

## Objectives

- Two-dimensional lists

Mar 31, 2007

Sprengle - CS111

1

## Lists

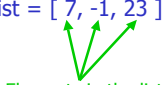
- We've used lists that contain
  - Integers
  - Strings
  - Cards (Deck class)
  - Songs (your MusicCollection class)
- We discussed that lists can contain multiple types of objects within the same list
- Lists can contain **any type** of object
  - Even **LISTS**!

Mar 31, 2007

Sprengle - CS111

2

## Review of Regular (1D) Lists

- Create a list `onedlist = [ 7, -1, 23 ]`  

- `len(onedlist)` is 3
- `onedlist[2]` is 23

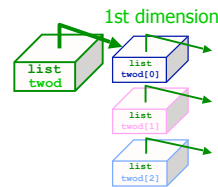
Mar 31, 2007

Sprengle - CS111

3

## A List of Lists: 2-dimensional List

```
twod = [ twod[0] twod[1] twod[2]
         [ 1,2,3,4], [ 5,6], [ 7,8,9,10,11] ]
```



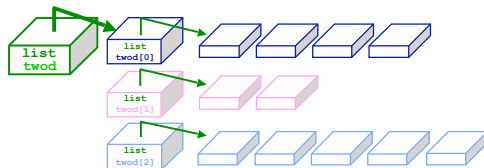
Mar 31, 2007

Sprengle - CS111

4

## A List of Lists: 2-dimensional lists

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



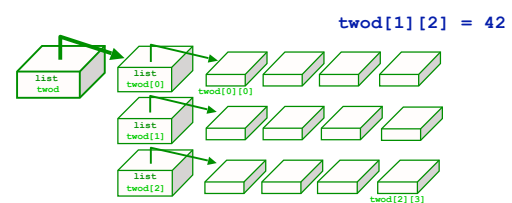
- “Rows” within two-dimensional list do **not** need to be same length
- However, it's often easier to have them the same length!
  - We'll focus on “rectangular” 2-d lists

Mar 31, 2007

Sprengle - CS111

5

## Handling Rectangular Lists



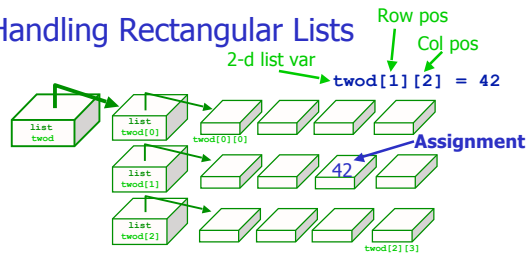
- What does each component of `twod[1][2]` mean?
- How many rows does `twod` have, in general?
- How many columns does `twod` have, in general?

Mar 31, 2007

Sprengle - CS111

6

## Handling Rectangular Lists



- What does each component of `twod[1][2]` mean?
- How many rows does `twod` have, in general?  
 > `rows = len(twod)`
- How many columns does `twod` have, in general?  
 > `cols = len(twod[0])`

Mar 31, 2007

Sprengle - CS111

7

## Practice

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

**twod 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(twod):
    """ "run" this on twod, at right """
    for row in range( len(twod) ):
        for col in range( len(twod[0]) ):
            if row == col:
                twod[row][col] = 42
            else:
                twod[row][col] += 1
```

**twod After**


Mar 31, 2007

Sprengle - CS111

mystery.py

8

## Practice

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

**twod 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(twod):
    """ "run" this on twod, at right """
    for row in range( len(twod) ):
        for col in range( len(twod[0]) ):
            if row == col:
                twod[row][col] = 42
            else:
                twod[row][col] += 1
```

**twod After**

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

Mar 31, 2007

Sprengle - CS111

mystery.py

9

## Creating a 2d List

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

Mar 31, 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

Mar 31, 2007

Sprengle - CS111

11

## 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

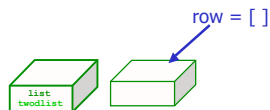
```
def create2DList(rows, cols):
    twodlist = []
    # for each row
    for row in xrange( rows ):
        row = []
        # for each column, in each row
        for col in xrange( cols ):
            row.append(0)
        twodlist.append(row)
    return twodlist
```

