# Cheat Sheet

### Rectangular two-dimensional lists (represents a grid of values)

name = [[],[]] #inner brackets pairs are the height of the 2d list

#the number of elements inside inner

#bracket pairs are the width

name[row][**col**] = value;

Example:

|  |  |  |  |
| --- | --- | --- | --- |
|  | *0* | *1* | *2* |
| *0* | 0 | 0 | 0 |
| *1* | 0 | 0 | 42 |
| *2* | 0 | 5 | 0 |
| *3* | 23 | 0 | 0 |

numbers = [[0]\*3,[0]\*3,[0]\*3,[0]\*3] # 4 x 3 list

numbers[1][2] = 42 # row: 1, column: 2

numbers[2][1] = 5 # row: 2, column: 1

numbers[3][0] = 23 # row: 3, column: 0

name = [[value, value, ...], [value, value, ...], ...]

Example:

|  |  |  |  |
| --- | --- | --- | --- |
|  | *0* | *1* | *2* |
| *0* | "x" | "o" | "x" |
| *1* | "o" | "x" | "o" |
| *2* | "x" | "o" | "x" |

grid = [["x", "o", "x"],

["o", "x", "o"],

["x", "o", "x"]]

### Two-dimensional lists traversal

for r in range(len(grid)):

for c in range(len(grid[r])):

**do something with grid[r][c]**

----------------------------------------

for row in grid:

for col in row:

**do something with element col**

NOTE: grid[r][c] and col are the same value,

these are two ways to access the same value

Examples:

for r in range(len(grid)):

for c in range(len(grid[r])):

print(grid[r][c], end="")

print()

----------------------------------------

for row in grid:

for col in row:

print(col, end="")

print()

### Two-dimensional lists notes

* Two-dimensional lists are just lists of lists (rows).
* In rectangular two-dimensional list, the lengths of all rows are the same, so len(m[0]) represents the length of any of the rows.

1. **Problems**

### Two-dimensional list simulation

**1.** What are the contents of numbers after the following code is executed?

numbers = [[0]\*4, [0]\*4, [0]\*4]

for r in range(len(numbers)):

for c in range(len(numbers[r])):

numbers[r][c] = r + c

**2.** Consider the following method:

def mystery(numbers):

for r in range(len(numbers)):

for c in range(len(numbers[r]) - 1):

if (numbers[r][c + 1] > numbers[r][c]):

numbers[r][c] = numbers[r][c + 1]

If a two-dimensional list numbers is initialized to

3 4 5 6

4 5 6 7

5 6 7 8

what are its contents after the call mystery(numbers)?

### Two-dimensional list programming

**3.** Assume that a two-dimensional rectangular list of integers called matrix has been declared with six rows and eight columns. Write a loop to copy the contents of the second column (at index 1) to the fifth column (index 4).

**4.** Write a function called matrix\_add that accepts a pair of two-dimensional lists of integers as parameters, treats the lists as two-dimensional matrixes, and returns their sum. The sum of two matrixes A and B is a matrix C, where for every row i and column j, Cij = Aij + Bij. You may assume that the lists passed as parameters have the same dimensions.

**5.** A square matrix of size n is a magic square if all of its row, column and diagonal sums are equal. Write a function is\_magic\_square that accepts a two-dimensional list of integers as a parameter and returns True if it is a magic square or False otherwise.

# Solutions

1.  
0 1 2 3  
1 2 3 4  
2 3 4 5

2.  
4 5 6 6  
5 6 7 7  
6 7 8 8

3.  
for r in range(len(matrix)):

matrix[r][4] = matrix[r][1]

4.  
def matrix\_add(mat1, mat2):

matrix\_sum = []

for r in range(len(mat1)):

sum\_list = []

for c in range(len(mat1[r])):

sum\_list.append(mat1[r][c] + mat2[r][c])

matrix\_sum.append(sum\_list)

5.

def is\_magic\_square(matrix):

if len(matrix) != len(matrix[0]):

return False

magic\_number = 0

for c in range(len(matrix[0])):

magic\_number += matrix[0][c]

for i in range(1, len(matrix)):

row\_sum = 0

col\_sum = 0

for j in range(0, len(matrix)):

row\_sum += matrix[i][j]

col\_sum += matrix[j][i]

if (row\_sum != magic\_number) or (col\_sum != magic\_number):

return False

diag\_1\_sum = 0

diag\_2\_sum = 0

for i in range(len(matrix)):

diag\_1\_sum += matrix[i][j]

diag\_2\_sum += matrix[i][len(matrix) - 1 - i]

if (diag\_1\_sum != magic\_number) or (diag\_2\_sum != magic\_number):

return False

return True