# Teacher Guide: Programme 1

# Planning and Programming (Python)

These notes are to support you to teach and assess AS 91883 (programming) and AS 91884 (planning).

Students will need an ePub reader such as [Readium](http://readium.org/) (a chrome add on), [Qiu Reader](https://addons.mozilla.org/en-US/firefox/addon/qiureader/) (a really good Mozilla add on) or iBooks. They will also need access to the ePub and support files.

Students should be encouraged to watch the videos in the ePub at least once so that they know what to do. All of the videos have sound **and** close captioning - the vast majority of students will probably find listening to the videos really useful. You may need to explain to some students that they should watch between 10 and 15 seconds of video and then copy what they see.

The “Teacher Answers” folder contains possible answers for all five problems included in the support files area. Inside the main answer folder, there are sub-folders which contain code for each task. There are also documents showing how students can present their planning and testing evidence.

|  |
| --- |
| **Important**: The ‘model answers’ show one possible approach to a given problem. I would expect student answers to be quite different to the ones provided in the ‘answers’ area. |

In the support files folder, there is a documentation template. Students could use this template and complete it for each task. Students should begin by decomposing the task and then for each component they should create a test plan, create the code and then test it using their test plan. Note that they don’t need to create all of the mini-test plans in one sitting. It is perfectly appropriate to plan each component \*just\* before they code it. Students will invariably need to add (or possibly combine) components. When this happens, please encourage them to comment on their decomposition / add the extra components using a different colour.

In their documentation students should make notes / justify any major decisions / changes. They don’t need a ‘justification’ for every single component but there will be times where they should explain why they have done things a certain way. If they ask which way to do something, please encourage them to trial both ways and then choose the best option - they will want to take screenshots showing they have done this as it forms part of the evidence needed to get an M / E grade. The emphasis in the documentation is to show the breakdown and testing of the task. Justifications should be less than three sentences long! Most of the documentation will be taken up with large (hopefully easy to read) screenshots. Feel free to remind students that they can submit screencast / video evidence if they prefer.

The resource uses Python as the programming language. The first task (Lucky Unicorn) can be used to teach students the programming skills they will need to successfully complete the other tasks. There is a complete video walk through for that task. For the second task (Higher Lower game), the videos only show code that students might not have seen / used before. The hope is that they will be able to create at least some of the code themselves.

The tasks have been ordered by difficulty with the final task (car racer) being fairly challenging. My answer code for that task exceeds the standard in a number of ways. On the plus side, it is fairly efficient and works well. On the minus side, some of the functions use two dimensional arrays as I wanted to return two items and the only way to do this using a function is to return a list with more than one value. I’d anticipate student answers would be very different to those provided in the ‘model answer’ area. This is not a problem! It could lead to some useful discussion. Ultimately you might want to reassure students that the assessment task can be completed without needing to resort to advanced skills.

Provided students can successfully complete the third task (Rock / Paper / Scissors) they will be ready to attempt the assessment.

In the eBook I have put in several pages where I would expect students to submit program code and documentation. The underlying idea is that they should meet at least 3 of the 5 checkpoints and as teachers we should mark their work and give them feedback.