SAMS
Programming A/B

Week 5 Lecture – 2-d Lists
July 31, 2017

Mark Stehlik

Two-dimensional lists

- Some data can be organized efficiently in a **table** (also called a **matrix** or **2-dimensional list**)
- Each cell is denoted with two subscripts, a row and column indicator

B	0	1	2	3	4
0	3	18	43	49	65
1	14	30	32	53	75
2	9	28	38	50	73
3	10	24	37	58	62
4	7	19	40	46	66

B[2][3] = 50

2-d Lists in Python

```
data = [ [1, 2, 3, 4],
         [5, 6, 7, 8],
         [9, 10, 11, 12]
       ]

>>> data[0]
[1, 2, 3, 4]
>>> data[1][2]
7
>>> data[2][5]   index error
```

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

Accessing row and column info…

```
lst = [ [1, 2, 3], [4, 5, 6] ]
print(lst) #prints [[1, 2, 3], [4, 5, 6]]

print(len(lst))  #prints 2
print(len(lst[0]) #prints 3
```

2-d List Example in Python

- Find the sum of all elements in a 2-D list

```
def matrixSum(table):
    total = 0
    for row in range(0, len(table)):
        for col in range(0, len(table[row])):
            total += table[row][col]
    return total
```

number of rows in the table

Number of columns in the given row of the table

In a rectangular matrix, this number will be the same for each row so we could use a fixed number for row such as len(table[0])

Tracing the Nested Loop

```
def matrixSum(table):
    total = 0
    for row in range(0, len(table)):
        for col in range(0,len(table[row])):
            total += table[row][col]
    return total
```

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

len(table) = 3

len(table[row])= 4 for every row

row	col	sum
0	0	1
0	1	3
0	2	6
0	3	10
1	0	15
1	1	21
1	2	28
1	3	36
2	0	45
2	1	55
2	2	66
2	3	78

Printing a 2-d list

```
print(lst)  # not very "pretty", as we saw

def print2d(lst):
    for row in range(len(lst)):
        print(lst[row])

#prints
[1, 2, 3]
[4, 5, 6]
```

2-dimensional lists – beware of aliasing!

- How to make a Tic-Tac-Toe board?

```
board = [' ',' ',' ']   # one row
board = [' ',' ',' '] * 3  # since I want 3 of them…
```

but it just makes a 9-element, 1-d list!

- OK, how about

```
board2 = [ [' '] * 3] * 3 #incorrect due to aliasing (but is 3x3!)
```

- Correct…

```
board = [ ]
for row in range(3):
        board += [ [' '] * 3]
```

2-dimensional lists

- Let's play a game of Tic-Tac-Toe (to be continued…)