Practice with 2d lists

For the examples below, assume you have some 2d lists like this:

```
matrix = [[1, 3, 5], [2, 4, 6], [3, 6, 9]]
matrix2 = [[5, 2, 8, 4], [-9, 0, 4, 1], [5, 6, 4, 8]]
```

1. Yesterday we saw two ways to create a function add up all the numbers on the upper-left to lower-right diagonal of a square matrix (a matrix with the same number of rows and columns). The key is that all the numbers on this diagonal have the property that their *row index is equal to their column index*.

   So one way is to use an if-test:

   ```
   def add_diagonal(grid):
     total = 0
     for row in range(0, len(grid)):
       for col in range(0, len(grid[0])):
         if row == col:
           total = total + grid[row][col]
     return total
   ```

   But this code is inefficient, because it wastes time by looping over large chunks of the matrix that we know don't matter (numbers not on the diagonal). Because there's a mathematical relationship (row == col), we can remove the nested loops and just use one loop:

   ```
   def add_diagonal(grid):
     total = 0
     for row in range(0, len(grid)):
       total = total + grid[row][row]
     return total
   ```

2. Write a function that adds up numbers on the upper-right to lower-left diagonal. Hint: Figure out the mathematical relationship between the numbers on this diagonal; there is a similar relationship to the one in problem #1.

3. Write a function to change each *odd* number in a matrix by multiplying it by 2 (the original matrix should be altered; don't create a new matrix).

   ```
   def mult2odd(grid):
   ```

4. Write a function to change all the numbers in *odd rows* of a matrix by multiplying them by 2 (the original matrix should be altered; don't create a new matrix).

   ```
   def mult2OddRows(grid):
   ```

5. Write a function to print the sum of each row of a matrix.

   ```
   def print_sum_each_row(grid):
   ```

   Example: `print_sum_each_row(matrix)` would print 9, 12, 18. (printing one number per line is fine)

   Challenge: Change this function so instead of printing the answer, it returns a list of these sums. E.g.: [9, 12, 18]

6. Write a function to print the sum of each column of a matrix.

   ```
   def print_sum_each_col(grid):
   ```

   Example: `print_sum_each_col(matrix)` would print 6, 13, 20. (printing one number per line is fine)

   Challenge: Change this function so instead of printing the answer, it returns a list of these sums. E.g.: [6, 13, 20]

7. Write a function to print the smallest number in each row of a matrix.

   `def print_smallest_in_row(grid):`

   Example: `print_smallest_in_row(matrix2)` would print 2, -9, 4.

8. Write a function to print the smallest number in each column of a matrix.

   `def print_smallest_in_col(grid):`

   Example: `print_smallest_in_col(matrix2)` would print -9, 0, 4, 1.

9. Write a piece of code that creates a 10 by 10 multiplication table in a grid. Hint: One idea is to start by using the function on the 2d list handout to create a 10 by 10 grid of zeroes, and then use nested for loops to change each element to its proper number.

10. Challenges: change the print smallest/largest functions to return lists of the smallest/largest items in each row/column, rather than printing them. So problem 5 would return the list [2, -9, 4].