PROG2007 PROGRAMMING II

Summary

Title	Assessment 1
Type	Portfolio
Due Date	Monday 18th March 11:59 pm AEST/AEDT (start of Week 3)
Length	NA
Weighting	20%
Academic Integrity (See below for limits of use where GenAI is permitted)	GenAI MAY NOT be used for this assessment.
Submission	Please see the Submission section below on how to submit your assessment.
Unit Learning Outcomes	This assessment task maps to the following ULOs: ULO1: modify and expand short object-oriented programs that use standard control structures ULO2: design, implement, test, and debug simple programs in an object-oriented programming language ULO4: analyse and determine appropriate data structures and iteration methods to support a solution.

Rationale

No matter what field of IT you go into when you graduate from university, the ability to develop software in an object-oriented programming language is an important skill to have. This may seem obvious for students planning on a career in software development. However, it's also important for those students pursuing careers in fields like networking, cyber security, and data analysis. For networking, the ability to develop programs simplifies tasks such as device configuration and network analysis. In cybersecurity, it's used to develop programs for penetration testing, vulnerability scans, and automation of security incident responses. For data analysis, programming enables efficient data manipulation and analysis, making it an indispensable tool for extracting insights from large datasets.

Task Description

In this assignment, you will start to master a very important component of software development – classes and objects. The assignment contains three programming parts as follows:

- In part one you will modify and expand an existing program that requires one class.
- In part two you will modify and expand an existing object-oriented program that requires a minimum of two classes.
- In part three you will design and implement a simple object-oriented program that requires a minimum of three classes.

This assignment also requires you to create a video explaining why you completed the assignment the way you did.

You can use PyCharm or VSCode to complete the assignment and your assignment must run using Python 3. This is covered in the Getting Started section of MySCU.

However, when submitting your assignment:

- Given that your marker may be using a different IDE, you are only required to submit your Python files and not your project files.
- You are also required to use an industry standard plugin in your IDE called WakaTime and submit a screenshot of your WakaTime dashboard when submitting your assignment.

Both above points are covered in the submission instruction video associated with this assignment. You can find this video in Task 1 folder under the Assessment Task and Submission section. Please DO NOT start the assignment without watching the video.

Task Instructions

Part 1 – Simple class

This part of the assignment assesses:

- ULO1: modify and expand short object-oriented programs that use standard control structures.

In the final assignment of Programming I, you were provided with the following code that reads a csv file:

```python
import csv

with open("student_folder/playerCustomisation.csv", "r") as input_file:
    reader = csv.reader(input_file)
    for row in reader:
        print(row)

```

Your player number was based on the last digit of your student id. For example:

- A student with a student id 21367890 will be customising the game for Player 0
- A student with a student id 21370098 will be customising the game for Player 8

Based on your assigned player number (rows 0-9 in the file) you were required to read that row from the file and assign the *width* and *height*, number of *monsters* and *walls*, as well as the player's *animal* and *sound* to variables. You then had to create a video and discuss and explain how to use a class to store the data.

In this assignment, you will create and use that class to store the data. You are required to:

- Design a simple class that is suitable for storing the data associated with your assigned player in the file. The file is available in Task 1 folder under the Assessment Task and Submission section.
- Update the code provided above that reads the file. Instead of storing the values associated with your player number in variables, you must create an instance of the class and use it to store the data associated with your assigned player in the file.
- Output the object's attributes formatted for readability.

You should have a good idea on how you will implement the class for this part of the assessment based on the video you made in Programming 1. While designing the class, think about the following:

- What attributes should this class have?
- Does it require any methods?
- How can you secure and validate the data in the class?

Part 2 – Shopping cart system

This part of the assignment assesses:

- ULO1: modify and expand short object-oriented programs that use standard control structures.
- ULO4: analyse and determine appropriate data structures and iteration methods to support a solution.

All online shops require a shopping cart system where users can add items, remove items, and view their cart contents.

Imagine that another programmer has started writing the code to implement a shopping cart system. However, they are unable to finish, and you must complete the program. You have been provided with the following Product class.

```python
class Product:
    def __init__(self, product_id, name, description, price):
        """
        Initialize a new Product instance.

        :param product_id: Unique identifier for the product (int or str)
        :param name: Name of the product (str)
        :param description: Description of the product (str)
        :param price: Price of the product (float or int)
        """
        self.product_id = product_id
        self.name = name
        self.description = description
        self.price = price

    def display_product_info(self):
        """
        Display the product's information.
        """
        print(f"Product ID: {self.product_id}")
        print(f"Name: {self.name}")
        print(f"Description: {self.description}")
        print(f"Price: ${self.price:.2f}")

```

You have also been provided with the following class requirements for the program.

Product class

You can make any changes that you think are necessary to the Product class. While doing so, consider the following questions:

- Are any other attributes required? If so, should they be object attributes or class attributes?
- What other methods would be useful for the product class?
- How can you secure and validate the data in the Product class?

