|  |  |  |
| --- | --- | --- |
| **Requirements for Achieved – AS91884**  **(1.8 Use Iterative processes):** | **ü** | **References / Examples** |
| **Use basic iterative processes to develop a digital outcome.** | | |
| The student:  Plans a digital outcome to address a problem, need, opportunity or interest. |  | Refer to Trello board and task decomposition slide (slide number 5)  Relevant Implications  Slides 3,4 |
| *For example (partial evidence):*  *The student identifies and describes the problem, need, opportunity or interest to be addressed.*  *The student lists the key steps to be taken in developing the required outcome.*  *The student identifies and describes any relevant rules or conventions to be followed during the process.* |  |  |
| Develops the digital outcome by decomposing the problem |  | Task Decomposition  Slides numbered 6, 7, 13,16, 22, 27, |
| *For example (partial evidence):*  *The student decomposed their basic computer game into the components that need to be developed and tested such as lists, functions, user interface, etc.* |  |  |
| Plans and trials components of the outcome in an iterative manner. |  | Test Plans  Slides 7, 8, 13, 17, 22, 26, 31  Test Results  Slides 9, 10, 14, 20, 23, 29 |
| *For example (partial evidence):*  *The student plans and tests the code for the quiz loop. They next plan and test the question and answer part of the quiz. They next plan and test the feedback part of the quiz. The check that quiz is easy to use and understand. Each component is planned and tested in an iterative manner until the final game quiz is produced.* |  |  |
| Tests that the programming outcome functions as intended |  | Assembled Outcome  Slides 33 – 39  End User Testing  Slide 42 |
| *For example (partial evidence):*  *The student plans to include testing the functionality of the quiz with various users to ensure the game works.* |  |  |
| Describes the relevant implications and end-user considerations that are relevant to the outcome. |  | End User Testing  Slide 42  Relevant Implications  Slides 3-4 and 40 |
| *For example (partial evidence):*  *The student spoke to students from their target audience to determine their quiz preferences. The student recognises that it is unethical to use copyrighted questions. They have recognised that quiz layout will affect the enjoyment of using the quiz. However, the student may not have chosen the best solution to address the considerations or could have more fully addressed these considerations.* |  |  |
| Examples provided above are indicative only. Final grades will be decided using professional judgement based on a holistic examination of the evidence provided against the criteria in the Assessment schedule | | |
| **Additional Comments:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirements for Achieved – AS91883**  **(1.7 Programming):** | **ü** | **References / Examples** |
| **The student has developed a computer program** | | |
| The student has written a simple, functional quiz program in the language of their choice. |  | **00\_MQ\_Base\_Final.py** |
| The program may not be structured very well but it:   * performs the specified task, using a suitable programming language * has variables which store at least two different data types * includes sequence, selection and iteration control structures * requires input either from a user or an external source * uses either   + data stored in at least 2 lists, arrays or dictionaries; or   + 2 user defined methods, procedures or functions   *Note that BOTH functions AND lists are not required to pass this standard.* |  | **00\_MQ\_Base\_Final.py** |
| *For example (partial evidence):*  *The student has written the code in a procedural manner, including use of loops, conditions and functions or indexed structures.* |  |  |
| Comments are present but may not be particularly descriptive of frequent. |  | All throughout that program / every python file  00\_MQ\_Base\_Final.py |
| *For example (partial evidence):*  *“# this code creates the quiz loop”* |  |  |
| Testing and debugging the program to ensure that it works on a sample of expected use cases. |  | Test Results  Slides 9, 10, 14, 20, 23, 29  Expected / Unexpected Cases  08\_Aim\_V4.py |
| The student has shown some evidence of expected use cases that were used to test and debug the program to show that the program works when the user inputs data that is expected.  Testing may be trial and error rather than clearly thought out. |  | Slides 10, 11, 15 |
| *For example (partial evidence):*  *The student has written about some expected use case data and has tested what happens when the program is run. Such testing may be observed by the teacher, presented in table form with minimal notes, or simply explained using examples.* |  |  |
| Examples provided above are indicative only. Final grades will be decided using professional judgement based on a holistic examination of the evidence provided against the criteria in the Assessment schedule | | |
