*Your submission for this assignment **must include your full name** and your nine-digit **student number as a comment** at the top of the **source file you submit**. All source code files must be written using the **Python 3 programming language** and must run on the course's **official virtual machine**.*

***Submissions that crash** (i.e., terminate with an error) on execution will **receive a mark of 0**.*

Officially, the Due Date for this Assignment is:

***Friday, November 5th, 2021**, at **11:59pm EST**.*

Late Submissions are Accepted Without Penalty Until Sunday, November 7th, by 11:59pm EST.
Submissions received after that will not be accepted and will receive a mark of 0.

For this assignment you will design and implement a simple board game, and then animate a random playthrough of that game by two computer-controlled opponents. This will allow you the opportunity to work with more complex nested looping structures, and will also help you recognize the need for functions (coming soon to a future lecture, but not required for this assignment).

In order to complete this task, you will need to:

- choose the grid[1] for your game board from these options: 42, 56, 72, 90, 110, 132, 156
- choose how many and what type of "dice" will be used[2] to play your game
- choose and implement at least two of the board game "features" from Table 1, below

Your submission for this assignment:

- must be a source code file with filename 'comp1405_f21_#########_assignment_06.py'
- must use a WHILE loop for the game, terminating when either player reaches the end
- must draw the grid for your board game using pygame nested FOR loops
- must draw the players on the board and use variables to store the location of each player
- must alternate "turns" between the two players
- must roll "dice" (using random.randint) each iteration, for the player whose "turn" it is

Table 1. Board Game "Features" to Consider

"Ladder Connections" reaching certain squares advances player to others	"Snake Connections" reaching certain squares returns player to others	"Numbered Tiles" all squares numbered using a pygame.font	"Double Moves" permit a double move on certain conditions
"Extra Turns" grant an extra turn on certain conditions	"Missed Turns" force a lost turn on certain conditions	"Exact Requirements" players can't move past the end of the board	"Sorry Collisions" if player lands on other, send it back to start

...other features might also be acceptable - contact your instructor if you have any ideas!

[1] Please note that the available options are all "pronic numbers", meaning they can be easily arranged on a rectangular grid

[2] For example, you might use two 6-sided dice (i.e., 2d6), or one 10-sided die (i.e., 1d10), or three 4-sided dice, etc.