Shopping Cart class

To complete the program, you will need a shopping cart class. The shopping cart class must keep track of products a user is planning on buying and have the ability to add products, remove products, and display all of the products in the shopping cart. When programming the shopping cart class consider the following questions:

- How are you going to store the products the user is planning on buying (you learned the concepts required for this in Programming 1).
- What attributes should this class have?

- Does the class require any default parameters?
- What methods should this class have?
- How can you secure and validate the data in the shopping cart class?

Other classes
You are not required to have any other classes in the program. However, if you want to have additional classes you are more than welcome to do so.

Testing
To complete this part of the assignment you are required to write the code that demonstrates that you can add a product to the shopping cart, remove the product from the shopping cart, and view the shopping cart contents.

Hints
You have learned all the skills you require to complete this part of the assessment in:
- Programming 1.
- In module one and the first half of module 2 (up until the section on inheritance)

Make sure that you have completed the final activities of module 1 and module 2 as all of the concepts required to complete this part of the assessments are demonstrated in the video content in these sections.

Part 3 – Student management system

This part of the assignment assesses:
- ULO2: design, implement, test, and debug simple programs in an object-oriented programming language.
- ULO4: analyse and determine appropriate data structures and iteration methods to support a solution.

Imagine you are part of a team that is writing a student management system for the University. Your job is to write the component of the student management system that tracks the following:
- Students.
- Units.
- Degrees.

Your program has the following requirements:
- Add students, remove students, and view students in a unit.
- Add units, remove units, and view units in a degree.

Hints
- This part is very similar to Part 2 with one additional class.
- While you should make this program as realistic as possible you do not need to track every attribute that a student, unit, or degree would have in the real world.

Testing
To complete this part of the assignment you are required to write the code that demonstrates that you can:
- Add a student to a unit, remove a student from a unit, and view all students in the unit.
- Add a unit to a degree, remove a unit from a degree, and view all units in the degree.

Hints
Just like in part 2 of this assignment, make sure that you have completed the final activities of module 1 and module 2 as all the concepts required to complete this part of the assessments are demonstrated in the video content in these sections.

Part 4 - Video

You are required to create a video explaining why you completed the assignment the way that you did. Your video should address each part of the assignment separately i.e. cover part one first, then part two, then part three. You are not required to explain your code line by line. Rather for each part of the assessment your video should focus on the following:

- The attributes in each class and why you chose them.
- The methods in each class and what they do.
- Any techniques you've employed to secure and validate your data.
- Any design considerations you made when designing your classes.
- The code you wrote to demonstrate your classes, what it does, and why you wrote it the way you did.

Resources

Everything that you need to know to complete this assessment was covered in:

- Programming I
- Module 1 and the first half of module 2 (up until the section on inheritance) in programming 2.

Task Submission

As mentioned at the start of this assessment you can use PyCharm or VSCode to complete the assignment and your assignment must run using Python 3. This is covered in the Getting Started section of MySCU.

However, when submitting your assignment:

- Given your marker maybe using a different IDE you are only required to submit your Python files and not your project files.
- You are also required to use an industry standard plugin in your IDE called WakaTime and submit a screenshot of your WakaTime dashboard when submitting your assignment.

Both above points are covered in the video submission video associated with this assignment. Please DO NOT start the assignment without watching the video.

The video covers how to submit your Python files, your screenshot of the WakaTIme dashboard, and your video.

Please note that all submission instructions in this assignment and the submission video must be followed EXACTLY, including the folder names you are instructed to use. Failure to do so will result in a requirement to resubmit. The reason for this is as a programmer, you will often work as part of a team and will be required to follow design documentation. If the design parameters are not followed precisely, bugs will be introduced into the software when all of the individual components of the program are assembled.

Academic Integrity

Please note the following points:

- Your source code for this assignment will be run through a plagiarism detection system designed for code that compares all assignments and highlights identical or very similar submissions. If you are found to have colluded with other students, you will be submitted for academic integrity. Test submissions generated using GenAI software will also be included in the source code comparison to pick up those assessments that were programmed using GenAI.
- If your marker deems your submission suspicious, you may be asked to attend an interview in your tutorial class to explain your code. You may be submitted for academic integrity if you cannot explain your code. Possible reasons your submission may be deemed suspicious could include:
    - Using programming concepts not taught in the unit.
    - Using programming concepts considered by your marker to be beyond your programming abilities as demonstrated in the class.
    - Submitting code suspected of being generated using GenAI software.

At Southern Cross University, academic integrity means behaving with the values of honesty, fairness, trustworthiness, courage, responsibility and respect in relation to academic work.

The Southern Cross University Academic Integrity Framework aims to develop a holistic, systematic and consistent approach to addressing academic integrity across the entire University. For more information, see: SCU Academic Integrity Framework

NOTE: **Academic Integrity breaches include** unacceptable use of generative artificial intelligence (GenAI) tools, the use of GenAI has not been appropriately acknowledged or is beyond the acceptable limit as defined in the Assessment, poor referencing, not identifying direct quotations correctly, close paraphrasing, plagiarism, recycling, misrepresentation, collusion, cheating, contract cheating, fabricating information.

