Two Dimensional Lists

Tables are often used in math. An example of a table of numbers with 3 rows and 4 columns (ie. a 3 by 4 table) is below:

2	4	7	6
5	8	1	0
7	8	4	2

While some programming languages (Turing) allow the programmer to easily create tables, Python does not. In Python, a table needs to be simulated by creating a two-dimensional list - a list which contains a series of other lists. For example, the table above would be initialized as:

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

To output the bolded **1** in the two-dimensional list above, the command would be:

```
print ( twoDList [1] [2] )      # Outputs the 3rd element (index of 2) of the 2nd list (index of 1)
                                # Remember that the first row and column have an index of 0
```

Creating and Ouputing Two-Dimensional Lists

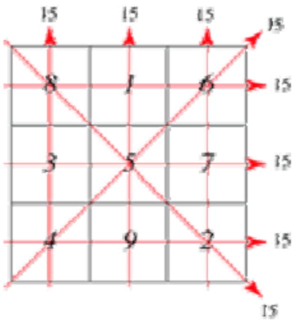
```
'''Allows the user to input data to create a 2D list and have it outputted as a 3 by 4 table'''

twoD_List = []      # Initializes the list which will become a 2D list

for rowNum in range (0,3):

    row = []            # Initializes the list that will store the values in each row

    for colNum in range (0,4):
        row.append (int (input ("Enter integer: "))) # Creates the “lists within the list”

    twoD_List.append (row)      # Appends the row list to the 2D list (ie. a list in a list)

print ("The entire list is", twoD_List)
print ("Element in row 1 column 2 is", twoD_List [1] [2])

print ("The list outputted as a 3 by 4 table is")

for rowNum in range (0,3):          # Outputs the “rows” values

    for colNum in range (0,4):      # Outputs the “column” values

        print (str (twoD_List [rowNum] [colNum]).rjust (3), end ="")

    print ()      # Required to move the output to the next line
```

Programming Assignment

1. Create a program that will ask the user to input the number of rows and columns they want in a table. It will then allow the user to input a series of integers and have an appropriately sized table outputted. Call your program **TwoDimList1.py**

2. a) Create a program that will allow the user to input a 3 by 3 table and output whether it is an “easy” magic square. An “easy” magic square is a table of numbers consisting of positive integers arranged such that the sum of the n numbers in any horizontal, vertical, or main diagonal line is always the same number. The integers can be repeated. Call your program **TwoDimList2Easy.py**

b) Create a program that will allow the user to input a 3 by 3 table and output whether it is an “true” magic square. An magic square is a table of numbers consisting of the **distinct** positive integers 1, 2, ..., n^2 arranged such that the sum of the n numbers in any horizontal, vertical, or main diagonal line is always the same number (see diagram on right). No integers are repeated. Call your program **TwoDimList2True.py**



8	1	6
3	5	7
4	9	2

c) Improve your program so the user can input a table of any dimensions and output whether it is a magic square. Re-save your program as **TwoDimList2True.py**

3. a) Rewrite the program above so it uses functions. To accomplish this, you will need to:

- Create a pure (one-result) function that will accept an N by N two-dimentinal list and return whether it is a magic square. A magic square is a table of numbers consisting of the distinct positive integers 1, 2, ..., n^2 arranged such that the sum of the n numbers in any horizontal, vertical, or main diagonal line is always the same number (see diagram above).

- Create a void function that will allow the user to input an N by N two dimentional list (a table) and pass the list to the main program.

- Create another void function that will accept a two-dimentional list and output whether it is a magic square. Note it will the pure function created above.

- Create a mainline to call the functions

Call your program **TwoDimList3.py**

b) Why can a function be used to input lists but could not be used in the Grade 11 Review Assignment to input variables? If you are not sure, watch the following videos:

- Mutable vs. Non-Mutable Data Types Video
- Mutable vs. Non-Mutable Parameters Video