**Computer Science**

**ATAR Year Twelve**

# Task 1: OOP Business Client Application

## Assessment type:

Project

## Conditions:

Period allows for the completion of the task: 5 weeks

Recommended time allocation  
Part 1: 2 weeks for the completion of the investigation and design of the project

Part 2: 3 weeks for the development and evaluation of the project

## Task weighting:

15% of the school mark for this pair of units

## Scenario

Your school is looking to create more opportunities for students in the drama and theatre space. As part of this, there will be an increase in the number of small productions to engage the community. It will also allow students to demonstrate their talent.

The current drama space/black box theatre can seat 80 people when configured with the stage. The school would like to be able to sell tickets for individual seats in the theatre. Currently people that are booking tickets call the school and reception takes their order and then emails them details of their ticket.

The school would like to assist the receptionist to make this process smoother, by having an application that can visualise and show how many tickets are remaining, what seats are still free, take bookings (including for multiple tickets) and then create a receipt and ticket that can be emailed to the customer.

The system needs to factor in different ticket prices for concession holders and students, and should make use of object-oriented programming concepts, with the use of two-dimensional arrays, and good programming practices.

## Overview:

Your task is to use the software development framework to design and create a simple software application in Python that will help your friend in the Community Store.

To successfully develop the application, you will need to make use of:

* Program control structures.
* Two-dimensional arrays.
* A modular approach using functions and parameters, and argument passing.
* Programmatically writing data from Python to a text file.
* Object oriented programming principles including creation and use of classes.

Your final project needs to be complex enough to demonstrate your understanding of software development.

### Part 1

#### Investigate

* Break down the steps required to produce your software project and develop a timeline for when each of these steps needs to be completed. You may like to use Trello or any other suitable method.
* Problem outline
  + Write a brief outline of the purpose of the software solution and its objectives.
* Problem description
  + Write a detailed description of the application that includes:
    - A written description of the scenario you will need to develop a solution.
    - A list of requirements that your solution needs to include to be useful, including programming requirements.
    - Recognition of any ethical, legal, and security implications for development of the program.

#### Design

* Using pseudocode, write an algorithm showing the core logic for the application to show how main components will work. Note: this algorithm will not resemble your final, fully functioning code. It should simply demonstrate how the core logic for the application will work.
* Outline using structure charts and structure design an overview of the program to assist development.

### Part 2

#### Develop

* Using a modular approach, create your program using Python. Ensure you use good programming practices these should include:
  + Validate input before processing
  + Use of meaningful variable names
  + Use constants for readability and maintenance
  + Use of comments to explain code
  + Appropriate use of standard control structures
  + Use of appropriate indentation and white space
  + One logical task per module
  + Use of stubs
  + Version control and regular backups
  + Meaningful names for modules
  + Exception handling
  + Functions are able to return a single data structure or value.
* Ensure that you are using your GitHub repository or other version control as appropriate.
* Develop and document the use of test data for your program to ensure (and demonstrate) you can effectively debug.
* This documentation should include appropriate test data, type and range checks. You should document this test plan in a table outlining the input, the expected output and what actually happened.

#### Evaluate

* Reflect on the success of your solution and how well it meets the system requirements.
* Verify that the application meets the requirements and expectations of end-users. Consider the opening scenario and conduct testing that is representative of how a user would be operating the application. You might want to get your peers to test your program and provide feedback.
* Take notes of any problems, documenting bugs, and prioritise what you would need to improve.
* Perform a developer retrospective and reflect on the process you followed to develop your solution and how you could improve this process. Some aspects to consider include:
  + What worked well?
  + What didn’t work well?
  + What you would do differently next time to improve the development process?
* Document the sources you used to get information about how to develop your system, including all websites and textbooks.

