Internal Assessment Resource- Mock Exam

Digital Technologies Level 2

2025

This resource supports assessment against

[**AS91896**](https://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2019/as91896.pdf) V1 - Use advanced programming techniques to develop a computer program (6 credits)

[**AS91897**](https://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2019/as91897.pdf)V1 - Use advanced processes to develop a digital technologies outcome (6 credits)

**Instructions:**

* Read the entire assessment task and assessment schedules for programming and processes
* This assessment is due by Thursday, 10th April
* Open this document in Desktop version. Save in your document and Turn in

Mock Assignment Section in Teams

* You must ensure you have not breached the Copyright Act

**Authenticity Statement:**

This is an open book assessment, and you can go to various sources for guidance—for example, OneNote, w3Schools. Acknowledge sources of learning, including any help from teacher, friends, and family before or during this assessment. If work is non-authentic in any aspect, no credit will be given regardless of the quality. This will apply to **ALL** pupils involved in the misconduct. There will be no opportunities offered for further assessment in this standard for the current year.

Tick the box to acknowledge that you read and understood the Authenticity Statement

Sign here:

# Overview/Kupu Arataki

This assessment activity requires you to **plan**, **trial**, **test** and **develop** a computer program using advanced programming techniques.

You will use a development process and project management tool to help you make informed decisions throughout the coding, testing and trialing of your program and show ongoing testing and refinement to improve the functionality and quality of your program.

You will be assessed on how effectively you plan your development, decompose the outcome into smaller components, test and refine your program so that it is a high-quality response to the task (e.g. well-structured, logical, flexible, robust and comprehensively tested).

When planning and developing your program, you must ensure your program:

* uses variables storing at least two types of data (e.g. numeric, text, Boolean)
* uses **sequence** (commands), **selection** (conditionals) and **iteration** (loops) control structures
* takes input from a user, sensor(s), or other external source(s)
* produces output

**AND** includes **two or more** advanced programming techniques, such as planning and writing code that:

* defines and manipulates multidimensional data in collections (e.g. lists, dictionaries)
* creates methods, functions, or procedures that use parameters and return values
* responds to events generated by a graphical user interface (EasyGui or Tkinter)

# Tasks/Hei Mahi- Choose any one of the brief to create a computer program.

**AS 91896- Use Advanced Techniques to develop a Computer Program**

**Brief 1: Quiz game with Data visualization:**

The program requirements are:

* **Quiz Questions**:
* Include at least **5 multiple-choice questions**.
* Each question should have **4 answer choices**.
* Use appropriate data structures to store the questions and choices (e.g., **lists**, **tuples**, or **dictionaries**).
* **User Interaction**:
* Display one question at a time and allow the user to select an answer.
* Keep track of the user's **score** (number of correct answers).
* Show the **final score** at the end of the quiz.
* **Data Visualization**:
* Use **Matplotlib** or a similar library to create a **bar graph**.
* Plot the **user’s score** and the number of **incorrect answers**.
* Add **labels**, a **title**, and different **colors** for correct vs. incorrect answers.
* **Code Structure**:
* Include at least **2 functions** for reusability (e.g., a function to ask questions or calculate score).
* Use **variables** and **constants** where appropriate (avoid hardcoded values).
* **Optional**
* Allow the quiz to support **multiple users** and plot scores for all users.
* Store results in a **CSV file** and read from it for visualization.

**Brief 2: Personal Budget Tracker with Data visualization**

Program Requirements:

* **User Input:**
  + Allow the user to enter at least 5 spending categories (e.g., food, transport, entertainment).
  + For each category, ask for the amount spent.
  + Ensure values entered are valid numbers.
* **Data Storage:**
  + Store the spending data in a suitable data structure (e.g., a dictionary or two lists).
  + Calculate the total amount spent.
* **Data Visualization:**
  + Use Matplotlib to create a bar graph or pie chart.
  + Show spending per category visually.
  + Include labels, a title, and use colors to differentiate categories.
* **Code Structure:**
  + Include at least one function (e.g., for entering spending or generating the graph).
  + Use variables and constants effectively.
  + Display the total spending at the end.
* **Optional (for bonus marks):**
  + Allow the user to save their spending data to a CSV file.
  + Read from the CSV file and plot spending trends over multiple months.

**AS 91897- Using Advanced Processes to develop a Digital Outcome**

This standard requires students to **plan, develop, and improve a digital outcome** using **advanced processes**. This includes **project management tools**, **programming**, and **testing**

**Specifications of the Project Management:**

**Specifications of the Project Management Task:**

**1. Planning Using a Project Management Tool (e.g., Trello):**

* Create a **Trello board** (or similar tool) to manage the project.
* Set up **lists** such as *To Do*, *Doing*, *Done*.
* Add **tasks/cards** for each stage of the project.
* Assign due dates and responsibilities (if working in a team).

**2. Timeline and Milestones:**

* Set a timeline for the project with milestones (key goals) clearly marked.
* Show progress tracking and any changes made during development.

**3. Task Decomposition:**

* Break the project into smaller tasks (decompose the problem).
* Clearly describe what each task involves.

