with students in sequence.

Lesson conclusion

- Show students a series of selection programs with errors. Ask the class to predict the output and correct the errors.

Lesson 5: Iteration and repetition – Part 1

The Western Australian Curriculum content addressed in this lesson is below.

Digital implementation

- Design algorithms involving nested control structures and represent them using flow charts and pseudocode
- Implement, modify and debug programs involving control structures in a general-purpose programming language

Lesson outline

Learning intentions

Students will:

- use multi-branch selection ('if', 'elif', 'else' statements) to control the flow of program execution
- describe how iteration ('while' statements) controls the flow of program execution.

Preparation for the lesson

- Ensure the students and you have access to a Python integrated development environment (IDE).
- Create a slide show of worked examples of visual/block code versus Python code.
- Familiarise yourself with flow charts and structured English. Place examples of these alongside code examples on a worksheet or in a slide show.
- Create a work sheet with practice tasks to cover the following concepts: multi-branch selection (if, elif, else) and iteration (while).
- Create a work sheet with an example of worked code that contains errors and the Python code to test it.
- Develop a Python cheat sheet on relational operators, to assist students to work through the activities.

Support notes

- Multi-branch selection (if, elif, else) checks successive conditions and selects one out of multiple branches to follow.
- It is required to use an 'if' block with an 'elif' block when there are more than two paths to follow. You can use multiple 'elif' blocks. You can nest anything inside the blocks of statements, including more 'if' statements. 'elif' is short for 'else if' and is used in Python programming.
- Iteration is when a program repeats actions that check for a terminating condition at the beginning of each new type, including for, while, loops, etc.
- You will need to be comfortable with the use of selection and iteration in Python programs.

Teaching and learning experience

Introduction

- Give students the work sheet with code that contains errors and the Python code to test it. Ask students to refer to the checklist of errors from Lesson 4 and then correct the errors in the Python code, before executing the program.
- Use an example of an 'if' and 'else' selection program to demonstrate how the branches could be extended to more than two options. For example, the program might be applied in the context of a robot at an intersection deciding to move forward, turn left or turn right. Include this in the slide show. Show the flow chart and structured English next to the program.

Main activities

Activity 1

- Students use pair programming to work through a series of programs that use multi-branch selection. End with a number guessing program.

Activity 2

- Referencing the number guessing program, pose the question: What if we wanted to repeat the process until the correct number is guessed?
- Mention that many programs have sections that could be repeated. Explain that iteration is when programs repeat actions and check for a terminating condition at the beginning of each new loop.
- Display the 'repeat until' and 'forever' blocks to activate prior learning on visual/block code.
- Using a flow chart, show that a 'while' statement is needed to repeat actions while a condition is being met.
- Show an example of a flow chart with structured English. Students predict what will happen. Go through each line of code step by step, to check for understanding.

Lesson conclusion

- Use either a digital or physical exit ticket to end the lesson by asking students to identify the number of options in different programming situations.

Lesson 6: Iteration and repetition – Part 2

The Western Australian Curriculum content addressed in this lesson is below.

Digital implementation

- Design algorithms involving nested control structures and represent them using flow charts and pseudocode
- Implement, modify and debug programs involving control structures in a general-purpose programming language

Lesson outline

Learning intentions

Students will:

- use iteration ('while' loops) to control the flow of a program
- use variables as counters in iterative programs
- introduce counters with trace tables.

Preparation for the lesson

- Ensure the students have access to a Python integrated development environment (IDE).
- Create a slide show of worked examples of visual/block code versus Python code.
- Familiarise yourself with flow charts and structured English. Place examples of these alongside code examples in the slide show.
- Create a scaffolding work sheet with instructions, a syntax error checklist and tips to support students' consolidation of concepts. This work sheet should include trace table examples.
- Create a work sheet with instructions to modify the count program starting from 10.

Support notes

- A trace table is used to test for logical errors in algorithms or programs. Modelling tracing techniques is embedded through worked examples to check for errors and reinforce students' understanding of concepts.
- Counters are the simplest form of variables used to compute results iteratively, with each new value accumulated over the previous ones.
- Import the time module to call on the sleep function and slow down the counting. Import the sleep module at the start of the program using the `time import sleep` command.

