

Objectives

- Two-dimensional lists

Dec 3, 2007

Sprenkle - CS111

1

Lists

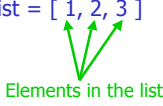
- We've used lists that contain
 - Integers
 - Strings
 - Cards (Deck class)
 - Songs (your MusicCollection class)
- We discussed that lists can contain multiple types of objects within the same list
- Lists can contain any **type** of object
 - Even **LISTS**!

Dec 3, 2007

Sprenkle - CS111

2

Review of Regular (1D) Lists

- Create a list `onedlist = [1, 2, 3]`

- `len(onedlist)` is 3
- `onedlist[2]` is 3

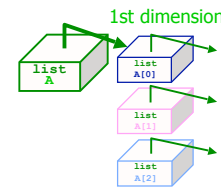
Dec 3, 2007

Sprenkle - CS111

3

A List of Lists: 2-dimensional List

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



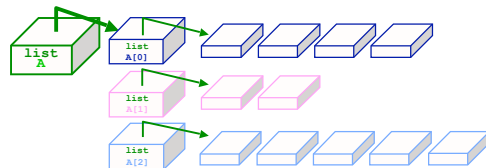
Dec 3, 2007

Sprenkle - CS111

4

A List of Lists: 2-dimensional lists

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



- "Rows" within two-dimensional list do **not** need to be same length
- However, it's often easier to have them the same length!
 - We'll focus on "rectangular" 2-d lists

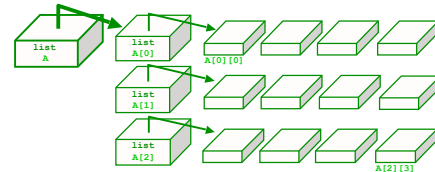
Dec 3, 2007

Sprenkle - CS111

5

Handling Rectangular Lists

`A[1][2] = 42`



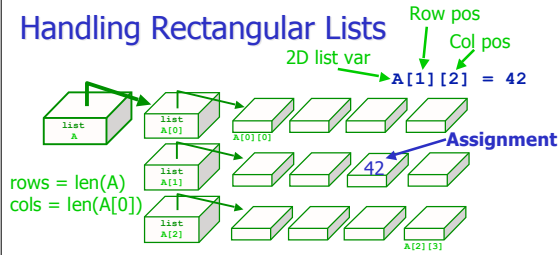
- What does each component of `A[1][2]` mean?
- How many rows does `A` have, in general?
- How many columns does `A` have, in general?

Dec 3, 2007

Sprenkle - CS111

6

Handling Rectangular Lists



- What does each component of `A[1][2]` mean?
- How many rows does `A` have, in general?
- How many columns does `A` have, in general?

Dec 3, 2007

Sprengle - CS111

7

Practice

Starting with the 2D list `A` shown here, what are the values in `A` after running this code?

A Before

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

```
def mystery(A):
    """ "run" this on A, at right """
    for row in xrange( len(A) ):
        for col in xrange( len(A[0]) ):
            if row == col:
                A[row][col] = 42
            else:
                A[row][col] += 1
```

A After

Dec 3, 2007

Sprengle - CS111 `mystery.py`

8

Practice

Starting with the 2D list `A` shown here, what are the values in `A` after running this code?

A Before

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

```
def mystery(A):
    """ "run" this on A, at right """
    for row in xrange( len(A) ):
        for col in xrange( len(A[0]) ):
            if row == col:
                A[row][col] = 42
            else:
                A[row][col] += 1
```

A After

42	3	4	5
6	42	8	9
10	11	42	13

Dec 3, 2007

Sprengle - CS111 `mystery.py`

9

Creating a 2d List

- > `matrix = []`
- Need to create a row of the list
 - > `row = [1, 2, 3, 4]`
- Then append that row to the list
 - > `matrix.append(row)`
 - > `print matrix`
 - `[[1, 2, 3, 4]]`
- Repeat
 - > `row = [1, 2, 3, 4]`
 - > `matrix.append(row)`
 - > `print matrix`
 - `[[1, 2, 3, 4], [1, 2, 3, 4]]`

Dec 3, 2007

Sprengle - CS111

10

Generalize Creating a 2D List

- Create a function that returns a 2D list with width **cols** and height **rows**
 - > Initialize each element in list to 0

Dec 3, 2007

Sprengle - CS111

11

Generalize Creating a 2D List

```
def create2DList(rows, cols):
    twodlist = []
    for row in xrange( rows ):
        row = []
        for col in xrange( cols ):
            row.append(0)
        twodlist.append(row)
    return twodlist
```

Dec 3, 2007

Sprengle - CS111

12

Generalize Creating a 2D List

- The following code **won't** work. Why?
- Explain output from example program

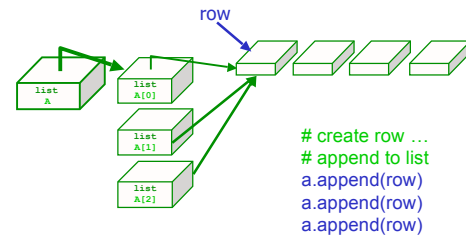
```
def noCreate2DList(rows, cols):
    twodlist = []
    row = []
    for col in xrange( cols ):
        row.append(0)
    for row in xrange( rows ):
        row = []
        twodlist.append(row)
    return twodlist
```

Dec 3, 2007

Sprengle - CS111 twod_exercises.py 13

All Rows Pointing at Same Block of Memory

- Each row points to the same row



Dec 3, 2007

Sprengle - CS111

14

Problem: Create a Tic-Tac-Toe board

- Write a function that returns a 2D list that represents a tic-tac-toe board
 - What elements should be in the 2D list?

