ENGR 102 **Assigned Nov 21, Due Dec 7**
Team Project **Report [250] + Presentation (graded separately)**
Submission: Submit all files you created, including py-files to Canvas. One submission per team.

This project is meant to be open-ended, requiring team participation and incorporating all the aspects of the course we have covered so far. This project will count for the equivalent of 2 homework/lab assignments (i.e. about 2 weeks' worth of regular work). (presentation is graded separately)

Your team should put together a program that lets a person play a 2-person turn-based (think board-style) game, either against another person or against the computer. Examples of the game you choose could range from the very simple (tic-tac-toe) to the much more complex (chess). Other examples of games could include Connect 4, Reversi (Othello), Mancala, Checkers, Dots and Boxes, etc.

Other options:

- **A text adventure game** based on a popular movie or book. For inspiration, have a look at A Dark Room. It is a very poplar, mostly text-based game.
  A complete text game will let users move through rooms based on user input and get descriptions of each room. To create this, you'll need to establish the directions in which the user can move, a way to track how far the user has moved (and, therefore which room he/she is in), and to print out a description. You'll also need to set limits for how far the user can move. In other words, create "walls" around the rooms that tell the user, "You can't move further in this direction."

The tricky parts here will involve setting up the directions and keeping track of just how far the user has "walked" in the game. I suggest sticking to just a few basic descriptions or rooms, perhaps 6 at most. This project also continues to build on using user inputted data. It can be a relatively basic game, but if you want to build this into a vast, complex word, the coding will get substantially harder, especially if you want your user to start interacting with actual objects within the game.

- **Guess 4 numbers** (cows and bulls) with Hangman to limit the amount of guesses.
  The Goal: The program will first randomly generate 4 numbers unknown to the user. The user needs to guess what that numbers are and their positions. (In other words, the user needs to be able to input information.) After the user's guess, the program should return some sort of indication (e.g. The number is correct – it is a caw, the number is correct, and the position is correct it is a bull). You'll need functions to check if the user input is an actual number, to see the difference between the inputted number and the randomly generated numbers, and to then compare the numbers. You may use a visual indication for the user how many guesses he/she did and how may is left like in a Hangman game.
- **TIC TOC TOI GAME** – is good example, but I am providing you a solution for that, so you cannot use it directly and submit it as your project.
  If you love to play computer games, then you can develop a Python game as your first python app. Games become cooler if having a good user interface so you can create a good UI and implement the concept of tic toc toi.

  https://www.tutorialspoint.com/artificial_intelligence_with_python/artificial_intelligence_with_python_gaming.htm

- **MAGIC 8 BALL**
  I believe you used magic 8 balls in your life. So now you can implement as a game and show you skill to others using Python. You can make it more attractive by using your creativity. Here you ask a question and see the answer in floating die. You can add very attractive answers here.
- **A SUDOKU GAME**
  Features:
  - User registration & login
  - Game Registration
  - Different Complexity
  - Timer
  - Hint Display
  - Error Checking
  - Solution Display
    Custom Themes
- **MAGIC SQUARE SOLVER IN PYTHON**
  Magic Square Solver is game which I developed in Python in features are given below
  ***Features*:**
  - Read magic square puzzle from an input file
  - Solve the puzzle using one of the various methods
  - Display the time taken for solving

Bingo (https://en.wikipedia.org/wiki/Bingo_(American_version))
Connect Four (https://en.wikipedia.org/wiki/Connect_Four)

You may code your own game, not listed here.

For your project you don't have to, but you may use idea of AI with Python and build aka Chatbot.
Chatbots are our new love. Chatbots are the new beginning. Chatbots are the new apps. However, everything you should know is that chatbots are new assistants that provide different services via chatting. Chatbots are a type of AI. To be more specific, chatbots are ANI, artificial narrow intelligence. They are not as clever as humans. Besides, chatbots can carry out a limited amount of tasks. Nevertheless, these functions still make our lives easier. There are many ways to do that.

If you want to create artificial intelligence chatbots in Python, you'll need AIML package (Artificial Intelligence Markup Language). First of all, create a standard startup file with on pattern. Load aiml b. Add random responses that make a dialog interesting. Now to write your own AIML file, browse for some files you already may use. For example, search among AIML files from the Alice Bot website. Enter Python.

When you create the startup file, it will then serve as a separate entity. Thus, you may have more AIML files without source code modifications. The program will start learning when there are many AIML files. Speed up the brain load. Add Python commands. So that's an introduction to how you can make artificial intelligence using Python.