Teaching and learning experience

Introduction

- Display a program on the board to check if students understand assignments and variables. Students do a *Think-pair-share* activity to predict the output. Provide multiple-choice answers. For example:

```
count = 3
print (count)
count= count -1
print (count)
```
- Trace through each line of the trace table on a new slide. Include questions to check students' understanding. Model the use of trace tables to make predictions in the worked example.

Line	Count	Output
1	3	
2		3
3	2	
4		2

- Extend the example used at the start of the lesson by showing repetition. Students use a work sheet with a trace table to predict the outputs. For example:

```
count = 3
print (count)
count= count -1
print (count)
count= count -1
print (count)
count= count -1
```
- Walk students through the example, filling in the trace table as you go.

Line	Count	Output
1	3	
2		3
3	2	
4		2
5	1	
6		1
7	0	

- Explain how the program seems to be repeating the same actions over and over again. Students use the work sheet to fill in the missing condition from a range of options.

For example:

```
count = 3
while _____
    print(count)
    count = count -1
```

Solution:

```
count = 3
while count >= 1
    print(count)
    count = count -1
```

Main activity

- Students program in pairs using a work sheet with instructions to modify the count program so that it starts from 10 instead of three. Students then write a statement for a game that includes a countdown. For example, 'Lift-off!' could be displayed for a space game when the countdown reaches zero.
- Modify the program so that it starts from one and counts in threes until it exceeds 20.
- Import the time module to call on the sleep functions so that the program waits a given number of seconds and slows down the count.

The two lines of code required are:

```
from time import sleep
sleep (1)
```

Lesson conclusion

- Explain to students that their assessment task will involve designing a Python text adventure game on the topic of fake news. You may ask students to research retro, text-based adventure games and to brainstorm ideas for their game outside class time.

Lesson 7: Password guessing game – formative assessment

The Western Australian Curriculum content addressed in this lesson is below.

Digital implementation

- Design algorithms involving nested control structures and represent them using flow charts and pseudocode
- Implement, modify and debug programs involving control structures in a general-purpose programming language

Lesson outline

Learning intentions

Students will:

- demonstrate their knowledge of digital implementation by presenting design plans (in a diagram and in English) for a password guessing game
- predict outputs and document any errors and modifications made
- use a Python IDE to produce a password guessing mini game (Appendix B).

Preparation for the lesson

- Ensure the students and you have access to a Python integrated development environment (IDE).
- Create a slide show of the worked examples of the number guessing game in flow charts and in structured English.
- Familiarise yourself with flow charts and structured English. Place examples of these alongside code examples in the slide show as above.
- Create a work sheet with practice tasks to cover the following concepts: assignment statements, assignments with expressions and using numbers as input.
- Develop a Python cheat sheet on all concepts covered, to assist students to work through the activities.
- Refer to Appendix B for the formative assessment covered in this lesson.

Teaching and learning experience

Introduction

- Provide students with the Python cheat sheet and sample code for the number guessing game.
For example:

```
number = 7  
print('You guessed:', number)
```

Main activity

- Students brainstorm how the sample code could be improved using the concepts covered. Guide the discussion, making sure the concept of user input is raised. Leave the list of improvements on the board.

- Demonstrate how to modify the program and make a prediction.
For example:

```
number = input ('Guess the number:')  
print ('You guessed:' number)
```
- Students work in pairs or small groups to extend the number guessing game into a password guessing mini game, linking it to the topics of privacy and security. Provide a work sheet with the full number guessing game worked example (Appendix B), including a flow chart and structured English. Provide feedback on student learning. See Appendix B for more information on using this activity as formative assessment.

Lesson conclusion

- The student groups compare the feedback they received and play each other's password guessing games. Students discuss how they could improve their mini games and search the internet for their text-based adventure games.

Lesson 8: User experience design plans

The Western Australian Curriculum content addressed in this lesson is below.

Digital implementation

- Design the user experience (UX) of a digital system

Design thinking skills

Designing

- Design processes and solutions considering a range of technologies and techniques, using appropriate technical terms

Evaluating

- Use student-developed contextual criteria to evaluate design processes and solutions

Lesson outline

Learning intention
Students will: • identify and use design tools to begin planning for the summative assessment task.

Preparation for the lesson

- Provide examples of text-based adventure games using screenshots or short videos of game play. Examples of American Standard Code for Information Interchange (ASCII) art may be included as well as examples of mini games, with design planning notes, such as brainstorm, flow charts, and storyboards, including annotations.
- Prepare an example game question with selection along with the flow chart and structured English.