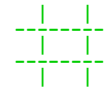
Dec 3, 2007

Sprengle - CS111

15

Problem: Print a Tic-Tac-Toe Board

- Print the board in a "nice" way, such as



Dec 3, 2007

Sprengle - CS111

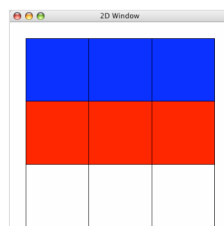
tictactoe.py

16

Graphical Representation of 2D Lists

- Module: csplot
- Allows you to visualize your 2D list
 - Numbers are represented by different colors

```
import csplot
...
# create 2D list...
twodlist=[ [0,0,0], [1,1,1], [2,2,2] ]
# display list graphically
csplot.show(twodlist)
```



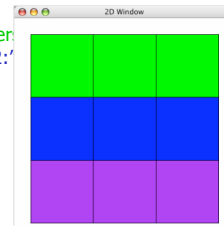
Dec 3, 2007

Sprengle - CS111

Graphical Representation of 2D Lists

- Can assign colors to numbers

```
import csplot
...
# create 2D list...
twodlist=[ [0,0,0], [1,1,1], [2,2,2] ]
# create optional dictionary of number
numToColor={0:"purple", 1:"blue", 2:"green"}
csplot.show(twodlist, numToColor)
```



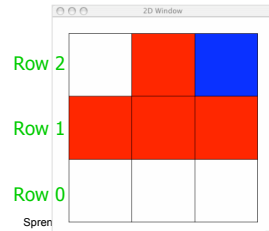
Dec 3, 2007

Sprengle - CS111

Graphical Representation of 2D Lists

- Note that representation of rows is backwards from how we've been visualizing

```
matrix = [[0,0,0], [1,1,1], [0,1,2]]
```



Dec 3, 2007

Sprenkle

19

Game Board for Connect Four

- 6 rows, 7 columns board
- Players alternate dropping red/black checker into slot/column
- Player wins when have four checkers in a row vertically, horizontally, or diagonally
- How to represent board in 2D list, using graphical representation?

Dec 3, 2007

Sprenkle - CS111

20

Game Board for Connect Four

- How to represent board in 2D list, using graphical representation?

Number	Meaning	Color
0	Free	Yellow
1	Player 1	Red
2	Player 2	Black

Dec 3, 2007

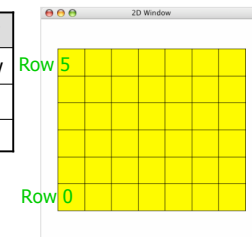
Sprenkle - CS111

21

Game Board for Connect Four

- How to represent board in 2D list, using graphical representation?

Number	Meaning	Color
0	Free	Yellow
1	Player 1	Red
2	Player 2	Black



Dec 3, 2007

Sprenkle - CS111

22

Connect Four (C4): Making moves

- User clicks on a column
 - "Checker" is filled in at that column

```
# gets the column of where user clicked
col = csplot.sqinput()
```

Dec 3, 2007

Sprenkle - CS111

23

Problem: C4 - Valid move?

- Need to enforce valid moves
 - In physical game, run out of spaces for checkers if not a valid move
- How can we determine if a move is valid?
 - How do we know when a move is **not** valid?

Dec 3, 2007

Sprenkle - CS111

24

Problem: C4 - Valid move?

- Solution: check the “top” spot
 - If it's free, then it's a valid move

Dec 3, 2007

Sprenkle - CS111

25

We're Making a Connect Four Class

- Data
 - Board
- Methods
 - Constructor
 - Display the board
 - Play the game
 - Repeat:
 - Get input/move from user
 - Check if valid move
 - Display board
 - Check if win

Wed's Group Work

Dec 3, 2007

Sprenkle - CS111

26

Problem: Determine Win in Tic-Tac-Toe

- Determine when a player wins in tic-tac-toe

Dec 3, 2007

Sprenkle - CS111

27

Problem: Determine Win in Tic-Tac-Toe

- Determine when a player wins in tic-tac-toe
 - What additional information would make determining a win a little more efficient?
- Note that we are not handling tie games well

Dec 3, 2007

Sprenkle - CS111

tictactoe2.py

28

Problem: Determine Win in C4

- Determine when a player wins in Connect Four
 - Why is this similar to but different from Tic-Tac-Toe problem?
- Wednesday's Group Work

Dec 3, 2007

Sprenkle - CS111

29

Final Grades

- Final Exam: Comprehensive
 - What was covered on first two exams PLUS
 - Linear vs Binary search
 - Two-dimensional lists
 - One question on Broader Issues in Computer Science (give specifically later)
- Final grade computation is online
 - Exam grade=30%
 - 13% worse score, 17% better score

Dec 3, 2007

Sprenkle - CS111

30

Plan for This Week

- Tomorrow: Lab 11
 - Music manager - binary search
 - 2D list practice
- Wednesday: back **in lab**
 - 2D list group work
 - Course evaluations
- Friday
 - Programs in other programming languages
 - Broader Issue - DARPA Urban challenge

Dec 3, 2007

Sprenkle - CS111

31