| **Additional Comments:** | | |
| **Requirements for Merit – AS91884**  **(1.8 Use Iterative processes):** | **ü** | **References / Examples** |
| **Use basic iterative processes to develop an informed digital outcome.** | | |
| The student uses information from testing and trialling to improve the outcome. |  | Development and de-bugging  Slides 11, 15, 24, |
| *For example (partial evidence):*  *The student provides screenshots with a brief annotation that shows the improvements in the quiz mechanics that were made after making changes. They also provide a short video to demonstrate improved functionality after correcting a bug in the code.* |  |  |
| The student trials multiple components and/or techniques and selects those which ensure the outcome functions as intended. |  | Component Trialling  Slides 18, 19, 27, 28 |
| *For example (partial evidence):*  *The student trials two different techniques for performing a particular aspect of their quiz and selects the choice that does not cause functionality issues. The student trials two different question generating techniques and chooses the one that is most efficient.* |  |  |
| The student addresses the relevant implications and end-user considerations that are relevant to the outcome. |  | End User Testing  Slides 42  Relevant Implications Addressed  Slides 40  Final Discussion  Slide 44 |
| *For example (partial evidence):*  *The student addresses that fact that it is unethical to use copyrighted questions by making up their own questions / using creative commons material that has been correctly attributed. They also ensure that any images used in the quiz are copyright free. They have addressed usability and aesthetic considerations through testing their game with a range of end users.* |  |  |
| Examples provided above are indicative only. Final grades will be decided using professional judgement based on a holistic examination of the evidence provided against the criteria in the Assessment schedule | | |
| **Additional Comments:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirements for Merit – AS91883**  **(1.7 Programming):** | **ü** | **References / Examples** |
| **The student has developed an informed computer program** | | |
| Documenting the program with variable names and comments that describe code function and behaviour: |  | **00\_MQ\_Base\_Final.py** |
| * The student has frequent clear comments throughout the code that helps to describe relevant functions or sections of code |  | **00\_MQ\_Base\_Final.py** |
| * The variable names clearly describe the data they hold. |  | **00\_MQ\_Base\_Final.py** |
| *For example (partial evidence)*  *eg. # this function tests that the user input is a number. It will continue to ask for input until the input is a number.*  *e.g. answer = is\_a\_number(question)* |  |  |
| Following conventions of the chosen programming language:  The student has followed most common programming conventions for their chosen language. |  | Throughout all python files  **00\_MQ\_Base\_Final.py** |
| *For example (partial evidence) Student has:*   * *Python files and functions contain a docstring explaining the purpose of the program/function.* * *Constants are ALL\_CAPS with underscores separating the words if required.* * *Functions appear before loose lines of code and main section of code is all at bottom, not between the functions, thus making the program easier to read.* |  |  |
| Testing and debugging the program in an **organised** way to ensure that it works on a sample of both expected and relevant boundary use cases.  The student tests frequently during development (observed) and the final program works when the user inputs the data that is expected and checks or handles when the data is outside of specific thresholds.  The student may have kept some form of notes showing what was tested and the outcome of that testing.  Test use cases by student include expected and boundary use cases. |  | Testing  Slides 9, 10, 14, 20, 23, 29  Development and de-bugging  Slides 11, 15, 24,  Expected and Unexpected Cases  08\_Aim\_V4.py |
| *For example (partial evidence)*  *Student code has been tested to show that if there are boundaries (eg: choose a level between 1 and 3), code has been tested for 0 (too low), 1 (low valid), 3 (high valid) and 4 (too high).*  *The program might not correctly handle unexpected data and fail when a word is typed in where a number was expected.* |  |  |
| Examples provided above are indicative only. Final grades will be decided using professional judgement based on a holistic examination of the evidence provided against the criteria in the Assessment schedule | | |
| **Additional Comments:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirements for Excellence – AS91884**  **(1.8 Use Iterative processes):** | **ü** | **References / Examples** |
| **Use basic iterative processes to develop a refined digital outcome.** | | |