### Submission requirements

#### Part 1

For Part 1, you are required to produce a single, well-formatted Word document. This document should include:

* A cover page
* Suitable headings to make each section clear
* Headers and footers
* Appropriate terminology, explanations, and written expression
* Diagrams created using appropriate software

#### Part 2

* A single folder that includes your entire project, including text files that may be required to run your program.
* A well-formatted Word document as per the points above for your evaluation.

Submit all documentation including all files required to run your program as part of a GitHub repository, ensuring permissions are set correctly.

Task authentication – you are required to prove that the work you complete is your own. Do this by using a GitHub repository with regular updates to your code, or screenshots showing the progress of development. Save these with your other project files. You may also be asked to provide a verbal explanation of your project and how it works.

**Marking Key:**

| **Description** | **Marks** |
| --- | --- |
| **Development Schedule** | |
| Breaks down the project planning into a series of meaningful steps and a realistic timeline for completing each step has been included | 2 |
| Breaks down the project into a limited series of steps with some attempt at showing a timeline. | 1 |
| **Subtotal** | **/2** |
| **Problem Outline** | |
| Accurately outlines of the purpose of the software based on the design brief/scenario. | 3 |
| Mostly outlines the purpose of the software based on the design brief/scenario. | 2 |
| Partially outlines the purpose of the software based on the design brief/scenario. | 1 |
| **Subtotal** | **/3** |
| **Problem Description** | |
| Provides a clear and detailed explanation of the application, its structure, how it will handle different requests, and user interactions. | 5 |
| Clearly explains of the application, its structure, how it will handle different requests, and user interactions. | 4 |
| Describes the application, its structure, how it will handle requests, with reference to some user interaction. | 3 |
| Gives a limited description of application with some reference to how it is structured/user interaction. | 2 |
| Gives a limited description of application that is unclear and/or incomplete. | 1 |
| **Subtotal** | **/5** |
| **Requirements** | |
| Provides a clear and detailed list of requirements that fully meet the needs of the problem description. Suitably classifies requirements. | 4 |
| Completes a list of requirements that meet the needs of the problem description. Classifies requirements. | 3 |
| Provides a list of requirements that mostly meet the needs of the problem description. Partially classifies requirements. | 2 |
| Provides an incomplete list of requirements that meet some of the needs of the problem description. Makes a limited attempt at classifying requirements | 1 |
| **Subtotal** | **/4** |
| **Algorithms** | |
| Provides a completed algorithm in pseudocode that provides a reasonable representation of the core application logic, using correct symbols and/or syntax. | 5 |
| Provides a completed algorithm in pseudocode that provides a representation of the core application logic, using correct symbols and/or syntax. | 4 |
| Provides a mostly complete algorithm in pseudocode that provides a partial representation of the core application logic, with symbols and/or syntax that is mostly correct. | 3 |
| Provides an algorithm with a partial solution using the application, with some errors in syntax, logic and/or symbols being used | 2 |
| Provides a partially correct algorithm and/or uses incorrect symbols/syntax. | 1 |
| **Subtotal** | **/5** |
| Develops an accurate algorithm that contains no logic errors, and demonstrates the use of a range of control structures | 3 |
| Develops an accurate algorithm that may contain minor logic errors, and uses a range of control structures. | 2 |
| Develops an algorithm that contains logic errors, and uses a minimal range of control structures. | 1 |
| **Subtotal** | **/3** |
| **Total Part 1** | **/22** |

