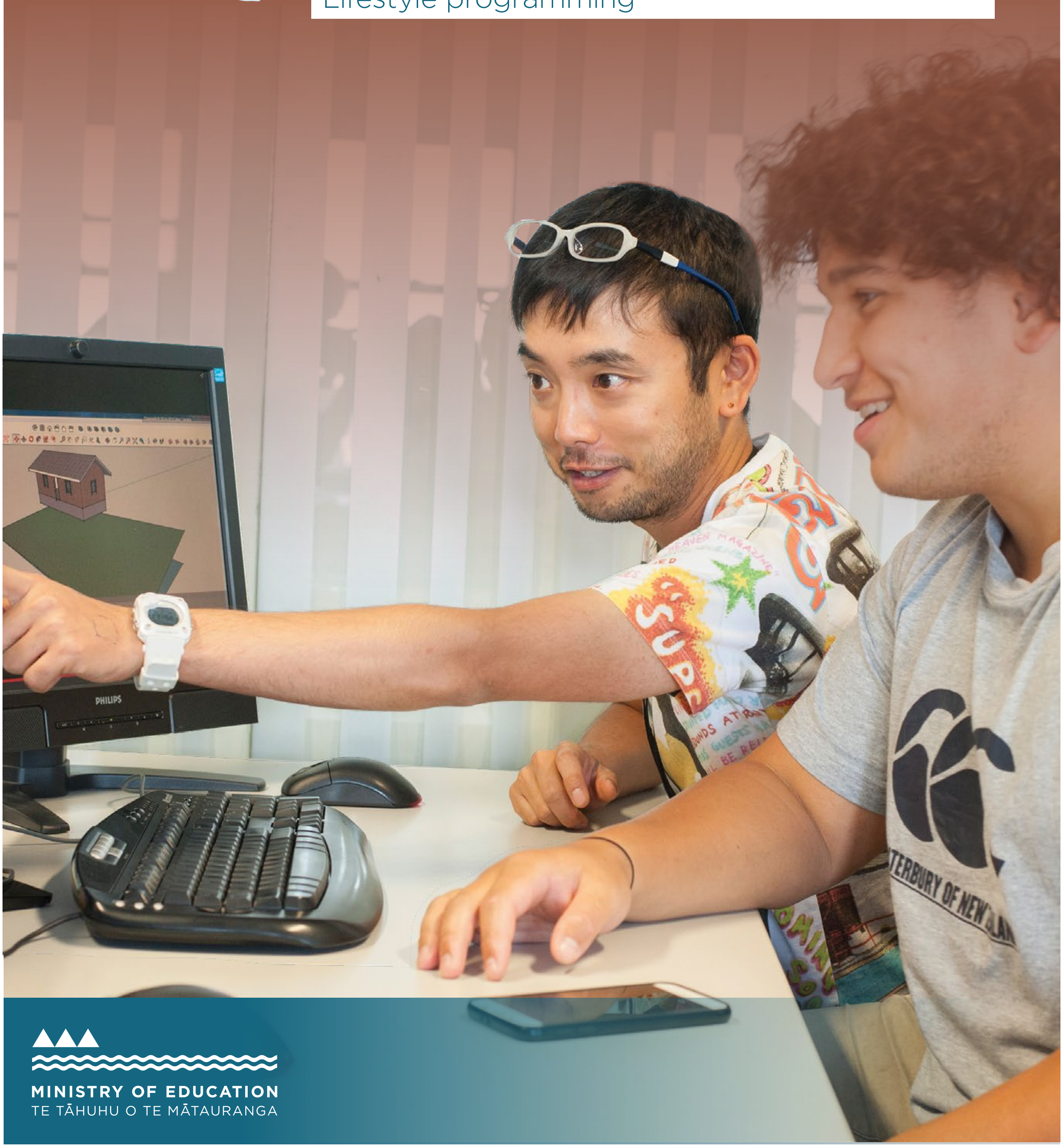
NCEA Level 2

DIGITAL TECHNOLOGIES & HANGARAU MATIHIKO

Teaching and learning programme

Lifestyle programming



MINISTRY OF EDUCATION
TE TĀHUHU O TE MĀTAURANGA

Developed by Jennifer Gottschalk, Massey High School

Published 2019 by the Ministry of Education
PO Box 1666, Wellington 6011, New Zealand

www.education.govt.nz

Summary of the teaching and learning programme

This teaching and learning programme is based on an eBook tutorial (flipped learning pedagogy) that includes embedded video. Students are given several problems related to real-world situations and are encouraged to develop programs to solve these problems. No prior programming knowledge is assumed, although previous programming experience is extremely useful.

