

## Objectives

- Two-dimensional lists

## Review

- What are the two types of search we discussed?
  - How do they work?
- What are the tradeoffs between using linear search and binary search?
- What was our (final) sorting algorithm?

## Algorithm: Merge Sort

```
def mergeSort( listOfNumbers ):  
    if len(listOfNumbers) == 2: # base case  
        # sort those two numbers  
        if listOfNumbers[0] > listOfNumbers[1]:  
            temp = listOfNumbers[0]  
            listOfNumbers[0] = listOfNumbers[1]  
            listOfNumbers[1] = temp  
        return listOfNumbers  
    firsthalf = listOfNumbers[:len(listOfNumbers)//2 ]  
    secondhalf = listOfNumbers[len(listOfNumbers)//2:]  
    sortedFirst = mergeSort( firsthalf )  
    sortedSecond = mergeSort( secondhalf )  
    whole = merge( sortedFirst, sortedSecond )  
    return whole
```

Mar 30, 2018

Sprenkle - CSCI111

3

## 2D LISTS

Apr 2, 2018

Sprenkle - CSCI111

4

## Lists

- We've used lists that contain
  - Integers
  - Strings
  - Cards (Deck class)
  - Persons (your Person class)
- We discussed that lists can contain multiple types of objects within the same list
  - Wheel of Fortune: ["Bankrupt", 250, 350, ...]
- Lists can contain *any* **type** of object
  - Even **LISTS**!

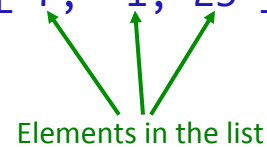
Apr 2, 2018

Sprenkle - CSCI111

5

## Review of Regular (1D) Lists

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



Elements in the list

- How do we find the number of elements in the list?
- How can we find the value of the third element in the list?

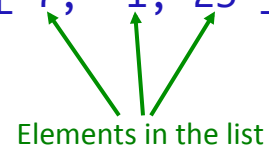
Apr 2, 2018

Sprenkle - CSCI111

6

## Review of Regular (1D) Lists

- Create a list `onedlist = [ 7, -1, 23 ]`
- `len(onedlist)` is 3
- `onedlist[2]` is 23



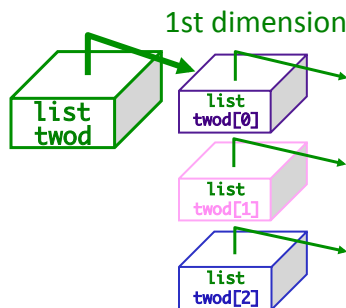
Apr 2, 2018

Sprenkle - CSCI111

7

## A List of Lists: 2-dimensional List

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



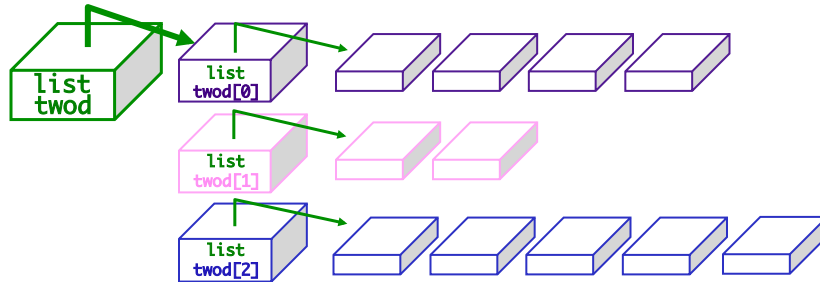
Apr 2, 2018

Sprenkle - CSCI111

8

## A List of Lists: 2-dimensional lists

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



- “Rows” within 2-dimensional list do **not** need to be the same length
- However, it’s often easier if they’re the same length!
  - We’ll focus on “rectangular” 2-d lists

