For this project, you will create a text-based computer game. It is recommended that you

use an existing game as the basis for your program. It is considered a culminating activity so that you can demonstrate the python programming skills you mastered throughout the semester. You are to work individually on this project using any and all resources at your disposal, but the work must be your own. Any cases of plagiarism in any way will result in a mark of zero. You may also create your own game, but this option has the added responsibility of (a) the creative process to make a reasonable game, and (b) producing documentation for the game itself.

The following programming concepts must be used **correctly** and **appropriately** in your program:

* Variables & Constants
* Input & Output
* Selection (if )
* Lists
* Arithmetic Operators & Expressions
* Formatting of output (e.g. rounding, spacing)
* Repetition (counted & conditional loops)
* Modularity (Methods)

Make sure to choose a program that meets your capabilities. Though the instructor will offer some help during the project development, it is expected that the student fully completes the coding themselves.

You should strive to use as many concepts as possible to show your understanding and ability to apply them in an application. **Your project must be approved by the teacher (as soon as possible)**.

# Step 1: Choose your program (Problem Definition / Program Description)

* Brainstorm final program ideas (search online, look over the resources provided)
* Narrow your brainstorm down to three ideas to choose from.
* When you have selected a final topic obtain approval from the teacher for that project.

# Step 2: Project Proposal & Planning (outline the program requirements/specifications)

Submit a **1 to 2-page** written report with the following sections (sub-titles):

### Program Description (Problem Definition)

* + detailed description of the application (program) that you plan to be create

### Analysis & Design

* + A flow chart describing all of the inputs, processes, and outputs of your program
  + a list of all knowns (constants) & unknowns (variables) required in your program

### Programming Concepts

* + outline the programming concepts you plan to use (e.g. if, for, while, do-while, lists, files, other topics you plan to research yourself, etc.) and how you plan/intend to use them
  + describe/explain how you plan to use these concepts in your application
  + list of major methods/functions you will use/create to show programming

### Research

* + concepts you research

# Completion of Steps 1 & 2 must be no later than June 7 (earlier if possible)

### Submit the document to google classroom.

**Step 3: Design & Implementation**

1. Design your program
2. Begin coding (implementing your program) in Python
   * Remember to keep your target dates in mind for particular items/steps
   * Document all along the way – NOT only at the end of coding

### IMPORTANT:

Create a Google Doc called ***FirstNameICS3Uculminating*** and share it with your teacher.

At the end of each class, COPY and PASTE your current program to the Google Doc, replacing any previous content. Google Docs keeps all revisions. I suggest you write your methods first (step-by-step design). Even if you only make minor progress, copy and paste your code into the google doc each day. You only need to share it once (the first time). I can see all revisions after that.

# Step 4: Testing, Maintenance, & Documentation

* Include detailed internal documentation in your code (i.e. comments explaining each class, method, loop, lists, logic, etc.)
* Document as you proceed – NOT only at the end of your programming
* Each part should have comments describing the purpose, parameters, and any return values
* Create a **User Manual or User Guide** for the project that explains how to use the application and its features and dropped off with the rest of your project.
* Ensure that all proper documentation and programming conventions are followed (e.g. spacing, indentation, naming of identifiers – variable and method names).
* Include an appropriate comment header at the top of every python file.

# Evaluation

* Your final project is **25% of your final grade** in this course (see rubric for details).

### After completing your code you need to make a short presentation explaining your code/game. Your face must be shown in the video initially and then you may focus on the code. This is worth 5% of your grade

**FINAL SUBMISSION →**

**Saving & Submitting Your Project**

* save all code files, supplementary resource files, and documentation files (including the user guide/manual)
* Name all files appropriately (descriptive, concise)
* Submit the work to google classroom

**Projects Ideas:**

Games involving dice, cards, letters, and words provide good candidates. The following list provides some examples of games that are suitable for this project.

* + Yahtzee
  + Hangman / Wheel of Fortune
  + Jeopardy
  + Trivia/Quiz Game
  + Pig Dice
  + 7/11
  + Mastermind
  + BlackJack

Or any program that applies lists, functions and loops. Work at your level.

Some games or concepts will require additional research beyond what you have covered in the course. Some games are well-suited to the use of two- dimensional arrays. It is important that you select a program/game that is within your capabilities.

