- ❖ You should write a `main()` function to prompt the user for inputs. Ask for a lower bound, an upper bound, and a function name. Then, your main function should call `summation()` using the given inputs and print out the result.
- ❖ You may assume the upper and lower bounds of the summation will be integers, but make sure you don't divide by zero ;)
- ❖ You may assume that a valid function name out of the three will be entered
- ❖ However, the parameter specifying the function to be summed must not be case-sensitive. 'Cube', 'INVERSE', and 'squArE' should all be valid user inputs
- ❖ You are not allowed to import any modules for this problem

An example program output is shown below:

```
Enter a lower bound for summation: 2
Enter an upper bound for summation: 5
Enter a function to be summed (square, cube, inverse): Inverse
The result of the summation is 1.2833333333333332
```

Once you're satisfied with your work, save your program as a source file named grayx501_3C.py, where "grayx501" is replaced by the x500 ID from the first part of your student email address. Put the file in a folder named hw3 (i.e. the path of the file should be /hw3/grayx501_3C.py). Then push your code to the remote GitHub server.

Problem D: Tic Tac Toe (10 points)

Tic Tac Toe is a two player game played on a 3 x 3 board. Players take turns placing a representative marking ('x' for one player, 'o' for the other) on an unoccupied space on the board until either one player makes a three in a row line (in which case that player wins), or the board fills up (in which case the game ends in a tie). A three in a row can mean any of the following:

- The player fills up all three slots in one row of the board
- The player fills up all three slots in one column of the board
- The player fills up either of the two diagonals

Your task is to write a Python implementation of Tic Tac Toe using three lists of length three to represent the board. The lists should be initialized to contain only empty strings at the start, and each of two players should be prompted to repeatedly give a space on the board (0-8) to add a mark to. Your program should then alter the string at that space to 'x' for player 1 and 'o' for player 2, according to the following indexing scheme:

```
[0, 1, 2]
[3, 4, 5]
[6, 7, 8]
```

Print the game board to the screen after each player's move. Make sure to check if the space a player chooses is already occupied, then print an error message and reprompt them to enter another space if it is. Your game should terminate when either every position on the board is filled up (in which you should print a message to the liking of "it's a tie!') or one player fulfills the winning criteria for three in a row (print "player 1 wins!" or "player 2 wins!", respectively). An example game is shown below:

```
Player 1, where would you like to move? 0
['x', ' ', ' ']
[' ', ' ', ' ']
[' ', ' ', ' ']
Player 2, where would you like to move? 1
['x', 'o', ' ']
[' ', ' ', ' ']
[' ', ' ', ' ']
Player 1, where would you like to move? 4
['x', 'o', ' ']
[' ', 'x', ' ']
[' ', ' ', ' ']
Player 2, where would you like to move? 0
Sorry, that space is already taken
['x', 'o', ' ']
[' ', 'x', ' ']
[' ', ' ', ' ']
Player 2, where would you like to move? 2
['x', 'o', 'o']
[' ', 'x', ' ']
[' ', ' ', ' ']
Player 1, where would you like to move? 8
['x', 'o', 'o']
[' ', 'x', ' ']
[' ', ' ', 'x']
Player 1 wins!
```

Constraints/Comments

- You may assume that the user(s) will always enter an integer from 0-8
- You must implement the Tic Tac Toe board using three Python lists, as shown above
- You must account for all possible victory states for either player: three in a row, three in a column, and the two diagonals, plus the tie case where the board is full without a winner

- ❖ Your game must ask for input from each player, alternating to the other player after each **valid** move. Reprompt players who try to place a mark on an already-occupied space. (HINT: loops will be your friends!)
- ❖ Your entire code must be contained within functions, with one call to `main()` at the end to start the game off
- ❖ You are not allowed to import any modules for this problem

Upon completion, save your program as a source file named grayx501_3D.py, where "grayx501" is replaced by the x500 ID from the first part of your student email address. Put the file in a folder named hw3 (i.e. the path of the file should be /hw3/grayx501_3D.py). Then push your code to the remote GitHub server.

Optional Bonus Problem: Tic Tac Toe Solitaire

Tic Tac Toe is fun and all, but you don't always have an opponent to play against. Sitting bored and playing yourself for the 62nd time, you suddenly get the idea to create your own player two! Using your extensive knowledge of conditionals and loops, you decide to code an AI to play Tic Tac Toe with you, so you'll never be bored or lonely again.

Your task is to design an algorithm for a computer opponent to implement instead of player two in your solution of Problem 3D. This opponent should have some sort of logic behind its moves involving conditionals or otherwise. For example, you might check if player 1 has a two in a row and then move to block them. Have fun with it!

An example interaction might be as follows:

```
Player 1, where would you like to move? 1
['x', ' ', ' ']
[' ', ' ', ' ']
[' ', ' ', ' ']
It's time for my turn
['x', 'o', ' ']
[' ', ' ', ' ']
[' ', ' ', ' ']
Player 1, where would you like to move? 5
['x', 'o', ' ']
[' ', 'x', ' ']
[' ', ' ', ' ']
It's time for my turn
['x', 'o', ' ']
[' ', 'x', ' ']
```

```
[' ', ' ', 'o']
Player 1, where would you like to move? 4
['x', 'o', ' ']
['x', 'x', ' ']
[' ', ' ', 'o']
It's time for my turn
['x', 'o', ' ']
['x', 'x', ' ']
[' ', 'o', 'o']
Player 1, where would you like to move? 7
['x', 'o', ' ']
['x', 'x', ' ']
['x', 'o', 'o']
You win again?!
```

Once you finally beat your AI opponent, save your program as a source file named grayx501_Bonus3.py, where “grayx501” is replaced by the x500 ID from the first part of your student email address. Put the file in a folder named hw3 (i.e. the path of the file should be /hw3/grayx501_Bonus3.py). Then push your code to the remote GitHub server.