One of the strengths of this resource is that students can work through the material at their own pace. Much of the teaching is included in the form of video tutorials, giving teachers more time to support individual students to gain the skills needed to create advanced programs.

By the end of this teaching and learning programme, students will be able to:

- design an advanced computer program using a variety of advanced programming techniques
- develop robust and flexible computer programs that are easy to use
- be ethical when designing and creating their outcome.

Duration

15–20 weeks.

The big ideas

- An outcome can be decomposed into smaller components.
- Each component can be developed, tested and, where relevant, refined.
- Component code can be combined to create a fully functional program.
- The program should be comprehensively tested to ensure that it works as intended.
- A function is a named section of a problem that performs a specific task eg the programme is a series of modules.
- Usability is important! Volunteers can test code and changes made to ensure that the final outcome is easy to use.
- Version control should be used to facilitate the process.

Alignment to the New Zealand Curriculum

DTHM – Computational Thinking: Progress outcome 7

Students use an iterative process to design, develop, document and test advanced computer programs.

DTHM - Designing and developing digital outcomes: Progress outcome 5

Links to other learning areas

The first task, a recipe moderniser, links to food technology and hospitality. The second task, a book-review generator, links to English. Throughout the process, students use skills learned in mathematics.

Teachers are welcome to add extra tasks (or edit the existing tasks) if they wish to connect to other learning areas

Teaching and learning pedagogy

The programme uses 'flipped' learning, where the process has been videoed and students are encouraged to create their own programs by following the video tutorials. They are also encouraged to go beyond the basics where possible. By using an eBook with video, teachers are free to work with individuals and troubleshoot in a way that would not be possible using more traditional methods.

Teachers could encourage students to collaborate and work in small groups during the learning phase for this standard.

At this level, students are well on their way to becoming independent learners, and they should be expected to use online resources such as Stack Overflow to find answers to their questions and help them solve coding problems.

Prior knowledge and place in the learning journey

No prior knowledge of Python is assumed. However, students who have experience in writing Python programs will be able to make rapid progress and will benefit from the use of this resource.

Resources required

The learning resource for this programme is found in the ePub and support files. Note that this includes 'teacher only' answers and a Teaching Guide. All of the resources are found here:

- ePub, videos and Teacher-only Answers
- Support files

Students will also need access to:

- Python
- Google Documents / Word
- gist.githum.com – used for version control

Software used:

- Python 3.x
- PyCharm (optional but very useful).

How might you adapt this in your classroom?

The tasks in the support files can be modified to suit a given class. Students should be encouraged to go beyond the basics if they have prior programming experience. While the resource is focused on Python programs, students could be encouraged to develop similar outcomes in the language of their choice.

The context of the programs can easily be changed. Teachers could ask students to create a nutrition calculator, calorie counter or even a meal suggestion app (where users input food that they have at hand and the program outputs recipes or meal ideas based on the listed ingredients).

Assessment

The default assessment task asks students to create a shopping-comparison tool using achievement standards 91896 and 91897. Students who wish to create an alternative program should be allowed to 'pitch' their idea and then create their desired outcome provided it will allow them to meet standards 91896 and 91897.

TERM OUTLINE

Students will need to develop a series of programs. For each program, they should work through the following steps so that when they are ready to be assessed, they have a good understanding of how to decompose the task into components, use project management tools (including version control) and create working code.

