# Assessment event 3 of 3: Project

## Criteria

### Unit code and name

ICTPRG302 | Apply introductory programming techniques

### Qualification/Course code and name

Select your Qualification/Course code and name from the dropdown.

ICT30120 | Certificate III in Information Technology (2)

## Student details

Student name

Daniel Ly

Student number

368263826

Version: 20230504

Date created: 4 May 2023

© TAFE NSW 2023  
RTO Provider Number 90003 | CRICOS Provider Code: 00591E

This assessment can be found in the TAFE NSW [Learning Bank](https://share.tafensw.edu.au/share/logon.do?.page=searching.do?in%3DC1b145167-45e0-41ec-9f64-92af668e3e54%26q%3D%26type%3Dstandard%26sort%3Drank%26dr%3DAFTER%26page%3D1).

The content in this document is copyright © TAFE NSW 2023 and should not be reproduced without the permission of TAFE NSW. Information contained in this document is correct at time of printing: 14 February 2025. For current information please refer to our website or your teacher or assessor as appropriate.

## Assessment instructions

Table 1 Assessment instructions

| **Assessment details** | **Instructions** |
| --- | --- |
| **Assessment event overview** | The aim of this assessment event is to assess your knowledge and performance in creating simple applications through introductory programming techniques.  This assessment is in 1 part:   * Part 1: Program debugging   And is supported by:   * A submission checklist * 1 Assessment Checklist * Assessment feedback   **Note**: This assessment may contain links to external resources. Access to the long URL is provided via the Error: Reference source not found section located at the end of this document. |
| **Unit assessment guide** | Refer to the unit assessment guide (UAG) before attempting this assessment event. The UAG contains information including assessment requirements and how to achieve a satisfactory result. |
| **Submission instructions** | When you complete this assessment:   * read the checklist at the end of the assessment to make sure you have completed everything * keep a copy of all the electronic and hardcopy assessments you submit to TAFE NSW * make sure you have completed the assessment declaration before you submit. |

# Specific task instructions

The instructions and the criteria in the tasks and activities below will be used by the Teacher/Assessor to determine if the student has satisfactorily completed this assessment event. Use these instructions as a guide to ensure the student demonstrates the required knowledge and skills.

## Scenario

As an ICT trainee with [Gelos Enterprises](https://share.tafensw.edu.au/share/items/d0b458dc-3922-409d-b1fe-9a2f785f4a38/0/GelosEnterprises.zip/index.html), the Software Development Team Leader, Christina Kaiser, has assigned you to undertake some debugging work on a program written by another trainee which is not functioning correctly.

## Part 1: Program debugging

Christina Kaiser has asked for your assistance with fixing a program written by another trainee, which is incorrect.

The trainee was asked to create an application that determines the average mark for a student based on their marks from five different subjects (see code below).

The instructions stated that the application must do the following:

* Ask the user to input the marks for the five subjects in a list.
* The program must ensure that all the marks are between 0 and 100 (no decimals).
* If any marks are invalid:
  + Display an appropriate error message.
* If all marks are valid:
  + Display the list of marks entered.
  + Find the sum of all the marks in the list (all five subjects) and display the output as:
    - The sum of your marks is: [sum]
  + Find the average of all the marks in the list (all five subjects) and display the output as:
    - The average of your marks is: [average mark]

Table 2 Code for marks program

| **Code for Marks Program** |
| --- |
| print("please enter your 5 marks below")  #read 5 inputs  mark1 = int(input("enter mark 1: "))  mark2 = int(input("enter mark 2: "))  mark3 = int(input("enter mark 3: "))  mark4 = int(input("enter mark 4: "))  mark5 = int(input("enter mark 5: "))  #create array/list with five marks  marksList = [mark1, mark2, mark3, mark4, mark5]  #print the array/list  print(marksList)  #calculate the sum and average  sumOfMarks = sum(marksList)  averageOfMarks = sum(marksList)/5  #display results  print("The sum of your marks is: "+str(sumOfMarks))  print("The average of your marks is: "+str(averageOfMarks)) |

#### Test data

Table 3 Test data

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | | | | | **Expected Output** | | |
| **mark1** | **mark2** | **mark3** | **mark4** | **mark5** | **Sum** | **Average** | **Comments** |
| 10 | 12 | 16 | 14 | 18 | 70 | 14 |  |
| 80 | 0 | 60 | 100 | 70 | 310 | 62 |  |
| 10 | A | *N/A* | *N/A* | *N/A* | *N/A* | *N/A* | Error message displayed. |
| 10 | 12 | 199 | *N/A* | *N/A* | *N/A* | *N/A* | Error message displayed. |
| 0 | 0024 | 99.99 | 100.0 | 1.2 | 225 | 45 | No Error messages. |

1. Create another test case using the samples above as a guide. Add this test data to the last row of the table above.
2. Use debugging and problem-solving techniques to detect and correct errors in the code (use the test cases to confirm the design specifications). Document your changes by including comments in the code using the correct syntax. Paste a copy of the code into the space provided.