## Name:

**ICS3U Culminating Project**

**/ 100**



**ICS3U – CULMINATING PROJECT RUBRIC (20% OF FINAL MARK)**

**THINKING – Program Analysis & Design (Project Proposal & Planning)**

| **Level 0**  **(<50%)** | **Level 1**  **(50 – 59%)** | **Level 2**  **(60 – 69%)** | **Level 3**  **(70 – 79%)** | **Level 4**  **(80 – 100%)** |
| --- | --- | --- | --- | --- |
| Program design proposal not provided.  No programming concepts outlined or explained.  Implementation steps not followed. | Program design proposal is minimal.  Few programming concepts outlined and explained.  Implementation steps minimally followed.  Little to no analysis & problem-solving skills shown. | Program design proposal has some missing parts/steps  Some programming concepts outlined and explained.  Some implementation steps were followed. Some analysis & problem- solving skills shown. | Program design proposal is clear and complete (includes all major features/ steps).  Most programming concepts outlined and explained well.  Most implementation steps were followed. Considerable analysis & problem-solving skills shown. | Program design proposal is clear, complete, concise, comprehensive, and modular.  All programming concepts outlined and explained thoroughly.  All implementation steps show excellent analysis & problem- solving skills. |

**/ 20**

**APPLICATION – Program Implementation & Functionality, Code Structure & Design**

| **Level 0**  **(<50%)**  The program is not provided. | **Level 1**  **(50 – 59%)** | **Level 2**  **(60 – 69%)** | **Level 3**  **(70 – 79%)** | **Level 4**  **(80 – 100%)** |
| --- | --- | --- | --- | --- |
| Incomplete. Program is missing some major features and/or has significant errors.  Demonstrates few required programming skills & concepts. | Somewhat complete. Program is missing some minor features and/or has some minor errors.  Demonstrates some required programming concepts & skills. | Mostly complete. Program has some minor errors and/or omissions.  Demonstrates most required programming concepts & skills. | Complete and passes all tests with no errors.  Demonstrates exceptional understanding of all programming concepts & skills.  Exceeds expectations. |

**/ 50**

**COMMUNICATION – Code Readability, Organization, Documentation & Maintenance**

| **Level 0**  **(<50%)** | **Level 1**  **(50 – 59%)** | **Level 2**  **(60 – 69%)** | **Level 3**  **(70 – 79%)** | **Level 4**  **(80 – 100%)** |
| --- | --- | --- | --- | --- |
| No formatting or comments provided.  Methods/variables not used or naming incorrect, not descriptive.  Documentation is not provided. | Minimal formatting and comments provided.  Method/variable naming incorrect, not descriptive. Documentation is minimal/incomplete. Many spelling and/or grammar errors interfere with  understanding. | Some formatting and comments provided. Method/variable naming somewhat correct, descriptive. Documentation is missing some parts. Some spelling and/or grammar errors interfere with understanding. | Good formatting and comments provided. Method/variable naming mostly correct, descriptive. Documentation is mostly complete.  Minor spelling and/or grammar errors do not interfere with understanding. | Excellent formatting and comments provided.  Method/variable naming meaningful and descriptive.  Documentation is exceptional and detailed. No spelling or grammar errors. |

**/ 20**

**Knowledge –Organization, Documentation , appropriate variable names**

| **Level 0**  **(<50%)** | **Level 1**  **(50 – 59%)** | **Level 2**  **(60 – 69%)** | **Level 3**  **(70 – 79%)** | **Level 4**  **(80 – 100%)** |
| --- | --- | --- | --- | --- |
| No formatting or comments provided.  Methods/variables not used or naming incorrect, not descriptive.  Documentation is not provided. | Minimal formatting and comments provided.  Method/variable naming incorrect, not descriptive. Documentation is minimal/incomplete. Many spelling and/or grammar errors interfere with  understanding. | Some formatting and comments provided. Method/variable naming somewhat correct, descriptive. Documentation is missing some parts. Some spelling and/or grammar errors interfere with understanding. | Good formatting and comments provided. Method/variable naming mostly correct, descriptive. Documentation is mostly complete.  Minor spelling and/or grammar errors do not interfere with understanding. | Excellent formatting and comments provided.  Method/variable naming meaningful and descriptive.  Documentation is exceptional and detailed. No spelling or grammar errors. |

**/ 10**