1. Decompose the problem (ie, break the problem down into a series of smaller components)
2. For each component:
   a. Plan how the component will be created (possibly by writing pseudocode or a 'to do' list in Trello).
   b. Write a test plan that allows all logical pathways for the component to be tested.
   c. Create the code using sensible variable names and comments that describe the code's behaviour. Students should ensure that they follow the conventions of their chosen programming language.
   d. Test the code.
   e. Refine the code if necessary, remembering to use version control as part of this process.
3. Once the components have been created and tested, make a working program by combining the components.
4. Test the program to ensure that it works.
5. Get a volunteer to test the program and make notes on improvements or refinements that can be made to make the program easier to use.
6. Make the changes and retest.
7. Write a brief paragraph explaining how the information from planning, testing and trialling of the components has resulted in a high-quality outcome. If students prefer, they can submit video evidence in which they explain how they used planning, testing and trialling to create (and refine) their outcome.

Teacher Note: *The duration or timing outlined below may vary – the times given assume students have no prior Python or programming experience. For students who have programmed at Level 1, the time scales will either be shorter, or their outcomes will be more sophisticated.*

Specific learning outcomes (may include what will be covered)	Duration	Learning activities	Resources provided
Students should be able to submit a fully functioning, easy-to-use program with evidence of how the program was developed, tested and refined. See Term Outline above for more details.	5 weeks	**Recipe moderniser task** See cycle above. In addition to learning how to decompose a problem and manage the development of a program, students will also learn how to: • get input from a user and check that it is not blank • get numeric input from a user and check that it is valid • find a scale factor based on user input • split user input into a number, unit and ingredient (string manipulation) • create a dictionary from a CSV file • convert amounts using information looked up in a dictionary (using a function) • refine the program to make it easier to use (ie, by adding in an introduction that is made available to first time users).	**Teaching material** ePub document relevant_implications.pdf **Support files** 01_ingredients_ml_to_g.csv 01_Recipe_Scaler document
See above for specific learning outcome	3 weeks	**Review generator** See cycle above. Students will also learn how to: • create robust functions that use multiple parameters • use a dictionary that has a key and a value. The value is a list (which is a clever way of having a single key with multiple values) • set up code so that it copes with a wide variety of possible responses • create lists based on the content of a single column .txt file.	**Teaching Material** ePub document relevant_implications.pdf **Support Files** 02_Review_Generator 02_review_adjectives

Specific learning outcomes (may include what will be covered)	Duration	Learning activities	Resources provided
See above for specific learning outcome	2 weeks	**Fund-raising profit calculator** See cycle above. Students will be given minimal video support for this task in the hope that they will be able to solve the problem and develop a working outcome by using knowledge gained from the previous tasks. They will need to pay close attention to how to work with or sort two-dimensional arrays. One of the requirements of the task will be for the program to output the costs of creating the project from highest to lowest.	**Teaching material** ePub document relevant_implications.pdf **Support files** 03_Fund_Raising_Calculator
See above for specific learning outcome	3 weeks	**Budget helper** See cycle above. Students should be able to complete this task independently.	**Support files** 04_Budget_Calculator
Assessment	3 weeks	See task below.	

ASSESSMENT TASK: LIFESTYLE PROGRAMMING – PRICE COMPARISON TOOL

Curriculum key concepts	*Computational thinking:* Students will decompose a program into its components and then write (and test) code to solve each part of the problem before assembling the components into a fully functional whole. That is, they will use a process to design, develop, document and test an advanced computer program. *Planning for practice:* students will use suitable planning and version control tools to support the development of their outcome. Students will also take the end-user into account and ensure that their outcome is easy to use.
Achievement standard(s)	91896 Use advanced programming techniques to develop a computer program 91897 Use advanced processes to develop a digital technologies outcome
NCEA Level	2
Credits	91896 – 6 91897 – 6
Learning time guidance	17 weeks for the learning 3 weeks, 15 hours' class time for the assessment
Length guidance if appropriate	Documentation showing the testing of the components as they are developed and refined can be quite lengthy because annotated screenshots tend to take up a large amount of space. However, this should be managed to be as concise as possible.
Due date	Teacher to insert

ASSESSMENT TASK: LIFESTYLE PROGRAMMING – PRICE COMPARISON TOOL

Achievement criteria (AS91896)

Achieved	Merit	Excellence
Use advanced programming techniques to develop a computer program.	Use advanced programming techniques to develop an informed computer program.	Use advanced programming techniques to develop a refined computer program.