Mar 31, 2007

Sprengle - CS111

12

## How Does This Work?

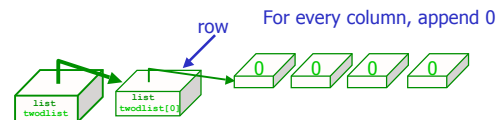


Mar 31, 2007

Sprengle - CS111

13

## How Does This Work?



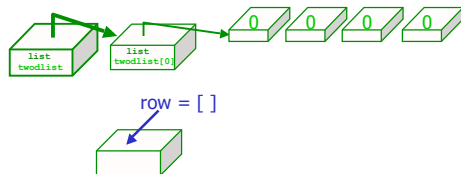
Append row to twodlist

Mar 31, 2007

Sprengle - CS111

14

## How Does This Work?

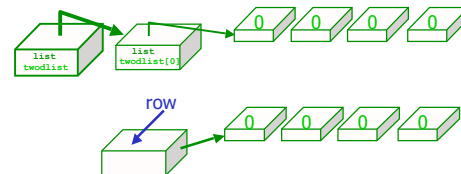


Mar 31, 2007

Sprengle - CS111

15

## How Does This Work?

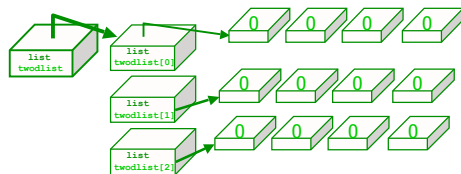


Mar 31, 2007

Sprengle - CS111

16

## How Does This Work?



Mar 31, 2007

Sprengle - CS111

17

## Generalize Creating a 2D List

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

```
def noCreate2DList(rows, cols):
    twodlist = []
    row = []
    # create a row with appropriate columns
    for col in xrange( cols ):
        row.append(0)
    # append the row rows times
    for row in xrange( rows ):
        twodlist.append(row)
    return twodlist
```

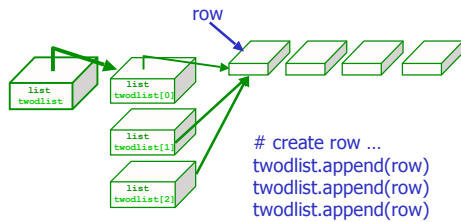
Mar 31, 2007

Sprengle - CS111 twod\_exercises.py

18

## All Rows Pointing at Same Block of Memory

- Each row points to the **same** row in memory



Mar 31, 2007

Sprengle - CS111

19

## Problem: Create a Tic-Tac-Toe board

- Returns a 2-d list that represents a tic-tac-toe board
  - What elements should be in the 2D list?

Mar 31, 2007

Sprengle - CS111

20

## Problem: Tic-Tac-Toe

- How do we represent player's moves?
  - How do we update the board to say "Player X goes into the bottom right corner."

Mar 31, 2007

Sprengle - CS111

21

## Problem: Print a Tic-Tac-Toe Board

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

```
x | | 
--|--
| o | 
--|--
| | | 
--|--
```

Mar 31, 2007

Sprengle - CS111

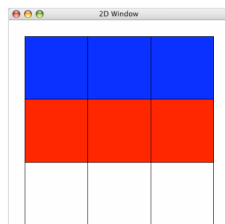
tictactoe.py

22

## 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)
```



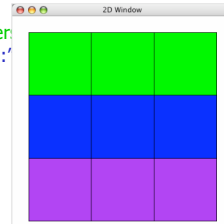
Mar 31, 2007

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)
```



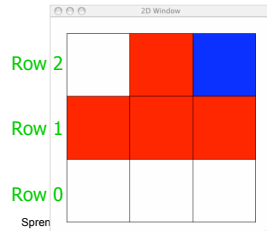
Mar 31, 2007

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]]
```



Mar 31, 2007

Sprenkle

25

## 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?

Mar 31, 2007

Sprenkle - CS111

26

## 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

Mar 31, 2007

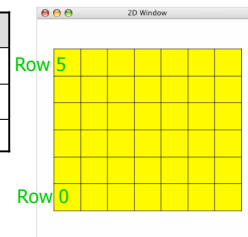
Sprenkle - CS111

27

## 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



Mar 31, 2007

Sprenkle - CS111

28

## 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()
```

Mar 31, 2007

Sprenkle - CS111

29

## 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?

Mar 31, 2007

Sprenkle - CS111

30

### Problem: C4 - Valid move?

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

Mar 31, 2007

Sprenkle - CS111

31

### ConnectFour 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
      - Change player

Mar 31, 2007

Sprenkle - CS111

32

### Problem: C4 - Making a Move

- The player clicks on a column, meaning that's where the player wants to put a checker
- How do we update the board?

Mar 31, 2007

Sprenkle - CS111

33

### Problem: Determine Win in Tic-Tac-Toe

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

Mar 31, 2007

Sprenkle - CS111

34

### Problem: Determine Win in Tic-Tac-Toe

- Determine when a player wins in tic-tac-toe
  - What additional information would make this problem a little easier to manage?
- Note that we are not handling tie games well

Mar 31, 2007

Sprenkle - CS111

tictactoe2.py

35

### Course Grades

- Final Exam: Comprehensive
  - Defining & using classes
  - Searches: Linear, Binary; Breadth-first, Depth-first
  - Two-dimensional lists
  - ...
  - See FinalPrep document on line
- Formula for final grade is on course Web page
  - Exam grade=30%
    - 14% worse score, 16% better score

Mar 31, 2007

Sprenkle - CS111

36