Rangitoto College Internal Assessment Resource

Achievement standard: 91883v1

Standard title: Develop a computer program

Credits: 4

Resource title: Te Reo Maori Quiz

Resource reference: adapted from Digital Technologies | Hangarau Matihiko 1.7A

Student/Ākonga instructions

Introduction/Kupu Arataki

This assessment activity requires you to create a language quiz to help people to learn Te Reo.

You are going to be assessed on how well you structure and refine your program. A well-structured, refined program will have code that is easy to follow, update and maintain, and is free from bugs.

You may discuss potential problem solutions or ideas for your quiz word questions/answers with others, but all of your program must be entirely constructed and written by you.

*This is an **individual** assessment activity. The work you submit needs to be entirely your own. You may use internet searches and other reference documentation for help, but not your peers, tutors, etc. Make sure to keep your code private to you, e.g. do NOT put your code on replit.com.*

Start Date:
Final Hand in Date:

Task/Hei Mahi

You are required to create a quiz to help your classmates learn Te Reo Maori. The two topics to choose from are:

- Places in NZ, e.g. Auckland in Te Reo is Tāmaki Makaurau
- Fruit and vegetables, e.g. apple in Te Reo is āporo

*There are **two possible quiz styles**:*

- A **multiple choice** quiz
  - one word presented at a time with multiple possible answers – user selects one
- A **word answer** quiz
  - present one word at a time, user enters their attempt at the matching word

The quiz must include at least five distinctive questions. You can choose to repeat the questions if the user fails to provide the correct answer the first time. Regardless of which quiz style you choose, the program must be able to restart once finished. At the end of the quiz, the user should be able to decide whether to restart or end the quiz.

Planning:

Decide on the style of quiz that you will make, keeping in mind the time you have available, the core requirements, your programming ability and the software and hardware resources available to you.

You need to think about:

- **How your quiz begins.** For example, should you have multiple levels of difficulty? Is your quiz only appropriate for a certain age range?
- **How to store your data.** What variables will you require and what type of data will your variables store (e.g. text, numeric, Boolean). Will you store data in collections (e.g. lists, arrays or dictionaries) to improve the structure, flexibility and robustness of your program? Will you be using functions?
- **How to structure your program.** What procedural structure will your program require? Will you create functions/method/procedures to improve the structure, flexibility and robustness of your program?
- **How you will give feedback to the user.** This could be in the form of a text prompt, a correct answer should the user answer incorrectly, a score that tracks user progress or any other method that you see fit to use. Also consider if you want to give immediate feedback, or only after quiz is complete.
- **How you will test that the program works properly.** To test a program in a comprehensive way, you need to plan how you will test the program for various cases such as expected, boundary and unexpected input. *Before you start coding* keep a record (eg table/test plan template) with:
    - the sample test inputs you plan to test (some expected, some boundary and some unexpected cases)
    - what you expect to happen (expected results)
    - include a space to record what actually happens for each case (actual results) after you perform the test.

Coding:

For achieved you must:

- get some input from the user
- include variables of at least two different types, and sequence, selection, and iteration control structures
- include one or more of:
    - data stored in collections (e.g. **lists**, **dictionaries** for the word sets). To meet the requirement for a plural, you *may* want to add difficulty levels to your quiz.
    - user-defined **functions**.
- give some feedback to the user (see ideas in planning section)
- set out your code clearly
- document your code with comments
- test and debug your code and ensure it works on an appropriate set of expected test cases

For higher grades ensure that:

- you **comment your code appropriately** as you develop it and use **variable names** and **comments** that **describe code function** and **behaviour**.
- you have **followed accepted programming language and organisation conventions (eg PEP-8, PEP-257, CONSTANTS, snake_case, order of layout)**
- you have chosen a **well-structured, logical response** to the task.
- your code is **robust***. * See NZQA AS91883 Explanatory Note 5
- Wherever possible your code has a **flexible structure*** to allow for continued development**. For example, it should be very easy to make your program randomise the order of quiz questions/answers each time the program is run (if not already coded).** * See NZQA AS91883 Explanatory Note 5
- Your code handles a range of expected, boundary and unexpected values

Handing in

Once completed you will need to hand in your code and test plan to G:\DigiTech\DROPBOX\91883 (1.7). Your Python file must be named according to the following format:

- 91883 Te Reo Quiz – *Your name*.py
- 91883 Test plan – *Your name*.pdf