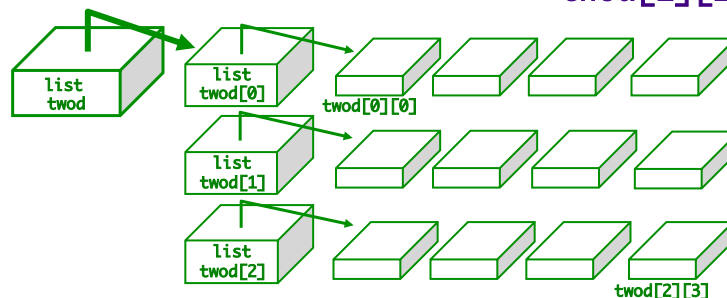
Apr 2, 2018

Sprenkle - CSCI111

9

## Handling Rectangular Lists

$twod[1][2] = 42$



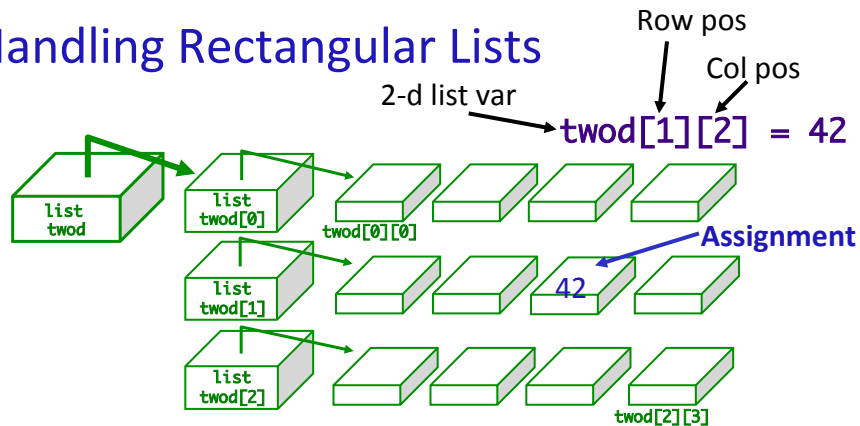
- What does each component of  $twod[1][2]$  mean?
- How many rows does  $twod$  have, in general?
- How many columns does  $twod$  have, in general?

Apr 2, 2018

Sprenkle - CSCI111

10

## 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])`

Apr 2, 2018

Sprenkle - CSCI111

11

## 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


Apr 2, 2018

Sprenkle - CSCI111

`mystery.py`

12

## 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

Apr 2, 2018

Sprenkle - CSCI111

mystery.py

13

## Typical Use of 2D List

1. Initialize the 2D list
  1. Make all the “spots” available in the list
  2. Initialize those spots to some value
2. Fill in the spots as appropriate.

Apr 2, 2018

Sprenkle - CSCI111

14

## Example: Creating a 2d List

- `twod = [ ]`
- Create a row of the list  
`row = [1, 2, 3, 4]` or `row = list(range(1,5))`
- 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] ]`

Apr 2, 2018

Sprenkle - CSCI111

15

## Generalize Creating a 2D List

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

Apr 2, 2018

Sprenkle - CSCI111

16



## 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

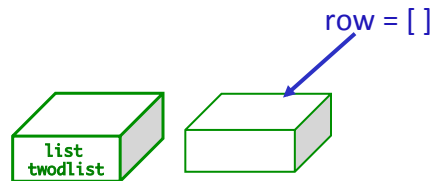
```
def create2DList(rows, cols):  
    twodlist = [ ]  
    # for each row  
    for row in range( rows ):  
        row = [ ]  
        # for each column, in each row  
        for col in range( cols ):  
            row.append(0)  
        twodlist.append(row)  
    return twodlist
```