GenAI May Not be Used

Generative artificial intelligence (GenAI) tools, such as ChatGPT, **may not be used** for this assessment task. You are required to demonstrate that you have developed the unit's skills and knowledge without the support of GenAI. If you use GenAI tools in your assessment task, it may result in an academic integrity breach against you, as described in the Student Academic and Non-Academic Misconduct Rules, Section 3.

Special Consideration

Please refer to the Special Consideration section of Policy.
https://policies.scu.edu.au/document/view-current.php?id=140

Late Submissions & Penalties

Please refer to the Late Submission & Penalties section of Policy.
https://policies.scu.edu.au/view.current.php?id=00255

Grades & Feedback

Assessments that have been submitted by the due date will receive an SCU grade. Grades and feedback will be posted to the 'Grades and Feedback' section on the Blackboard unit site. Please allow 7 days for marks to be posted.

Assessment Rubric

Marking Criteria and % allocation	High Distinction (85–100%)	Distinction (75–84%)	Credit (65–74%)	Pass (50–64%)	Fail 0–49%
Enhancing object-oriented programs (Parts 1 and 2) (ULO1) 20%	Demonstrates exceptional proficiency in **enhancing an existing object-oriented program** utilizing all of the concepts covered in Module 1 and the first half of Module 2.	Demonstrates advanced expertise in **enhancing an existing object-oriented program**, applying key concepts from Module 1 and early Module 2, with minor areas for improvement.	Shows solid understanding and skill in **enhancing an existing object-oriented program** with Module 1 and early Module 2 concepts, despite some inconsistencies.	Displays basic skill in **enhancing an existing object-oriented program** using foundational concepts from Module 1 and early Module 2, though the application is somewhat shallow and incomplete.	Shows insufficient understanding and application of Module 1 and early Module 2 concepts in **enhancing an existing object-oriented program**, indicating a fundamental implementation shortfall.
Design object-oriented programs (Parts 2 and 3) (ULO2) 30%	Demonstrates exceptional proficiency in **designing and implementing an object-oriented program** utilizing all of the concepts covered in Module 1 and the first half of Module 2.	Demonstrates advanced expertise in **designing and implementing an object-oriented program**, applying key concepts from Module 1 and early Module 2, with minor areas for improvement.	Shows solid understanding and skill in **designing and implementing an object-oriented program** with Module 1 and early Module 2 concepts, despite some inconsistencies.	Displays basic skill in **designing and implementing an object-oriented program** using foundational concepts from Module 1 and early Module 2, though the application is somewhat shallow and incomplete.	Shows insufficient understanding and application of Module 1 and early Module 2 concepts in **designing and implementing an object-oriented program**, indicating a fundamental implementation shortfall.
Data Structures and Methods (Parts 2 and 3) (ULO4) 30%	Demonstrates exceptional proficiency in using data structures and programming associated methods to	Demonstrates advanced expertise in using data structures and programming associated methods to	Shows solid understanding and skill in using data structures and programming associated methods to	Displays basic skill in using data structures and programming associated methods to manipulate the data	Shows insufficient understanding in using data structures and programming associated methods to

	manipulate the data structure to meet the programs requirements.	manipulate the data structure to meet the programs requirements.	manipulate the data structure to meet the programs requirements.	structure to meet the programs requirements.	manipulate the data structure to meet the programs requirements.
Explain Design and Implementation Decisions in Video (ULO1 ,2, 4) 20%	Provides an exceptionally clear, comprehensive, and insightful explanation of design and implementation decisions, demonstrating advanced understanding and reasoning that significantly enhances the assignment's objectives.	Provides a very detailed and coherent explanation of design and implementation choices, showing a high level of understanding and analytical thinking with minor areas for further clarification.	Explains design and implementation decisions clearly communicating the rationale behind approaches with some room for deeper analysis or clarity.	Adequately explains design and implementation decisions, covering basic rationales and justifications but with some gaps in clarity or detail.	Fails to provide a coherent explanation of design and implementation decisions, with significant gaps in understanding or the ability to articulate the reasoning behind approaches.

Southern Cross University

Description of SCU Grades

High Distinction:

The student's performance, in addition to satisfying all of the basic learning requirements, demonstrates distinctive insight and ability in researching, analysing and applying relevant skills and concepts, and shows exceptional ability to synthesise, integrate and evaluate knowledge. The student's performance could be described as outstanding in relation to the learning requirements specified.

Distinction:

The student's performance, in addition to satisfying all of the basic learning requirements, demonstrates distinctive insight and ability in researching, analysing and applying relevant skills and concepts, and shows a well-developed ability to synthesise, integrate and evaluate knowledge. The student's performance could be described as distinguished in relation to the learning requirements specified.

Credit:

The student's performance, in addition to satisfying all of the basic learning requirements specified, demonstrates insight and ability in researching, analysing and applying relevant skills and concepts. The student's performance could be described as competent in relation to the learning requirements specified.

Pass:

The student's performance satisfies all of the basic learning requirements specified and provides a sound basis for proceeding to higher-level studies in the subject area. The student's performance could be described as satisfactory in relation to the learning requirements specified.

Fail:

The student's performance fails to satisfy the learning requirements specified.