Level 6 – New Zealand Curriculum

Digital Technologies | Hangarau Matihiko

Teaching and learning programme

Planning and programming (Python)

MINISTRY OF EDUCATION
TE TĀHUHU O TE MĀTAURANGA

Developed by Jennifer Gottschalk,
Whangaparaoa College 2017

The full teaching and learning programme resources, associated materials and an assessment task will be supplied in 2018.

External links to websites
The Ministry of Education does not accept any liability for the accuracy of information on external websites, nor for the accuracy or content of any third-party website accessed via a hyperlink from this resource. Links to other websites should not be taken as endorsement of those sites or of products offered on those sites. Some websites have dynamic content, and we cannot accept liability for the content that is displayed.

Published 2017 by the Ministry of Education
PO Box 1666, Wellington 6011, New Zealand

www.education.govt.nz

Summary

The teaching and learning programme is based around an eBook/tutorial that includes embedded video. Students are given several problems related to games of chance, and they are encouraged to develop programs to solve the problems. No prior programming knowledge is assumed.

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

- decompose a problem into smaller sub-problems
- create and test code that solves each problem
- combine the code into a fully functional program
- test that the program is easy to use.

Duration (terms, weeks, teaching periods)

13 weeks including assessment; this assumes five hours class time per week.

Key teaching and learning concepts – the big ideas

- A problem can be decomposed into smaller sub-problems.
- Each sub-problem can be developed and tested.
- The code fragments can then be composed into a fully functional program.
- Testing should include boundary and unexpected values.
- Functions can be used to avoid repeating code.
- Usability is important! Volunteers can be asked to test code, and changes can then be made to ensure that the final outcome is easy to use.

Alignment to NZC and/or Te Marautanga

Students will:

- design and develop a basic program
- ensure that their outcome is easy to use (preferably by including some usability testing as part of the process)

- be ethical when it comes to designing and creating their outcome; specifically, they will honour copyright
- meet the criteria set out in achievement standards 91883 and 91884.

Links to other learning areas

This programme involves developing games of chance and links to the Chance and Data standards in mathematics.

Teaching and learning pedagogy

The programme makes considerable use of 'flipped' learning, where the process has been videoed and students are encouraged to develop their own programs by following the video tutorials. They are also encouraged to go beyond the basics where possible. Using an eBook with embedded video means that teachers are free to work with individuals and trouble-shoot 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 teaching and learning.

Prior knowledge/place in learning journey

No prior knowledge is assumed. If students have programmed before, this programme will help to formalise their learning and also ensure that they develop good coding (and design) habits.

Resources required

- Python
- Google Documents and Word

Software used:

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

How you might adapt this in your classroom

The tasks in the support files area can be modified to suit a given class. Each task has suggestions for extension activities, and students should be encouraged to go beyond the basics if they have prior programming experience. Whilst the programme is focused on Python programs, students could be encouraged to develop similar outcomes in the language of their choice.

Assessment

An assessment task will be created for this programme. The default task asks students to create a mathematics quiz. It is hoped that student aiming for Merit and Excellence grades will go beyond the basics in their assessment and use this an opportunity to create a high-quality outcome showing what they have learned.

Week 1	Weeks 2 & 3	Weeks 4 & 5	Weeks 6 & 7	Weeks 8 & 9	Weeks 10 & 11	Weeks 12 & 13
Skills intro	Lucky unicorn task	Higher/lower game	Rock, paper, scissors	Collect them all	Car racer game	Assessment

Programmes: The materials below are used for each task.

- ePub
- Support files
- Documentation template
- Program planning helper

Term outline

The Learning context:

What is being covered	Approximate duration	Specific Learning Outcomes Students will be able to:	Learning Activities
Basic skills	5 hours maximum	Learn or review how to: • name variables • get user input • loop code • cope with unexpected input • write functions.	• Discuss the importance of commenting code • Learn how to name variables correctly • Learn how to get user input and the difference between strings and integers • Learn how to write ‘if’ statements and use <, >, ==, etc • Learn how to write ‘while’ loops • Learn how to implement try or except code • Combine what we have learned to create a number checker • Change our number checking code into a function.