Support notes

- A text-based adventure game is a computer game that uses a text-based interface instead of raster or vector graphics.
- The most basic text-based adventure games are built around 'if', 'elif' and 'else' statements, with pathways leading to the next set of user input options or a 'game over' screen.
- A user interface (UI) is the way users and machines interact through a graphical user interface (GUI). There are two main UI types in computing: command line interfaces (CLIs) and GUIs, i.e. the visual appearance of the program.
- User experience (UX) design is the way in which interaction is designed. Design planning tools can include wire frames, storyboards and flow charts.
- Refer to Appendix C for more information on the Python text adventure game summative assessment.

Teaching and learning experience

Introduction

- Place students in small groups, then use the *Graffiti* cooperative learning strategy to recall the concepts covered in Lesson 1 of this learning sequence such as Privacy and security content. Collate the main concepts on the board, leaving this up for the remainder of the lesson.
- Explain what is meant by user experience design in relation to text-based adventure games, using screenshots and short videos of gameplay as examples. Students identify characteristics of the games, such as the start screen, types of questions, amount of text, animation of text, 'game over' screen etc. Show examples of design tools used to make games like these, such as flow charts and storyboards.
- Walk students through a worked example of a game question, showing students the associated flow chart and structured English.

Main activity

- Explain the purpose of the summative task in Appendix C.
- Students form pairs or small groups to begin brainstorming their games for the summative assessment in Appendix C.

Lesson conclusion

- Students present a one-minute pitch on the main idea for their game.

Lessons 9–12: Python text adventure game – summative assessment

See Appendix C for syllabus links.

Learning intentions and support notes

- Students complete the Python text adventure game task to the best of their ability.

Preparation for the lessons

- Familiarise yourself with the Python text adventure game task in Appendix C.
- Students should be familiar with the content and skills required to complete the task to the best of their ability.

Teaching and learning experience

Introduction

- Explain to students each element of the task. This task can be broken into sections, with set milestones for student planning and feedback.

Main activity

- Student groups recap their knowledge and understanding by showing another group the start of their design planning for the task from Lesson 8.
- Students complete the Python text adventure game task (Appendix C).

Lesson conclusion

- On the completion of this task, students play each other's games. They can do this in the style of a game showcase.