| The student has applied information from planning, testing and trialling of components to develop a high quality outcome |  | **Assembled Outcome**  **Slides 33-39**  **Final Code**  **00\_MQ\_Base\_Final.py**  **Development and de-bugging**  **Slides 11, 15, 24,**  **Testing**  **Slides 9, 10, 14, 20, 23, 29**  **Addressing of Relevant Implications**  **Slide 40**  **Component Decomposition**  **Slides 6, 12, 16, 21, 25, 30,**  **Task decomposition**  **Slide 5** |
| *For example (partial evidence):*  *The student has provided evidence that their planning has allowed them to meet project timelines and include all the planned for components and information. Their outcome functions as intended and has no obvious errors in functionality or presentation of the information. Evidence gained from trialling and thorough and organised testing has been integrated into the outcome in an ongoing manner to ensure the outcome is of high quality, including aesthetics, functionality and usability.* |  |  |
| Examples provided above are indicative only. Final grades will be decided using professional judgement based on a holistic examination of the evidence provided against the criteria in the Assessment schedule | | |
| **Additional Comments:** | | |

|  |  |  |
| --- | --- | --- |
| **Requirements for Excellence – AS91883**  **(1.7 Programming):** | **ü** | **References / Examples** |
| **The student has developed a refined computer program.** | | |
| The student ensured that the program is a well-structured, logical solution to the task |  | **This is shown in 00\_MQ\_Base\_Final.py**  These have all been shown in my final version and throughout all slides and python files |
| *For example (partial evidence):*  *The code is clean, concise, efficient and easily readable. The main program may be short and might consist of multiple reusable user defined functions which do most of the logic and processing.*  *The layout might include sections as follows (all the material might be in a loop to allow multiple questions to be asked until a condition is met)*   * *import modules (eg: import random)* * *user defined functions* * *set up constants* * *initialise variables* * *generate / ask question* * *calculate correct answer* * *get user answer and check it is valid* * *compare user answer with correct answer and update score / counters* * *give user feedback* |  |  |
| The student has made their program flexible and robust:   * The student has used methods, functions, procedures, actions and control structures effectively. * User input is checked to ensure that it is valid * Expected, boundary and invalid user input is handled correctly <see testing> * Constants, variables and derived values are used instead of hard coded values |  | * The use of methods, functions, procedures, actions and control structures effectively is shown in **00\_MQ\_Base\_Final.py** * User input using expected and unexpected cases are shown in **08\_Aim\_V4.py, 01\_Instructions\_V4.py, 04\_Question\_V5.py and 00\_MQ\_base\_Final.py.** It has also been shown in **development and testing** on **slides 9, 10, 11, 14, 15, 20, 24.** * **Variables** are shown throughout all code and python files as well as **00\_MQ\_Base\_Final.py** * Example of **Try and Except code** has been used in **08\_Aim\_V4.py** |
| *For example (partial evidence):*  *Student uses a series of if, elif, else statements rather than multiple, single ‘if’ statements . eg: to check a user answer where ‘x’ is a special exit code, their code might say…*  *if user\_ans == ‘x’:*  *do something*  *elif user\_ans == correct*  *do something*  *else*  *do something*  *rather than separate statements*  *if user\_ans == ‘x’:*  *break out of loop*  *if user\_ans == correct*  *do something*  *if user\_ans != correct*  *do something*  *Code handles expected and boundary values correctly (as for merit). In addition, it has also been tested for a decimal (eg: 2.5, invalid) and a string (eg: ‘two’, invalid). Students may have used ‘try / except’ code to ensure that their program handles invalid data gracefully.* |  |  |
| The student has comprehensively tested and debugged the program in an organised and time-effective way to ensure the program is correct on expected, boundary and invalid use cases.  The student tests their program in a systematic way to ensure that it works correctly for all logical pathways. |  | Testing Results  Slides 9, 10, 14, 20, 23, 29  Test Plans  Slides 7, 8, 13, 17, 22, 26, 31  Development and De-bugging  Slides 11, 15, 24,  Assembled Outcome  Slides 33-39 |
| *For example (partial evidence):*  *Test cases have been well-thought out and notes may have been made showing that the code works as expected for all use cases. The student has checked all the logical pathways for their program to ensure that it works as expected.* |  |  |
| Examples provided above are indicative only. Final grades will be decided using professional judgement based on a holistic examination of the evidence provided against the criteria in the Assessment schedule | | |
| **Additional Comments:** | | |