The Learning context *(continued)*:

What is being covered	Approximate duration	Specific Learning Outcomes Students will be able to:	Learning Activities
Lucky unicorn task	10 hours	• Break down the task • Code and test each section • Create a fully working program from the components • Test that the program is usable.	• Analyse the 'lucky unicorn' problem and break it down into a series of smaller components • Create test plans for each component (note that this can be done at the start of the task, or test plans can be generated before creating a given component) • Learn how to create 'for' loops (ie, 'loop Interlude') • Randomly choose items from a list • Check that the probability of getting a unicorn is not too high • Set up payment mechanics • Set up end-game mechanics • Combine all the sub-programs into a fully functioning program • Usability test the program • Fix the output statements so they are easy to read • Retest that the program works correctly **Checkpoint 1:** Submit the entire 'lucky unicorn' program and the associated documentation for feedback. Your program may well be different from the one in the video walkthrough.

The Learning context *(continued)*:

What is being covered	Approximate duration	Specific Learning Outcomes Students will be able to:	Learning Activities
Higher/lower game	10 hours	• Break down the task • Code and test each section • Create a fully working program from the components • Test that the program is usable.	• Decompose the problem and create test plans for the various components (note that test plans can be created at a later date if preferred, as long as they are generated before the code for that section is written) • Create the game by using what you learned in 'lucky unicorn' and following the provided videos for 'new' code • If possible, create a fully featured game rather than a basic version. **Checkpoint 2:** Submit the entire 'higher/lower' program and the associated documentation for feedback. Your program may well be different from the one in the video walkthrough.
Rock, paper, scissors	10 hours maximum	• Break down the task • Code and test each section • Create a fully working program from the components • Test that the program is usable.	• Decompose the problem and create test plans for the various components (note that test plans can be created at a later date if preferred, as long as they are generated before the code for that section is written) • Create the game by using what you have learned in previous tasks • If possible, make the game your own. Consider how you'd like the scoring to work (eg, best of x or the first to x). *Teacher note: Students should be able to compare user and computer choice using five 'if' statements. If they want to use more than that, encourage them to think about how they can do it in less. Using more statements is OK and will work but it is inefficient and probably indicates that the student is working at an A level.* **Checkpoint 3:** Submit the entire 'Rock, paper, scissors' program and the associated documentation for feedback

The Learning context *(continued)*:

What is being covered	Approximate duration	Specific Learning Outcomes Students will be able to:	Learning Activities
Collect them all	10 hours maximum	• Break down the task • Code and test each section • Create a fully working program from the components • Test that the program is usable.	• Decompose the problem and create test plans for the various components (note that test plans can be created at a later date if preferred, as long as they are generated before the code for that section is written) • Create the tool by using what you learned in previous tasks. It is OK to recycle or repurpose functions that you have used in previous tasks. **Checkpoint 4:** Submit your 'collect them all' and the associated documentation for feedback.
Car racer game	10 hours	• Break down the task • Code and test each section • Create a fully working program from the components • Test that the program is usable.	• Decompose the problem and create test plans for the various components (note that test plans can be created at a later date if preferred, as long as they are generated before the code for that section is written) • Create the game by using what you learned in previous tasks. It is OK to recycle or repurpose functions that you have used in previous tasks. *Teacher note: There are a wide number of possible correct solutions to this task. Students could be encouraged to use the turtle module and race turtles instead of cars. This entire exercise is an extension task, and if students don't get this far, that is not a problem. This is really an opportunity for kids to explore and have a bit of fun. My solution uses 2-dimensional arrays, and this is not required at level 1. Your very top students might need to know that functions can only return one thing, but that thing can be a list.* **Checkpoint 5:** Submit your 'car racer' game and the associated documentation for feedback.