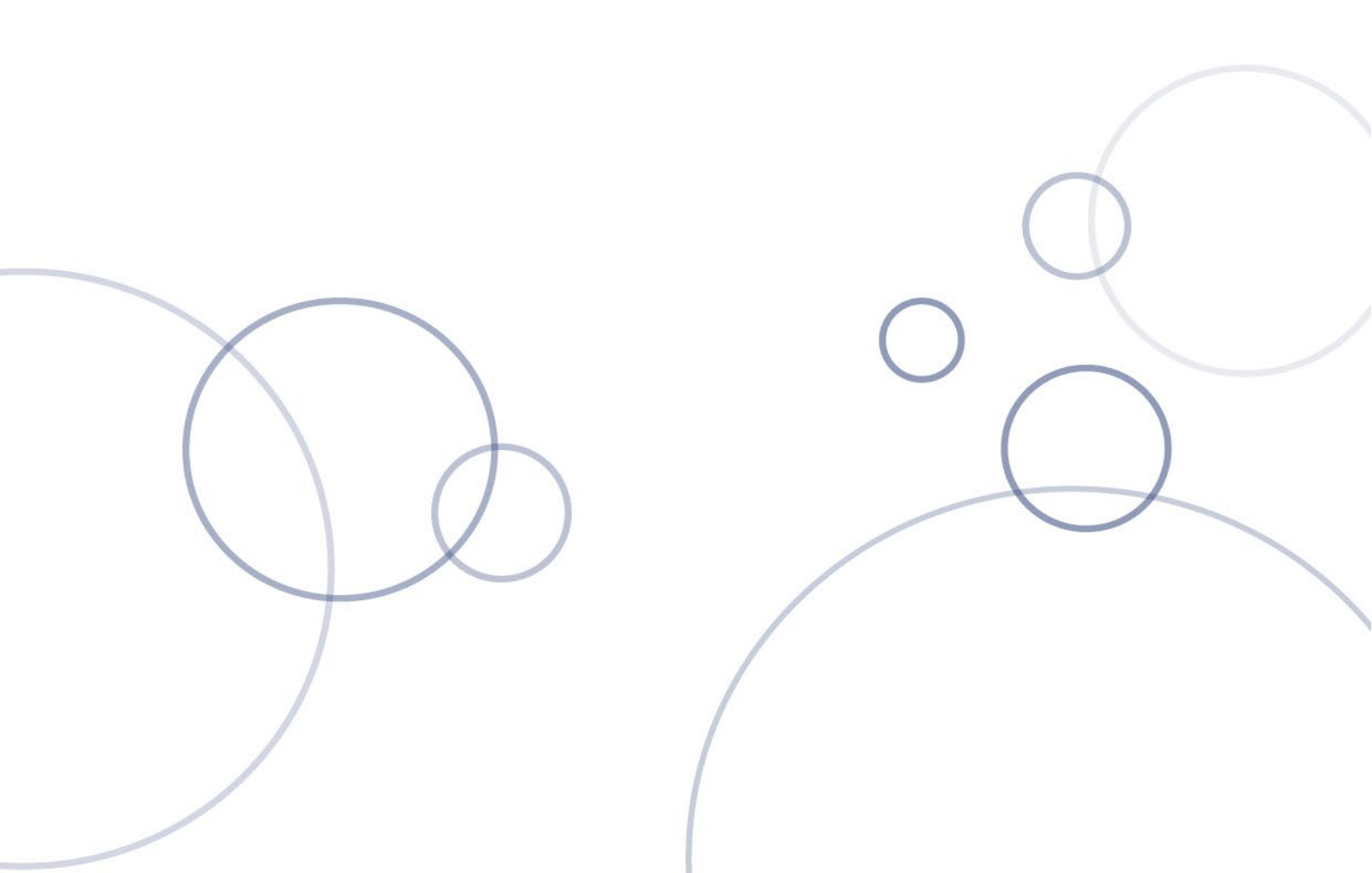


Appendix A

Resources

Resources

Lesson	Resource	Link/information
1	PBS Learning Media	PBS Learning Media. <i>Why Is Fake News So Effective?</i> https://www.pbslearningmedia.org/resource/ilcpbn18-ela-ilfakenews/why-is-fake-news-so-effective/
1	BTN Media Literacy	ABC Behind the News. <i>How to Spot Fake News – BTN Media Literacy</i> https://www.youtube.com/watch?v=X_yj1UMQEac .



Appendix B

Formative assessment task

Password guessing mini game

Task details

Title	Password guessing mini game
Description	Students work in pairs or small groups and use their knowledge and understanding to extend the number guessing game.
Type of assessment	Formative
Purpose of assessment	To assess students' ability to predict and design algorithms using flow charts and structured English. To assess their ability to correctly use technical terminology and apply user interfaces within a programming environment, including the need for a choice of options and/or repeating options
Evidence to be collected	Observations of students collaborating and managing Flow chart Structured English/pseudocode Python program
Suggested time	1–2 one-hour lessons (depending on class size)
Differentiation	Teachers should differentiate their teaching and assessment to meet the specific learning needs of their students, based on their level of readiness to learn and their need to be challenged. Where appropriate, teachers may either scaffold or extend the scope of the assessment task.

Content descriptions

Digital implementation

- Design algorithms involving nested control structures and represent them using flow charts and pseudocode
- Trace algorithms to predict output for a given input and to identify and fix errors
- Implement, modify and debug programs involving control structures in a general-purpose programming language

Design thinking skills

Project management

- Plan, develop and communicate, using project management processes, considering time, resources and costs to achieve solutions

Designing

- Design processes and solutions considering a range of technologies and techniques, using appropriate technical terms

Resources

- Access to a Python integrated development environment (IDE)
- Design documentation with flow chart and algorithm
- Marking key

Instructions for teachers

Separate students into groups of up to three students. Provide these groups with the number guessing game sample program (included below).

```
# Set secret number
number_secret=43

# Display welcome message and instructions
print("** Welcome to the Number Guessing Game **")
print()
print("Enter a number between 1 and 100")
print("You have 5 attempts")
print()

# User to guess the number
for i in range(5):
    number_guess=int(input("Enter a guess: "))
    if number_guess==number_secret:
        print("You won!")
        break
    else:
        print("Not a match, try again")

# Sign off message
print()
print("Game over")
```

Allow students enough time to plan their modifications to the game, including time for brainstorming and creating a flow chart, as well as writing down annotations and references. Use oral questioning techniques to give feedback on student progress.