The tutorial about AI in Python is here
https://www.tutorialspoint.com/artificial_intelligence_with_python/index.htm

AI with Python – Gaming is here
https://www.tutorialspoint.com/artificial_intelligence_with_python/artificial_intelligence_with_python_gaming.htm

How to Install PyGame and Create a Template for Developing Games in Python 3
https://www.digitalocean.com/community/tutorials/how-to-install-pygame-and-create-a-template-for-developing-games-in-python-3#updating-the-display

Project requirements

The project has some basic requirements. You will earn more points for going beyond the minimum requirements, including handling more advanced games.
You don't have to but you may want to learn to use a very simple 2D graphics library, like Gosu, Processing, or Pygame, and it will open up a whole new world of possibilities.

Requirements

Here are the minimum requirements your games should have:

- [5 points] There should be some sort of display of the board itself each turn. This can be text output to the screen if you wish. [+5 points for good graphics, if pictures are presents]
- [5points] (for grading: rules of the game should be stated clearly) The rules of the game should be enforced. For example, if your team chose to implement chess, then you should ensure that each piece can only make valid moves (and that all valid moves are possible). Also, you should be able to detect and report when there is a winner (or if the game ends tied).
- [5 points] You should allow the user to select whether they are playing a 2-person game or playing against the computer. For a 2-person game, just alternate asking for moves.
- The computer should, at a minimum, make a valid move if the user is playing against the computer. The minimum requirements are just that the computer always makes some valid move if one is available. [this is graded during the presentation]
- You should include try-except blocks to handle user input, and possibly other aspects of your program. That is, you should ensure that users enter valid moves, using try-except to catch cases when they might enter their move in an invalid way.
    - If for some reason your input is not amenable to using try-except statements, you can use try-except statements in different areas of your code, but they should be included to catch exceptional situations.
- You must provide 3 documents and Py – package.
    - [50] Document 1: is **technical documentation** for developers like you. In this document you should include the technical information like what you used, and how. You should give at least 5 examples of methods/ techniques you implemented from the class lectures, for example: [5 point per example = total 25 points]
        - Top-down and/or Bottom-up design processes
        - Incremental (pyramid-style) development
        - Functions, modules, loops, and so on…
        - Using methodical debugging approaches to find and handle bugs

      [25points] In this package, you should include a full set of tests, and ensuring the program meets all tests. You may include any other useful information. You can picture if you are selling your program, what would you include here?
    - [100] Document 2: **Design Document and Work Statement.** Statement of group work for your team's project, where you include code of cooperation [10 points] and a table with time effort/task load table per team member [10 points]. Here you include a history [10 points] of your code developing: debugging, changing of algorithms, changing ideas and goals. More details what to include her see below.
    - [100] Document 3: A user manual for a player where you explain the rules of the game.

The minimum requirements will account for 80% of the overall grade. Earning more than 80% will require adding additional functionality or features beyond the minimum. Examples of additional functionality (you are not limited to these – these are just some ideas) could include:

- [10 points for Using AI] A more advanced game (i.e. chess is more complex than a basic game like Connect 4). Games with randomness (like Backgammon) would also be more advanced. A game like ordinary tic-tac-toe would lose some credit in this category.
- Computer AI behavior. Making "smarter" moves, etc. Even making a random move from the set of all available moves is a slight improvement over the minimum. Somehow trying to determine a "best" move is even better.
- [10 points] A win/loss record-keeping system, high score, etc.
- [10 points] A better user interface (e.g. a graphical user interface)

You are certainly not expected to include all of these; the idea is that some of these improvements can be included.

You may use external modules to assist in your program. However, be sure to use the following guidelines regarding external modules:

- Be sure that the module is publicly available (e.g. via PyPi) and works with Python 3
- Be sure that license restrictions do not prevent you from using the module)
- Realize that the module should not solve the problem for you. For instance, a "Checkers game" module would not be acceptable to use in your checkers game. A "Checkers AI" module that lets a computer play checkers would be OK to include, and you would get some bonus for including that over the minimum, but not as much as if you came up with your own computer program.
- Clearly document in your code (and in your final document) what you used, and how.

Approach

Your team should begin by choosing a game that you would like to implement.
(At this moment do not forget to start taking notes to document what you are doing. You may delegate one team member to be responsible for that.)
You should use the techniques we have discussed in class when designing and building your program. This includes:

- Top-down and/or Bottom-up design processes
- Incremental (pyramid-style) development
- Generating a full set of tests, and ensuring the program meets all tests
- Using methodical debugging approaches to find and handle bugs.