**4. Algorithm Design and Flowcharts:**

* Write algorithms (step-by-step instructions) for key processes.
* Draw flowcharts to visually represent the logic of your program.

**5. Coding the Digital Outcome:**

* Use an appropriate programming language (e.g., Python).
* Follow coding conventions (clear structure, comments, meaningful variable names).
* Ensure the code is functional and efficient.

**6. Testing and Documentation:**

* Perform systematic testing of your program.
* Take screenshots of test results to show evidence of testing.
* Provide videos of your program in action to demonstrate functionality.

**7. Version Development:**

* Develop at least 2 or 3 versions of the program and version development should be based on peer feedback
* Version 1: Basic functional version.
* Version 2: Improved version with added or enhanced features.
* Version 3: Further refinement, including bug fixes, usability improvements, or new functionality.
* Clearly document the differences and improvements in each version.

# Task 1 Project and project requirements

* Start with a good introduction to your project:

My Chosen Brief:

**Requirements:**

* Python version 3.2.1
* Libraries and modules:
* Mat plot lib
* EasyGui
* Trello for project management
* Testing Document

**Overview of my program: (What does your program do? What is its purpose? Who are the target audience etc.)(you can draw a sample diagram with arrows to show how your program works.)**

My program is basically a Quiz program that stores question answers in a dictionary and iterates them using for loop.

The purpose of the quiz is to help level 2 students to quickly go through the subject just before the exam which can help them to memorize the important aspects of the subject.

**Flowchart:**

**Relevant Implications:**

|  |  |  |
| --- | --- | --- |
| Relevant Implications | Description / Explanation | How is this implication implemented in my program. (Do this after developing all your versions.) |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |
| --- | --- |
| Name of app |  |
| Project management | Give a shared link to Trello. Add screen shots of project management to developmental logs. Your development logs for Trello should show effective management of the development of project with tasks, timelines and their completion. |
| Version control system | Save files as different version and submit them for your final submission! List the Versions of the program |
| Flowchart/  Pseudocode | Give a shared link to draw.io or similar. Add screen shots of plan in Task 5 |

# Task 2 Developmental Logs (Processes)

|  |  |
| --- | --- |
| Date | Task |
| 3/17/2025 | Initial work done such as setting up answer sheet and answering first two questions |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Include a Trello Screenshot:

Version1:

Flowchart:

Test cases (Programming)

Test cases describe different scenarios that occur when the user inputs data into the system you build. There are three basic types – expected, boundary and invalid. To call it comprehensive testing, you must plan several test cases in the first four columns before the development and complete the fifth column before you hand in:

* Expected: These are values that you would expect the user to enter in normal circumstances. They fall within the expected range of input. For example: age = 5 or 18 or 50 or any number from 0-100
* Boundary: These are values that are just at, below and above the boundary values.

These are the edge cases for normal use. If the minimum age of the user is 13. Then age = 12 or 13 or 14 are boundary values.

* Exceptional (Invalid): These are values that you wouldn’t expect the user to enter in normal circumstances. They fall outside the expected range of input. For example: age = 150 or ten or -15 or 13.5 or apple.

**Include a Video of testing of Version 1:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| User input (*A short explanation)* | Test Case  (*What would the user input?)* | Input Type  *(expected, boundary or invalid)* | Output  *(How would your system respond?)* | Actual Output  *(Screenshot)* | What are next steps? How to fix or improve |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Testing Table of Version1**

**Feedback of End Users on Version 1:**

|  |  |
| --- | --- |
| End Users | Feedback on Version 1 |
|  |  |
|  |  |
|  |  |

Include a Trello Screenshot here:

What is lacking in version 1? What did the end users say? What features are you going to add in version 2?

Version2 Flowchart

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| User input (*A short explanation)* | Test Case  (*What would the user input?)* | Input Type  *(expected, boundary or invalid)* | Output  *(How would your system respond?)* | Actual Output  *(Screenshot)* | What are next steps? How to fix or improve |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Include a Video of Version 2**

**Testing Table of Version2**

**Feedback of End users on Version 2**

|  |  |
| --- | --- |
| End Users | Feedback on Version 2 |
|  |  |
|  |  |
|  |  |

Include a Trello Screenshot:

What is lacking in version 2? What did the end users say? What features are you going to add in version 3?

Version3 Flowchart

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| User input (*A short explanation)* | Test Case  (*What would the user input?)* | Input Type  *(expected, boundary or invalid)* | Output  *(How would your system respond?)* | Actual Output  *(Screenshot)* | What are next steps? How to fix or improve |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Include a Video of Version 3**

**Testing Table of Version3**

**Feedback of End users on Version 3**

|  |  |
| --- | --- |
| End Users | Feedback on Version 2 |
|  |  |
|  |  |
|  |  |

Include a Trello Screenshot:

Reflective summary

Present a summary of how the information from planning, project management and design thinking processes, relevant implication, and testing and trialing assisted you to develop a high-quality outcome. This may include annotated screen shots of Trello, programme and iterative process that guided your development process and helped you complete all components and manage your time effectively.

Acknowledgements:

Acknowledge sources of learning, including any help from teacher, friends, and family before or during this assessment.