Consider the class's abilities and need for support or extension when implementing this task.

Instructions to students

Work collaboratively to manage your time and resources to document your planning, modifications and annotations to convert the number guessing game into a password guessing mini game.

You are required to:

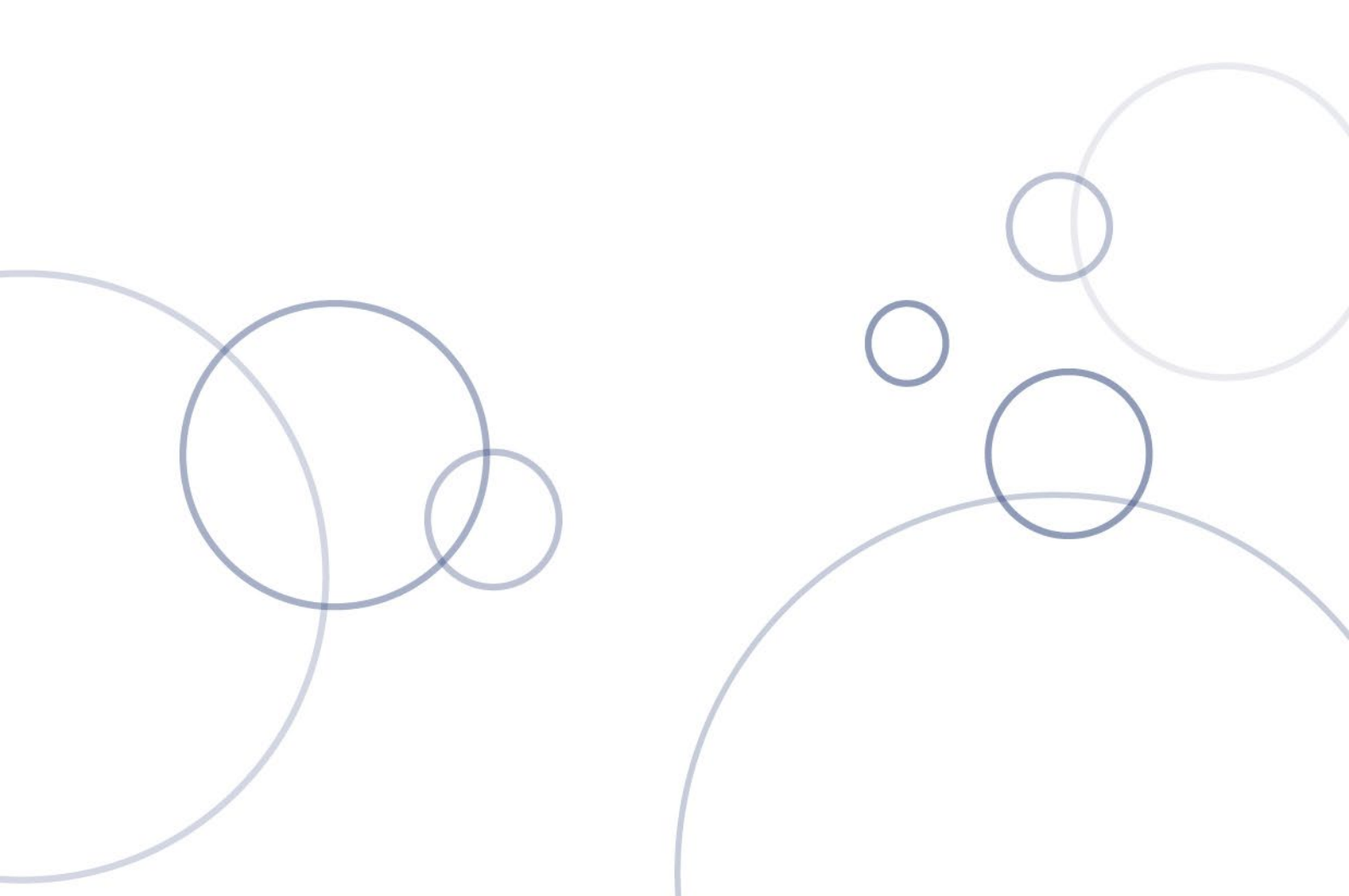
- predict errors in the guessing game using a trace table
- modify the guessing game provided
- use your guessing game to create a flow chart and algorithm. You may choose to use the sample provided by the teacher.