Note: it is critical that you approach this task using an incremental development approach. That is, ensure that one (small) part of your program is working before adding another, small, part on. It is very common when people face a task like this to try to "bite off more than you can chew" – imagining a program with far more complexity and features than is feasible to implement in a given time frame. If you follow an incremental development approach, you should be able to ensure that you always have a working, bug-free program.

To help ensure that your team gives sufficient thought to the project, you will need to provide a design description stated below.

When working as a team, it is common for different team members to assume different roles and responsibilities for the project. Not every person needs to write the same amount of code, but each person should contribute equally to the project. For example, one team member might focus primarily on writing test cases for your team's functions, another might focus only on the interface, another might do less coding but handle all the design documentation and ensure that everyone else on the team is doing their part, etc. You are free to organize your team however you see fit, but you will need to document everyone's role at the end.

Please note that your team will not be able to sit together during class time to develop most of this program – the actual programming will need to be done outside of class time.

Design Document and Work Statement

Your team should put together a design document (Document 2, part 1), laying out the way that your program will be set up. This should include the following:

- The game you will be using.
- An initial design breakdown.
    - You should follow a top-down and/or bottom-up design process to determine what functionality you want to implement. You should end up with a list of core functions, each broken down
    - [10 points. List of modules, variables, functions, any of those] Also, your team should describe any "important" variables you will use and how they will be defined. An example of an "important" variable would be one holding the information about where pieces are on the board at a particular time. An example of an "unimportant" variable (that you don't need to include in the design document) would be an iterator for some loop in your program.
- An initial plan of work: what responsibilities does each team member have on the project, and what is the timeline you expect to follow.

After your team has finished your program, your team should put together **a statement of work** (still Document 2, part 2). In this document, you should do the following:

- [10 points] List what was accomplished from your original plan
- [10 points] List any items not accomplished from your original plan
- [10 points] List any external modules (if any) that you included (outside of the Python Standard Library) that someone would need to install to run your program.
    - Provide a very short description of how that library was used in your code.
- [10 points] Provide a brief (1-3 paragraph) summary of your team's work. What problems did you encounter along the way, and what incorrect assumptions did you have that led you to be able to accomplish less or more than originally planned?
- [10 points, if a team member is not listed or did not contribute, deduct these 5 points or do not give any points at all for the project] Provide a list of what each team member contributed to the project. Be specific: it is not enough to say "team member X helped code". You should identify exactly which parts each person did on the project.
- [10 points] Provide a brief (1-2 paragraph) "reflection" statement. Given what you now know from the project, how would you have approached it differently from the beginning?
- [5 points] Pagination of each page within the Design Notebook
-

Oral Presentation
On the final class day your team should be prepared to present a **brief** overview of your project in class. During this overview, you should be prepared to state:

- What game you are implementing
- A brief demonstration of your program working
- A summary of any "extras" you added on to your code.

Your presentation should take a **maximum** of 5 minutes. It is OK to have a significantly shorter presentation. Make it attractive to buyers!

Dates/Schedule
To understand the remaining time that you will have available to work on this project, here are the remaining dates:

- *November 21* (M) – Project Assigned
- *November 28 -* Regular Classes: we will have both a lecture topic and possible Quiz. Your team will have some time to work on the project during the class.
    - It is recommended that your team complete your design document this day, including assigning any work you expect team members to do in the interim.
    - You should also determine if (and if so, when) your team will want to meet or check-in with each other during the remainder of the project. You will probably want to ensure you all have a way of communicating and sharing code since there will not be much class time to meet to work on it.
- *November 22*
    - Regula class, exam review
    - Your team should have completed any design documents by this point, and all team members should have clear tasks to work on (after the final exam) at this point.
- *Nov 30 W* – Final Exam (during class time)
    - Your team will not have class time available to meet, here.
- *December 5 M*
    - Regular day
- *December 7* (W) - This is a REDEFINED day – students attend their Monday classes. The class will meet this day.
    - This is the final class meeting.
    - Each team will give a short oral presentation this day (if time permits).
    - Your team should expect to have some **short** time to work together this day. You should expect to use this time to work on your summary/reflection document.
    - You should turn in your final project no later than midnight on this date. This includes the source code, user manual, the design document, and the reflection document.

Grading:
See inside of the document.