Achievement criteria (AS91897)

Achieved	Merit	Excellence
Use advanced processes to develop a digital technologies outcome.	Use advanced processes to develop an informed digital technologies outcome.	Use advanced processes to develop a refined digital technologies outcome.

ASSESSMENT TASK: LIFESTYLE PROGRAMMING – PRICE COMPARISON TOOL

Your task:

You want an easy way to compare the price of various products and have decided to write a computer program to allow you to do this.

Specifications:

- Your program will calculate the unit price for each product, display the price comparison information in an easy-to-read format and recommend the 'best buy' based on value for money and the available funds.
- Your program should ask the user how much money they have on-hand. You should decide on a suitable minimum amount (eg, $10.00)
- Your program should allow users to enter the details for multiple products that are being compared.

What you need to think about before you begin this assessment:

- Who your program is aimed at
- What user inputs you will need
- How you can ensure that users enter valid input
- How the data will be manipulated, stored and retrieved to solve the problem
- How you will give feedback to the user
- How you can explain and address the relevant implications.

What you need to do (follow these steps):

1. Decide on an appropriate planning methodology and what project management and version-control tools you will use to manage your program development.
2. Set up any necessary planning or project management tools.
3. Decide how you will collect input from your users and how you will structure your output. For example:
   a. How will you get users to give the input you need to compare prices?
   b. How will you ensure that users enter valid input?
   c. Will your program allow users to enter amounts in different units (eg, grams and kilograms)?
   d. Will you allow users to go over budget for comparison purposes?
   e. How will you display the results to the user? Will they be alphabetical or by price (or will you let the user decide)?
4. Decompose your program into the different components you need to incorporate into the final program (eg, get input, calculate unit price, sort data, present results).
5. Throughout your development, you must trial multiple components. For example, this could include different ways to get user input, different ways of handling prices that are over-budget, etc.
6. Select the best components to include in your final program, based on the results of your testing and trialling.

7. Use your selected version control tools and techniques to save successive versions of your code and keep evidence of how you created the program in an ongoing manner (eg, screenshots showing your file structure with appropriately named versions and program components, including brief annotations of the changes made in each version).
8. Ensure that your testing and trialling includes both expected cases and relevant boundary cases (eg, what happens when users get close to their budget). You may want to get other students or your family or whānau to test your program at each stage and provide feedback to help you improve your final program. Using others to test the program will help to ensure that it is comprehensively tested for many different cases (including expected and relevant boundary cases). Note the improvements that could be made based on the testing and implement your changes.
9. You should also think about the advanced programming techniques that will best make your program flexible and robust.
10. Throughout the development of your program code, ensure that you document your program with appropriate variable and module names and comments that describe code function and behaviour. Follow the common conventions of your programming language (eg, naming conventions or rules for program layout). The code should be well-structured and logical.
11. Comprehensively test your final program to ensure that it functions correctly and is of high quality (eg, bug-free, has well-presented and easy-to-understand instructions, contains all the required information).
12. Note: Testing can be recorded by making a brief screencast showing the outcome being comprehensively tested. If desired, you can take screenshots of your screencast and annotate them.
13. Discuss how the information from planning, testing and trialling the components of your program assisted you to develop a high-quality outcome. This can be in the form of a screencast, document with annotated screenshots, online presentation or oral presentation to your teacher or class.
14. Show how your program has addressed the relevant implications.

What you will hand in:

- Material that:
  - explains the relevant implications (A) and shows how they have been addressed (M/E)
  - shows how project management tools have been used to manage the programming process (eg, evidence could include a series of screenshots showing how a Trello or Kanban board has been used to manage the outcome's development)
  - shows how the task has been decomposed
  - shows how each component has been developed and tested, including trialling and testing of techniques where appropriate
  - provides evidence that the completed program has been comprehensively tested
  - includes evidence of how version control has been used
  - explains how you used the information from your trialling and testing to develop a high-quality outcome.
- Your code. This should include:
  - the code for the various components
  - code for a fully working program (and any other files that are needed to run the program).