Apr 2, 2018

Sprenkle - CSCI111

17

## How Does This Work?

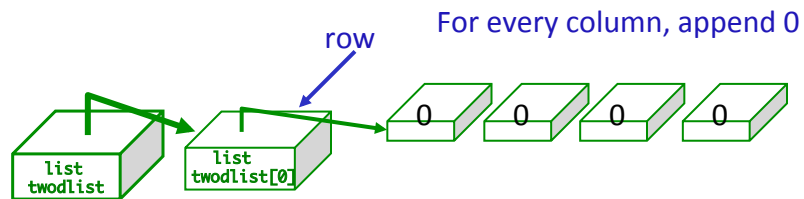


Apr 2, 2018

Sprenkle - CSCI111

18

## How Does This Work?



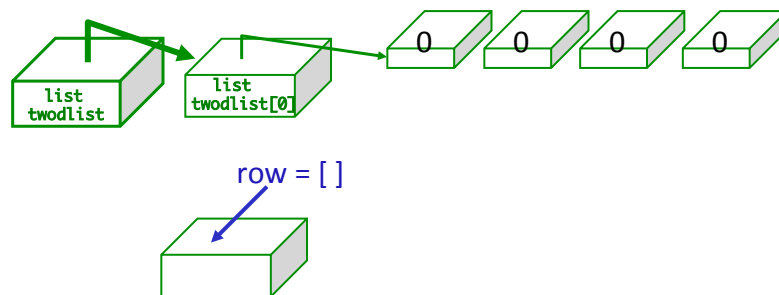
Append row to twodlist

Apr 2, 2018

Sprenkle - CSCI111

19

## How Does This Work?

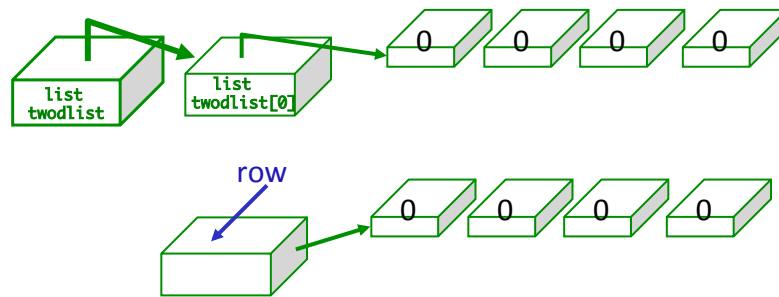


Apr 2, 2018

Sprenkle - CSCI111

20

## How Does This Work?

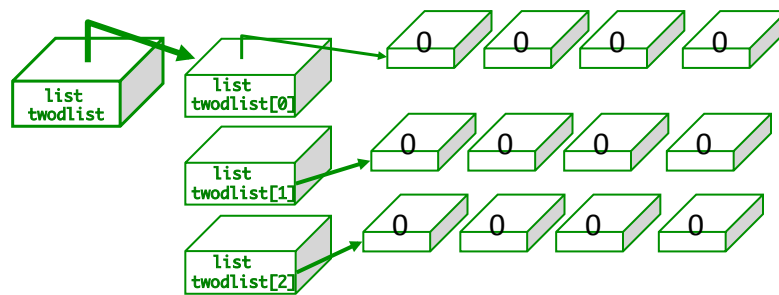


Apr 2, 2018

Sprenkle - CSCI111

21

## How Does This Work?



Apr 2, 2018

Sprenkle - CSCI111

22

## Incorrect: 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 range( cols ):  
        row.append(0)  
    # append the row rows times  
    for r in range( rows ):  
        twodlist.append(row)  
    return twodlist
```

twod\_exercises.py

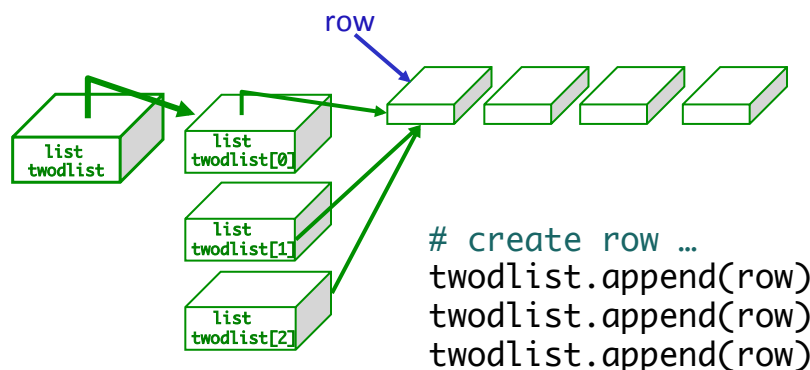
Apr 2, 2018

Sprenkle - CSCI111

23

## All Rows Pointing at Same Block of Memory

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



Apr 2, 2018

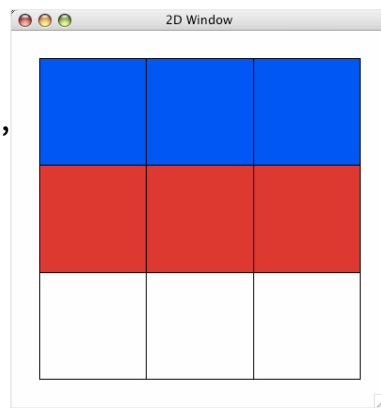
Sprenkle - CSCI111

24

## 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)
```



