SAMS
Programming A/B

Week 5 Lecture – 2-d Lists (and Tuples)
July 30, 2018

Mark Stehlik

Two-dimensional lists

- But first, a look at aliasing, and random numbers!
- Let's look again at problem 1 from the homework

Random numbers

- First, import random
- Generate a random real value between 0 and 100?
  - random.random() * 100 # random.random() returns a float in the range [0, 1)
- Generate a die roll (integers from 1 – 6)?
  - random.randint(1, 6)
- What should be true of the values returned?
  - they should be uniform
- How can we test that?

On to Two-dimensional lists

- Some data can be organized efficiently in a **table** (also called a **matrix** or **2-dimensional list**)
- A 2d list is just a 1d list whose individual elements are themselves lists, e.g.,

  a = [[42, 13, 4], [3, 0, 1]]
- This list, a, has two elements: so a[0] is the list [42, 13, 4] and a[1] is the list [3, 0, 1]

Two-dimensional lists

- Each cell is denoted with two subscripts, a row and column indicator, i.e., [row][col]

B[2][3] == 50

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

2d 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 number of rows and columns

```
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
```

2d List Example in Python

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

```
def matrixSum(table):
    total = 0
    for row in range(len(table)):
        for col in range(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 2d list

```
print(lst)  # not "pretty", as we saw, but we can do better

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, **1d** list!

- OK, how about

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

- Nope, here's the correct way…

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

Tuples

- Similar to lists, except entries are *immutable* (not changeable), so no
  - *tuple*.append(), insert(), remove(), sort()
  - but accessed like a list, i.e. *tuple*[0]
- Used when items are not going to change (well…) and are not the same type; also to return more than one value from a function
- Examples:
  - student = ("Mark", "Stehlik", "mjs", [100,80,85])
  - student2 = ("Susan", "Jones", "sjones", [100,100,100])
  - could I add a grade to either student? How?

Tuple Example

```
def firstIndex(table, target):
    for row in range(len(table)):
        for col in range(len(table[row])):
            if (table[row][col] == target)
                return (row,col) # returns one value!
    return -1

a = [ [1,2,3,4], [5,6,7,8], [10,45,12] ]
index = firstIndex(a,45)
print(index)  ->  (2,1) # the tuple (2,1)
print(index[0])  ->  2
print(index[1])  ->  1
```

2-dimensional lists

- Up until now we've written functions "in isolation", if you will
- Let's write a program to play a game of Tic-Tac-Toe (to be continued on the homework…) which uses a 3x3 2d list of single-character strings to store the board and keep track of moves