ASSESSMENT SCHEDULE

AS91896 USE ADVANCED PROGRAMMING TECHNIQUES TO DEVELOP A COMPUTER PROGRAM

Programme 6: Lifestyle programming

Credits: 6

Final grades will be determined on a holistic judgment of the evidence against the achievement criteria.

CRITERIA	JUDGMENTS	COMMENTS
Written code for a program that performs a specified task	**For example (partial evidence):** The student's program allows text variables entered by users to be stored under the product's name, uses Boolean variables to control loops that check data is valid and uses float variables for the cost-unit price. They have used sequence, selection and iteration control structures within their code. The program processes input from the user and outputs a valid recommendation.	
Used advanced techniques in a suitable programming language	**For example (partial evidence):** Functions are used to check if user input is valid and/or do calculations that would otherwise be repeated in the code. The program uses a two-dimensional array or list to store and update information (eg, item name, amount, price and unit cost).	
Set out the program code clearly and documented the program with comments	**For example (partial evidence):** Most of the variable names are clear, and the code includes some comments, but these comments don't describe the code's function. Eg, # Unit cost	
Tested and debugged the program to ensure that it works on a sample of expected cases	**For example (partial evidence):** The student has provided evidence of testing their program. The program works on expected input but may crash on boundary or invalid input. For example, screenshots, testing tables.	

ASSESSMENT SCHEDULE

AS91896 USE ADVANCED PROGRAMMING TECHNIQUES TO DEVELOP A COMPUTER PROGRAM

Programme 6: Lifestyle programming

Credits: 6

Final grades will be determined on a holistic judgment of the evidence against the achievement criteria.

CRITERIA	JUDGMENTS	COMMENTS
Documented the program with appropriate variable or module names and comments that describe code function and behaviour	**For example (partial evidence):** The student uses descriptive variable and function names, eg, the number-checking function might have been called ‘num_check’, the variable holding the value of the unit cost might be called ‘unit_cost’. The code has comments at key points, eg, ‘function checks that user input is a number that is between a given lower and upper bound’.	
Followed common conventions for the chosen programming language	**For example (partial evidence):** The student uses snake_case. Function definitions are placed before or after the main function, as per the programming language. Layout conventions are followed, eg, whitespace between definitions. Indentation and/or bracketing conventions are followed as per the programming language. The student has used an automated tool to check that their code follows common conventions.	
Tested and debugged the program effectively to ensure that it works on a sample of both expected and relevant boundary cases	**For example (partial evidence):** The student has provided evidence of testing to confirm that it works correctly on a range of boundary cases, eg, for a budget of between $5 and $100, has tested at $4.99, $5.00, $100.00 and $100.01. Student testing methodology is effective within the context of the problem.	

ASSESSMENT SCHEDULE

AS91896 USE ADVANCED PROGRAMMING TECHNIQUES TO DEVELOP A COMPUTER PROGRAM

Programme 6: Lifestyle programming

Credits: 6

Final grades will be determined on a holistic judgment of the evidence against the achievement criteria.

CRITERIA	JUDGMENTS	COMMENTS
Ensured that the program is a well-structured, logical response to the task	**For example (partial evidence):** The student has used functions where appropriate. Functions have been used to avoid repeated code. The code is a well-structured, logical response. For example, lack of use of global variables, narrowest possible scopes for variables, functions which perform one task, exceptions handled as close to where they were raised as possible.	
Made the program flexible and robust	**For example (partial evidence):** It is easy to extend the functionality of the code (eg, a function has been used to check that units are valid, it would be easy to update the function to add units from other countries, ie, ounces and pounds instead of g and kg). The code works for expected, unexpected and boundary values. They have used appropriate techniques, such as try/except to check for validity.	
Comprehensively tested and debugged the program.	**For example (partial evidence):** Student has supplied test plans and/or annotated screenshots or a screencast showing that the program components (and final program) have been tested to ensure that it works correctly, eg, they have used others to comprehensively test their program to ensure that it responds gracefully to a variety of input.	