Table 4 Corrected marks code

| **Corrected marks code** |
| --- |
| # Daniel Ly 14/02/2025 # This program determines the average mark for a student # based on their marks from five different subjects  # define a function that loops until a valid mark is entered def getvalidmark(prompt):  while True:  value = input(prompt)  try:  # make sure mark is an integer  mark = round(float(value))  # set boundaries for mark  if 0 <= mark <= 100:  return mark  else:  print(mark, "is not a valid mark!")  except ValueError:  print(value, "is not a valid mark!")   print("please enter your 5 marks below")  #read 5 inputs mark1 = getvalidmark("enter mark 1: ") mark2 = getvalidmark("enter mark 2: ") mark3 = getvalidmark("enter mark 3: ") mark4 = getvalidmark("enter mark 4: ") mark5 = getvalidmark("enter mark 5: ")  #create array/list with five marks marksList = [mark1, mark2, mark3, mark4, mark5]  #print the array/list print(marksList)  #calculate the sum and average sumOfMarks = sum(marksList) averageOfMarks = sum(marksList)/5  #display results print("The sum of your marks is: "+str(sumOfMarks)) print("The average of your marks is: "+str(averageOfMarks)) |

1. For each of the test data entries that generates an error, include labelled screenshots below of your debugging techniques. This must include examining the contents of the variables. (2 screenshots)

Table 5 Debugging screenshots

| **Debugging screenshots** |
| --- |
| Sample screenshot 1 |
| Sample screenshot 2 |

## Submission checklist

Submit the following for marking:

This completed document – Assessment event 3 of 3: Project

## Assessment checklist

The assessment checklist lists the **requirements for each task** in this assessment as outlined in the student's assessment instructions. The assessor will use this checklist to ensure **all** required tasks have been completed and submitted and provide feedback for each task.

Note that S = Satisfactory and U/S = Unsatisfactory.

Table 6 Checklist

| **Task number** | **Did the student do the following?** | **S** | **U/S** | **Assessor comments**  **Record your comments in enough detail to demonstrate your judgement of the student's performance against the criteria required.** |
| --- | --- | --- | --- | --- |
| **Part 1** | **Program debugging** |  |  |  |
| Task 1.1 | Student created another appropriate test case. | ☐ | ☐ |  |
| Task 1.2 | Student fixed code and added comments to document changes appropriately. |  |  |  |
| Task 1.3 | Student provided 2 labelled screenshots showing debugging techniques including examining variable contents. |  |  |  |

### Additional adhoc question/s asked by the Assessor

Additional ad hock questions may be asked by the assessor during or after the assessment event. This section provides the assessor opportunity to record these questions and your responses.

1. Assessor question (as required):

[Record your additional questions here]

Student response (as required):

[Record the student response/s]

1. Assessor question/s (as required):

[Record your additional questions here]

Student response/s (as required):

[Record the student response/s]

1. Assessor question/s (as required):

[Record your additional questions here]

Student response/s (as required):

[Record the student response/s]

## External resources – Links and URLs

Long URLs and permalinks are provided for access to content when the assessment is not used digitally, for example, not clickable.

Table 7 Long URLs

| **Resource Name** | **Long URL** |
| --- | --- |
| The Learning Bank | https://share.tafensw.edu.au/share/home.do |
| TAFE NSW Assessment Guidelines | https://share.tafensw.edu.au/share/items/d36df03f-9651-4d43-8c9d-a299699e8585/0/?attachment.uuid=30e52f91-8a9f-4df1-bf7f-91168307cfb9 |
| Gelos Enterprises | https://share.tafensw.edu.au/share/items/d0b458dc-3922-409d-b1fe-9a2f785f4a38/0/GelosEnterprises.zip/index.html |

This page is not required for online assessment submissions.

### Student assessment declaration

This assessment is my original work and has not been:

* copied from any source without proper referencing
* written for me by any other person except where such collaboration has been approved by a teacher or assessor.

Student signature and date

### Reasonable adjustment

Reasonable adjustment was in place for this assessment event.

If so, please provide details of any reasonable adjustment strategies that were implemented:

[Insert reasonable adjustment strategies]

### Assessment outcome

Satisfactory  Unsatisfactory

Comments

[Insert comments]

Assessor name, signature and date

Student acknowledgement of assessment outcome

[Would you like to make any comments about this assessment?]

Student name, signature and date