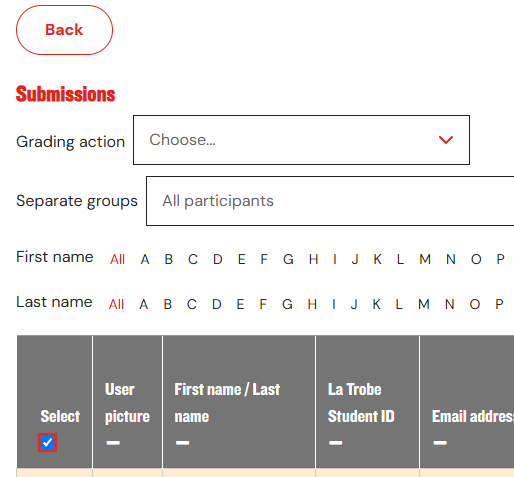
# Assignment Marking Instructions

**CSE1PE/CSE1PES, Semester 1, 2025**

## Getting ready to mark

1. Go to the “Assessments” > “Assessment 2: Details, Instructions, and Submission” > “View all submissions” page on LMS
2. Use the “Select” checkbox to select all students
   1. 
3. Scroll to the bottom of the page, select “With selected…” > “Download selected submissions” and click “Go”
   1. A screenshot of a computer

      Description automatically generated
4. Extract the downloaded ZIP folder to a location on your computer
5. Upload the unzipped folder of submissions to a folder in your Google Drive

## Marking procedure

1. Go to the “View all submissions” page on LMS, and set the options:
   1. “Filter” to “Submitted”
   2. “Workflow filter” to “Not marked”
2. Click the “Grade” button for the first student in the list
   1. You can make the grading area larger by collapsing the view:
      1. A screenshot of a computer

         Description automatically generated
      2. A screenshot of a computer

         Description automatically generated
3. In another window, go to Google Drive and find and open the student’s .ipynb file
4. Create a copy of the marking sheet template specifically for this student
   1. Either doing this locally on your PC, or by uploading to Google Drive
5. Mark the assignment by filling in the marking sheet (fill in all yellow cells)
   1. Edit the assignment sheet in either Microsoft Excel or Google Sheets
   2. Ensure you don’t accidentally give students more marks than the task is worth. The cell will become highlighted in red if this happens
6. Export the completed marking sheet as a PDF
   1. In Excel: File > Export > Create PDF/XPS
   2. In Google Sheets: File > Download > PDF
      1. You may need to set “Scale” to “Fit to width” if it goes beyond the width of the page
7. Go back to the LMS page
8. Enter “Grade out of 100”, upload the PDF as “Feedback files”, and change “Marking workflow state” to “In Review”
9. Click “Save and show next”

## Late submissions

* Ensure that you **include** weekends in the late day calculations (**Important:** This is different to the policy from last year)
  + E.g. for a student without an extension, submitting at 11:30PM on Sunday 18th May would be considered 2 days late and incur a 10% penalty
* Submissions that are more than 5 days late do not need to be marked
* Please allow a **1 hour grace** **period** for late submissions
  + Don’t apply a penalty to a student who submits < 1 hour late
  + Likewise, when considering the ‘days’ late. E.g. A student submitting at 12:30AM on Sunday should only be given a 1 day late penalty
* LMS has been updated to reflect students who have received an extension. When marking a student, look at the field at the top to determine if the student has submitted on time (it will be red if the student submitted late)
  + A green and black text

    Description automatically generated
* When marking, put the number of days late in the “Days late” field of the marking rubric. If it is not late, put 0 in that field
  + The marking sheet will automatically compute and use the late penalty when calculating the total mark

## Task 1: Academic Integrity reflection (5 marks)

* Mark this question generously, most students should be getting 5 marks.
  + Award 0 marks if the student has not attempted the question
* Award marks for responses to each question regardless of “correctness”:
  + 2 marks: Have any of Ioni, Rohan or Sami engaged in academic misconduct? Explain.
  + 1 mark: Have their choices led to them learning the subject material better?
  + 1 mark: What could be the possible outcomes for Ioni, Rohan and Sami now?
  + 1 mark: What could they have done differently?

### Task 2-7:

* Run the code cells
* Test with other inputs if you are not certain their solution meets the task criteria (see below).
* Mark according to criteria in the marking sheet. In the sample solution I’ve also highlighted which parts we are looking for to address each criteria (solutions from students may be more succinct).
  + If a solution does not work completely, marks can still be awarded for criteria that are met (e.g. if Task 2 has correct code for joining the list into a single string, then 2 marks should be awarded for that even if nothing else works)
  + Similarly, if a student has made an attempt at the criteria but did not quite get it working, partial marks for that criteria can be awarded (e.g. They wrote the joining code but had a syntax error stopping it for working, 1 mark could still be awarded for joining based on their attempt)
* Some students may add extra logic for validating user input beyond what is asked for in assignment criteria (e.g. handling more than 2x numbers passed into the !multiply command, etc.). This is not required but should not be penalised either.
* Task 5: Students have noted that the subject learning guide contains capitalisation of each word (e.g. ‘Algorithms and Flowcharts’), while the weekly topics are not all capitalised (e.g. ‘Algorithms and flowcharts’). When specifying the list constant, either of these two variations should be considered correct.
  + Be a little lenient with typos, one or two typos is okay
  + The words should otherwise match. Ensure students are using the correct names for each topic
  + Check that students have the correct number of topics (12) and the correct ordering. Partial marks should be awarded accordingly