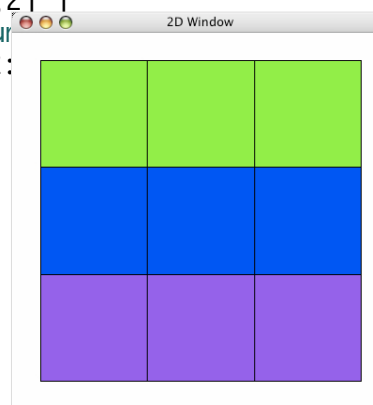
Apr 2, 2018

Sprenkle - CSCI111

## 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 num
numToColor={0:"purple", 1:"blue", 2:"green"}
cspLOT.show(twodlist, numToColor)
```



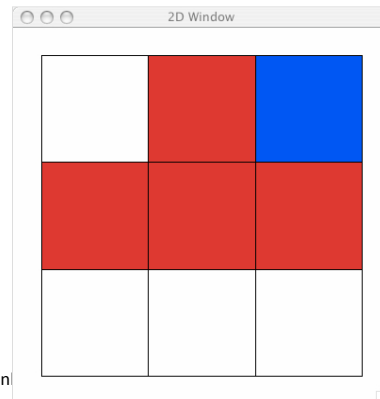
Apr 2, 2018

Sprenkle - CSCI111

## Graphical Representation of 2D Lists

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

What values map  
to which colors  
by default?



Apr 2, 2018

Spren

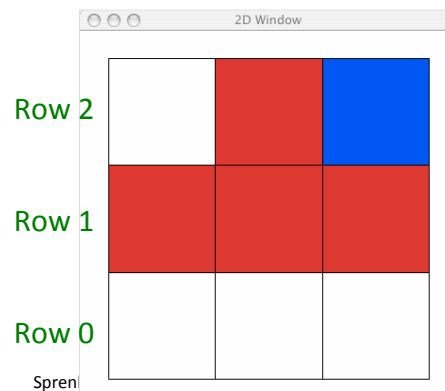
27

## 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]]
```

What values map  
to which colors  
by default?



Apr 2, 2018

Spren

28

## 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 do we represent the board as a 2D list, using a graphical representation?

Apr 2, 2018

Sprenkle - CSCI111

29

## 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

Apr 2, 2018

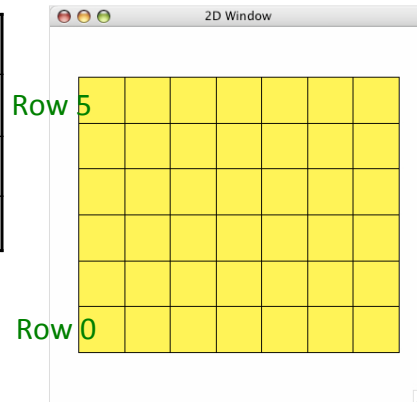
Sprenkle - CSCI111

30

## 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



Apr 2, 2018

Sprenkle - CSCI111

31

## ConnectFour Class

- What is the data associated with the class?
- What methods should we implement?

Apr 2, 2018

Sprenkle - CSCI111

32



## ConnectFour Class

- Data
  - Board + constants
    - 6 rows, 7 columns, all FREE to start
- Methods
  - Constructor
  - Display the board
  - Play the game
  - Get input/move from user
  - Check if valid move
  - Make move
  - Check if win

Apr 2, 2018

Sprenkle - CSCI111

33

## ConnectFour Constants

```
class ConnectFour:
    """ Class representing the game Connect Four. """

    # Represent different values on the board
    FREE = 0
    PLAYER1 = 1
    PLAYER2 = 2

    # Represent the dimensions of the board
    ROWS = 6
    COLS = 7
```

Apr 2, 2018

Sprenkle - CSCI111

34

## ConnectFour Class

- Play the game method implementation

- Repeat:

- Get input/move
    - Check if valid move
    - Make move
    - Display board
    - Check if win
    - Change player

```
def play(self):
    won = False
    player = ConnectFour.PLAYER1

    while not won:
        print("Player %d's move" % player)
        if player == ConnectFour.PLAYER1:
            col = self._userMakeMove()
        else: # computer is player 2
            # pause because otherwise move happens too
            # quickly and looks like an error
            sleep(.75)
            col = self._computerMakeMove()

        row = self.makeMove(player, col)
        self.showBoard()
        won = self._isWon(row, col)

        # alternate players
        player = player % 2 + 1
```

Apr 2, 2018

Sprenkle - CSCI111

35

## 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()
```

```
def _userMakeMove(self):
    """ Allow the user to pick a column."""
    col = csplot.sqinput()
    validMove = self._isValidMove(col)
    while not validMove:
        print("NOT A VALID MOVE.")
        print("PLEASE SELECT AGAIN.")
        print()
        col = csplot.sqinput()
        validMove = self._isValidMove(col)
    return col
```

Apr 2, 2018

Sprenkle - CSCI111

36

## 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?

Apr 2, 2018

Sprenkle - CSCI111

37

## Problem: C4 - Valid move?

- Solution: check the “top” spot
  - If the spot is FREE, then it’s a valid move

Apr 2, 2018

Sprenkle - CSCI111

38

## 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?

Apr 2, 2018

Sprenkle - CSCI111

39

## Looking Ahead

- Lab 11 – Tomorrow
- Broader Issue: Facebook – Friday
- Bring Exam envelopes

Apr 2, 2018

Sprenkle - CSCI111

40