ASSESSMENT SCHEDULE

ACHIEVEMENT STANDARD 91897 USE ADVANCED PROCESSES TO DEVELOP A DIGITAL TECHNOLOGIES OUTCOME

Programme 6: Lifestyle programming

Credits: 6

Final grades will be determined on a holistic judgment of the evidence against the achievement criteria.

CRITERIA	JUDGMENTS	COMMENTS
Used an appropriate planning methodology to plan the development of a digital technologies outcome	**For example (partial evidence):** The student has decided to follow an Agile-based planning methodology. The student has used Trello (or an off-line visual planning board) to manage their development process. Student code is clearly named and shows version numbers or indicates which part of the decomposition has been coded. They have used GiST to create versions of the various components at key points. This allows the teacher access to the code and ensures that it is backed up in the cloud. At least 2 project methodology techniques or tools are required.	
Decomposed the outcome into smaller components	**For example (partial evidence):** They have broken their outcome down into a series of components. For each component, they have created a piece of code and tested that code.	
Trialled the components	**For example (partial evidence):** They have trialled different methods for getting user input, such as allowing users to enter one item at a time in one line vs prompting users for each item's name, amount, unit and cost. Towards the end of the process, the components have been combined into a fully working version of the outcome.	
Tested that the digital outcome functions as intended	**For example (partial evidence):** The student has provided evidence of testing with expected cases. They have also provided screenshot or screencast evidence showing that the actual program works as expected. They may not have had others test the program or looked at boundary or unexpected cases.	
Explained the relevant implications	**For example (partial evidence):** Student has explained the importance of creating code that is functional and easy to use. They have explained why their program needs to be clearly laid out.	

ASSESSMENT SCHEDULE

ACHIEVEMENT STANDARD 91897 USE ADVANCED PROCESSES TO DEVELOP A DIGITAL TECHNOLOGIES OUTCOME

Programme 6: Lifestyle programming

Credits: 6

Final grades will be determined on a holistic judgment of the evidence against the achievement criteria.

CRITERIA	JUDGMENTS	COMMENTS
Trialled multiple components and/or techniques and selected those that are most suitable	**For example (partial evidence):** The student trialled various techniques for getting input from the user. They trialled alternative functions to get users to input data. They selected the function that provided users with the best way of entering the data.	
Used information appropriately from testing and trialling to improve the functionality of the digital technologies outcome	**For example (partial evidence):** They trialled different input methods with friends and family and selected the one that was rated easiest to use and understand by the testers.	
Used project management and version control tools and techniques to effectively manage the development of a digital technologies outcome	**For example (partial evidence):** The student has updated their development plan when they realised a new component was needed to improve the program, making it easier to sort and display the data. They have saved versions of the outcome (eg, developing new versions of the working file) at each stage of the development. They have written easily understood descriptions for each GiST that has been created (these are automatically dated by the system). They have used evidence from their trialling to inform their decisions when updating their project plan.	

Addressed the relevant implications	**For example (partial evidence):**
	For example (partial evidence):
	The student has created a program that functions as intended and is easy to use (rather than just explaining that these are important attributes). They have provided annotated screenshots of their program to illustrate what they have done to address functionality, usability, accessibility (etc.) implications. As the output is essentially text, aesthetics is less relevant than other implications but having a clear, easy-to-read layout is still important.	

ASSESSMENT SCHEDULE

ACHIEVEMENT STANDARD 91897 USE ADVANCED PROCESSES TO DEVELOP A DIGITAL TECHNOLOGIES OUTCOME

Programme 6: Lifestyle programming

Credits: 6

Final grades will be determined on a holistic judgment of the evidence against the achievement criteria.

CRITERIA	JUDGMENTS	COMMENTS
Discusses how the information from planning, testing and trialling of components assisted in the development of a high-quality outcome.	**For example (partial evidence):** The student has presented a brief, reflective summary of how the information from planning, testing and trialling of the components of their program assisted them to develop a high-quality outcome. They provided annotated screenshots of the changes they have made throughout the process and how feedback from the users and the testing process helped them to refine their program. They also included screenshots of their Trello board with a reflection on how it guided their development process and helped them to complete all the components and keep on track with their time management.	