|  |  |  |
| --- | --- | --- |
| Task | Input | Expected output |
| 2 | bot\_whisper(['HELP', 'Me']) | **Chatbot:** help me |
| 3 | bot\_multiply(['7.0', '3.301']) | **Chatbot:** 7.0 \* 3.301 = 23.1070 |
| 4 | bot\_count(['1000', '2']) | **Chatbot:** 1000  **Chatbot:** 1001  **Chatbot:** Sum: 2001 |
| 5 | bot\_topic(['Using', 'Modules']) | **Chatbot:** Topic 9: Using Modules |
| 6 | dispatch\_bot\_command('!multiply', ['1', '3']) | **Chatbot:** 1 \* 3 = 3.0000 |
| 7 | Hello  !whisper QUIET PLEASE!  !Topic Iteration  !QUIT | **You:**  Hello  **You:**  !shout QUIET PLEASE!  **Chatbot:** quiet please  **You:**  !Topic Iteration  **Chatbot:** Topic 4: Iteration  **You:**  !QUIT |

## Overall quality of submitted code (12 marks)

* Consider all attempted tasks
* This is where we separate technically correct but messy submissions from excellent submissions
* Award 0 marks if student did not write any code in their notebook submission

### Adherence to instructions (3 marks)

* Assessment of how well the student adhered to instructions, including beyond what is stated in task criteria
* This ranges from how well they followed the expectations in each task, with 3 marks being perfectly in each task, and 1 mark being they often did not follow the instructions

|  |  |
| --- | --- |
| Excellent (3) | The student demonstrated a complete understanding of what was asked in all the tasks that they attempted. |
| Good (2) | The student demonstrated a good understanding of what was asked for most of the tasks, with some oversights. |
| Poor (1) | The student frequently missed key details of task requirements. |

### Deliberate and intentional programming (3 marks)

* Assessment of how well the student understands what they have written
* Some things to look for:
  + Definition of variables that aren’t used
  + Expressions that don’t have their return values used
  + Variables that are update unnecessarily (e.g. x = 1 followed by x = 2 without x being used in between)
  + Leaving code in that does not serve any purpose

|  |  |
| --- | --- |
| Excellent (3) | Every line of code has a purpose for all tasks attempted. Commented-out lines of code are OK. |
| Good (2) | There are 1 – 3 lines of code that don’t do anything overall. |
| Poor (1) | Solutions frequently include lines of code that are redundant and unnecessary. |

### Consistency of coding style across solutions (3 marks)

* Penalise submissions that “feel wrong”, as if parts were written by different people/AI or patched together from other online resources

|  |  |
| --- | --- |
| Excellent (3) | All solutions look like they were written by one person with a very consistent coding style, using similar naming conventions, the same indentation levels, same spacing around operators, etc. |
| Good (2) | The solutions are written with a loose sense of consistency. The student follows certain style rules across the tasks but is inconsistent in other areas. |
| Poor (1) | There is very little consistency in the solutions, and the submission feels like a patchwork of snippets copied from different sources. |

### Readability and clarity of code (3 marks)

* There is intentionally some overlap with the consistency criterion. That is, a solution that is inconsistent can’t achieve full marks here

|  |  |
| --- | --- |
| Excellent (3) | Variables are named well, indentation is always 4 spaces, constants are always named *IN\_UPPER\_CASE*, variables are always named *in\_lower\_case*.  A submission with some unnecessary complexity but comments explaining how the code works can still be considered for full marks. |
| Good (2) | As above, but with a few occasional exceptions. Some algorithms are more complex than they need to be and are not properly explained. |
| Poor (1) | Python style conventions are frequently ignored, the code is often confusing and difficult to follow. |

## Feedback comments

* Written feedback to students is extremely important. Please ensure that every student receives feedback comments on the marking sheet
* Even if the student receives 100%, still leave a comment like: “Great work!” or “Well done!”
* If the student receives a high grade (e.g. 90%+), explain why they did not receive full marks (matching the marking allocation). E.g. “Great work! Recall that the convention for specifying constants in Python is to use ALL\_UPPERCASE\_LETTERS.”
* If the student has made a lot of mistakes, try to group and highlight common themes. E.g. “You use of lists in this assignment did not meet the task criteria. Please ensure to revisit data structures as part of your exam revision.”
* I recommend keeping a document of comments that you can reuse by copy-pasting.