Familiarise yourself with the marking key.

Work together in groups to achieve the best possible outcome.

Marking key

Description	Marks
Digital implementation – modification of number guessing game	
Proficiently predicts output for a given input to identify errors Modifies and effectively implements new sequences	4
Predicts output for a given input to identify errors Modifies and implements a new sequence	3
Predicts output for a given input to identify errors Modifies and implements a sequence	2
Modifies and implements digital solutions (with errors), including an incomplete or inappropriate sequence within a programming environment	1
Subtotal	/4
Designing – algorithm	
Produces a simple plan that problem-solves through a detailed and logical sequence of steps	4
Produces a simple plan that problem-solves through a logical sequence of steps	3
Produces a simple plan that problem-solves through a sequence of steps	2
Produces a simple plan designed to solve a problem, using broad steps	1
Subtotal	/4
Designing – flow chart	
Consistently and correctly identifies, and effectively implements, two-way or multi-way decisions (branching)	4
Consistently and correctly identifies two-way or multi-way decisions (branching)	3
Correctly identifies two-way or multi-way decisions (branching)	2
Creates an incomplete sequence of steps that attempt to solve a problem	1
Subtotal	/4
Project management	
Consistently plans, develops and communicates, using effective project management processes	4
Plans, develops and communicates, using effective project management processes	3
Plans, develops and communicates, using project management processes	2
Sometimes plans, develops and communicates simple ideas when managing projects	1
Subtotal	/4
Total	/16



Appendix C

Summative assessment task

Python text adventure game

Task details

Title	Python text adventure game
Description	Students design and create a text-based adventure game covering the topics of data representation, privacy and security.
Type of assessment	Summative
Purpose of assessment	To assess students' ability to predict and design algorithms using flow charts and structured English To assess students' ability to correctly use technical terminology and to apply user interfaces within a programming environment, including the need for a choice of options and/or repeating options
Evidence to be collected	Planning documentation – flow chart and algorithm Python program Bibliography Self-evaluation
Suggested time	Four one-hour lessons
Differentiation	Teachers should differentiate their teaching and assessment to meet the specific learning needs of their students, based on their level of readiness to learn and their need to be challenged. Where appropriate, teachers may either scaffold or extend the scope of the assessment task.

Content descriptions

Digital implementation

- Design algorithms involving nested control structures and represent them using flow charts and pseudocode
- Trace algorithms to predict output for a given input and to identify and fix errors
- Implement, modify and debug programs involving control structures in a general-purpose programming language

Design thinking skills

Project management

- Plan, develop and communicate, using project management processes, considering time, resources and costs to achieve solutions

Investigating and defining

- Investigate a problem for a given need or opportunity
- Consider technologies, resources and/or components to develop solutions, identifying constraints

Designing

- Design processes and solutions considering a range of technologies and techniques, using appropriate technical terms

Producing and implementing

- Implement agreed protocols, a range of technologies, techniques, components and processes to produce designed solutions

Evaluating

- Use student-developed contextual criteria to assess design processes and solutions

Task preparation

Prior learning

Students have previously completed Lessons 1 to 7.

Students begin brainstorming their games for the summative assessment in Lesson 8.

Assessment task

Assessment conditions

Students may work individually or collaboratively to complete the task.

Resources

- Access to a Python integrated development environment (IDE)
- Design documentation, including flow chart and algorithm
- Design booklet for Python text adventure game task
- Self-evaluation sheet

Instructions for teachers

Students create a text-based adventure game using Python. The concept of the game should aim to increase awareness of fake news or other Privacy and security content covered in class. The game should meet the following criteria:

- have a minimum of five questions
- include at least one repeated option
- have a start menu with instructions on how to play the game
- have a 'game over' screen and post-game credits
- the player must be given feedback after typing a response
- more advanced students may create multiple game endings.

Students must provide a bibliography of all sources used as research for the content of the game.

Once students have produced their text-based adventure game, other students should test the game and provide peer feedback.

Note: At this level, students need to independently develop their own criteria for evaluating products.