| **Description** | **Marks** |
| --- | --- |
| **Use of programming structures** | |
| Makes consistent and appropriate use of a variety of control structures. Uses data types appropriately, including effective use of arrays, constants, variables, selection and iteration. | 5 |
| Makes appropriate use of a variety of control structures, including selection and iteration. Mostly uses data types appropriately with constants and variables. | 4 |
| Makes use of a variety of control structures, although may not use most appropriate structures at times. Attempts to make appropriate use of a variety of data types for variables, including some use of arrays. | 3 |
| Attempts to use a variety of control structures such as selection and/or iteration. Makes limited use of data types with variables and some attempt at using arrays appropriately. | 2 |
| Makes minimal use of selection and iteration in code, with inappropriate use of different data types. Provides arrays that are not used, or are used inappropriately, and do not serve required purpose. | 1 |
| **Subtotal** | **/5** |
| **Object oriented programming** |  |
| Makes consistent and appropriate use of OOP principles including classes with appropriate structures, attributes, and methods. | 5 |
| Mostly uses OOP principles including classes with structures, attributes, and methods. | 4 |
| Uses some OOP principles including classes, with structures, attributes, and methods. | 3 |
| Attempts to use OOP principles including classes. | 2 |
| Makes minimal use of OOP principles, or does not use any OOP within the project. | 1 |
| **Subtotal** | **/5** |
| **Good programming practice** | |
| Appropriately structures code, making effective use of modularisation and parameter passing with appropriate naming conventions and use of white space. | 5 |
| Mostly structures code appropriately, making use of modularisation and parameter passing with use of appropriate naming conventions and white space. | 4 |
| Creates simplistic code using modularisation with some use of parameter passing. Mostly uses appropriate naming conventions with some use of white space. | 3 |
| Attempts to make code and use of modularisation with limited consideration of parameters. Uses some naming conventions used, although these may be inconsistent. | 2 |
| Produces poorly structured code that makes minimal or no use of modularisation. Uses naming conventions throughout the code are inconsistent and/or not meaningful. | 1 |
| **Subtotal** | **/5** |
| Uses accurate and useful comments throughout the code to explain the purpose of modules where necessary. | 3 |
| Uses comments that help make code readable. | 2 |
| Makes limited use of comments throughout code. | 1 |
| **Subtotal** | **/3** |
| **Functionality** | |
| Develops an effective and efficient program with minimal bugs. | 4 |
| Develops an effective program, but may contain some bugs. | 3 |
| Develops a program with a significant number of bugs. | 2 |
| Partially completes program implementing minimal system requirements. | 1 |
| **Subtotal** | **/4** |
| **Test Plan** | |
| Completes a detailed test plan and documents testing of the program. Considers all possible inputs and user interactions with the program. | 5 |
| Completes a test plan and documents testing of the program. Considers possible inputs and user interactions with the program. | 4 |
| Completes test plan and documents testing of the program. Considers some possible inputs and user interactions with the program. | 3 |
| Completes and documents a partial test plan. Considers limited inputs and user interactions with the program. | 2 |
| Develops a minimal test plan and/or provides minimal documentation of program testing. | 1 |
| **Subtotal** | **/5** |
| **Testing and review** | |
| Demonstrates a detailed evaluation of how the program meets the requirements identified in Part 1, including discussion of the user experience. | 5 |
| Evaluates how the program meets the system requirements , with discussion of the user experience. | 4 |
| Completes a partial evaluation of how the program meets the requirements, with a superficial discussion of the user experience. | 3 |
| Completes a limited evaluation of how the program meets the system requirements. | 2 |
| Completes a superficial evaluation of the program and how it meets the system requirements. | 1 |
| **Subtotal** | **/5** |
| Provides a detailed discussion of how the final product could be improved and documents any bugs and/or limitations. | 5 |
| Describes bugs and/or limitations with reference to how the final product could be improved. | 4 |
| Identifies bugs and/or limitations, without reference to their impact on the final product. | 3 |
| Attempts to identify bugs and/or limitations, with no or limited discussion. | 2 |
| **Subtotal** | **/4** |
| **Retrospective** | |
| Completes a detailed evaluation of the development process and suggests future impacts. | 3 |
| Completes an evaluation of the development process that was used including some suggested future impacts. | 2 |
| Completes a minimal evaluation of the development process with superficial comments on development process used and suggested future impacts. | 1 |
| **Subtotal** | **3** |
| **Total Part 2** | **/39** |
| **Total** | **/61** |