Instructions to students

Task description: Python text adventure game

Design, produce and evaluate a text-based adventure game programmed using Python that aims to increase awareness of fake news.

The game must be designed to cover all three of the following topics:

- how to identify suspicious information online
- what are the ways that we can check if information is authentic or not
- what actions we must take when discovering that information is not authentic.

Your game must have a start menu with instructions on how to play the game, plus a 'game over' screen and post-game credits. There must be a minimum of five in-game questions with a series of responses that the player can select from. When the player types a response, they should receive immediate feedback. For example:

- Well done – you cracked the code! Proceed to the next room.
- Incorrect. Try again.
- You failed to crack the code. Game over.

When planning and producing your game, make sure that each response generates the correct feedback for the player and that players can proceed through the game to complete it.

Ensure that all your concepts and planning, including any sketches or diagrams, are recorded in a digital document.

Once the game development stage is over, other students will have the opportunity to play your game and provide feedback. You will also play other students' games and provide feedback; this will help you complete your self-evaluation.

At the end of the task, you will need to submit the following for assessment:

- planning documentation
- Python program
- bibliography
- self-evaluation.

Investigating and defining

If I am in a group, I am working with: _____

Lesson 1	Lesson 2	Lesson 3	Lesson 4
Tasks:	Tasks:	Tasks:	Tasks:



Designing – storyboard

Your game must include a start menu, a minimum of five in-game questions, a 'game over' screen and post-game credits. Plan your game using the storyboard below.

List of sources or game references used while researching:

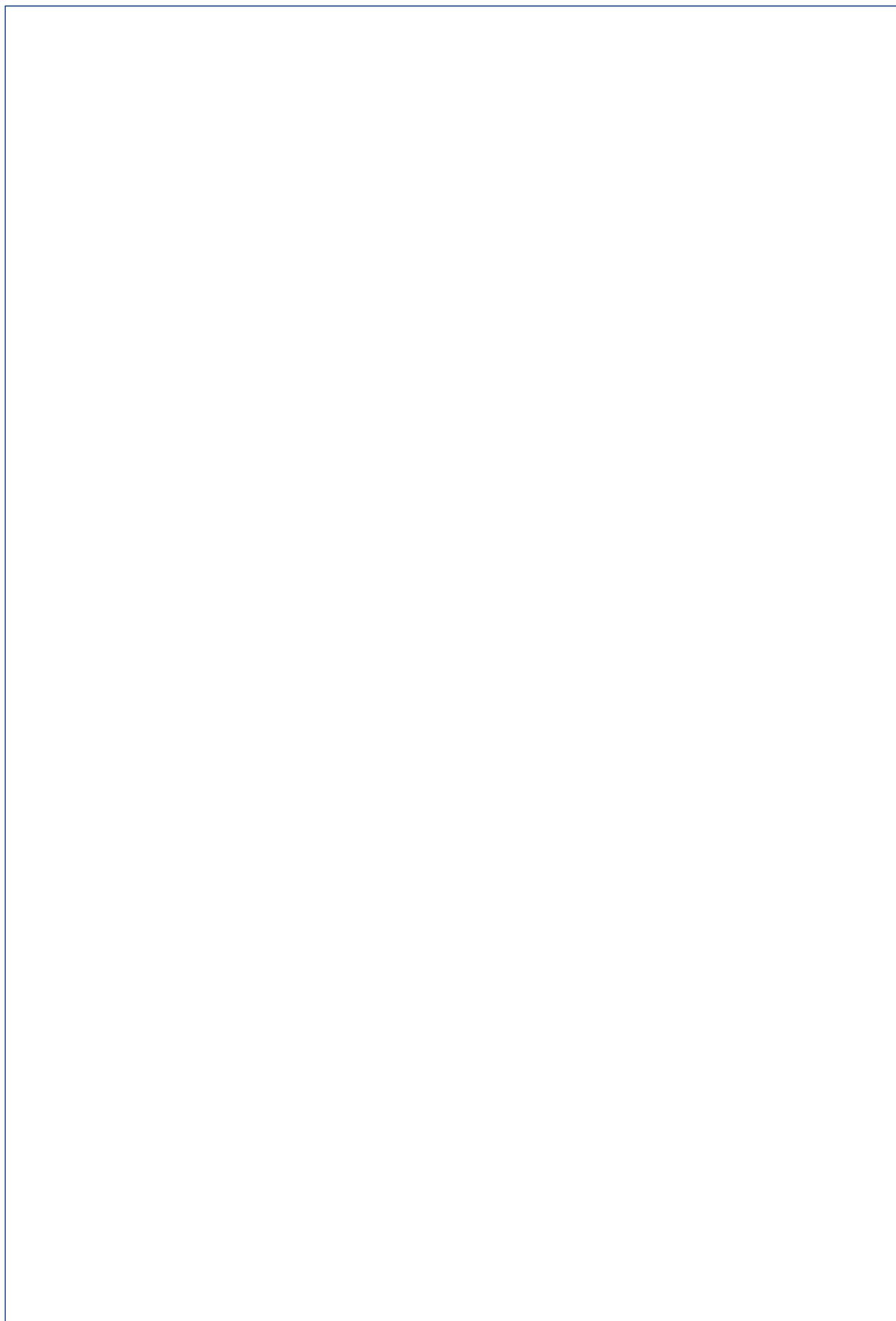
Designing – flow chart

Use the following symbols to create a flow chart to represent how your game works. Note that each symbol has a particular meaning/use. Write any additional annotations next to your flow chart.

Flow chart symbols

	Start or end (terminator)
	Process – used to represent any process, task or activity that takes place.
	Input/output – used to represent information that the user needs to input or information that the program will output.
	Decision – used when there is more than one alternative pathway, e.g. true/false, yes/no.
	Arrow – used to connect other elements of the flow chart, show the flow of data or indicate repeat structures.

Draw your flow chart in the box below.

A large, empty rectangular box with a thin blue border, intended for drawing a flow chart. The box occupies the majority of the page below the instruction.

Marking key

Description	Marks
Investigating and defining	
Comprehensively investigates and defines the target audience, giving consideration to all components to achieve an appropriate solution	4
Investigates and defines in detail the target audience, giving consideration to all components to achieve an appropriate solution	3
Investigates and defines the target audience, giving consideration to some components to achieve a solution	2
Gives little consideration to achieve a solution for a target audience	1
Subtotal	/4
Designing – storyboard	
Produces a simple plan that problem-solves through a detailed and logical sequence of steps	4
Produces a simple plan that problem-solves through a logical sequence of steps	3
Produces a simple plan that problem-solves through a sequence of steps	2
Produces a simple plan designed to solve a problem, using broad steps	1
Subtotal	/4
Designing – flow chart	
Consistently and correctly identifies, and effectively implements, two-way or multi-way decisions (branching)	4
Consistently and correctly identifies two-way or multi-way decisions (branching)	3
Correctly identifies two-way or multi-way decisions (branching)	2
Creates an incomplete sequence of steps that attempt to solve a problem	1
Subtotal	/4
Producing and implementing	
Produces a program that runs with no errors, makes accurate comments, follows variable naming conventions, includes a minimum of five decisions, accurately incorporates iteration, provides questions and answers at an appropriate level, reflects the topic, provides feedback to the user and incorporates multiple endings	9–10
Produces a program that runs with no errors, makes accurate comments, follows variable naming conventions, includes all five decisions, accurately incorporates iteration, provides questions and answers at an appropriate level, reflects the topic and provides feedback to the user	7–8
Produces a program that runs with no errors, follows variable naming conventions, includes all five decisions, accurately incorporates iteration and provides feedback to the user	5–6
Produces a program that runs with no errors, includes some decisions and accurately incorporates iteration	3–4
Attempts to produce a program but it doesn't run or is unfinished	1–2
Subtotal	/10

Description	Marks
Evaluating	
Independently develops and applies contextual criteria to comprehensively examine design processes and solutions, providing a detailed and logically sequenced evaluation on design processes and the final game	4
Independently develops and applies contextual criteria to assess design processes and solutions, providing an evaluation on design processes and the final game	3
Independently develops contextual criteria to assess design processes and the final game	2
Lists unclear contextual criteria, simply recalling design processes and/or the game design	1
Subtotal	/4
Project management	
Consistently works independently, and/or collaboratively, to plan, develop and effectively communicate detailed ideas and information when managing projects	3
Works independently, and/or collaboratively, to plan, develop and effectively communicate ideas and information when managing projects	2
Sometimes works independently, and/or collaboratively, to plan simple ideas and shares some information when managing projects	1
Subtotal	/3
Total	/29

