

Objectives

- Handling exceptions
- Two-dimensional lists

Apr 4, 2011

Sprenkle - CSCI111

1

Computer Science Understanding

- Do you understand what a class is and its purpose? What is a class made up of?
- Can you implement a class (of "reasonable" size), given what it is supposed to represent and what it is supposed to do?
- Given a class's API, can you solve problems with it?
 - When you write the UI for FaceSpace, you are using the API for the `SocialNetwork` class
- Do you understand the strengths and weaknesses of linear and binary search? When would you use one over the other?

Apr 4, 2011

Sprenkle - CSCI111

2

Validating User Input

Recall your calculating birthday year program.
What happened when a user entered something like "B6"?

Apr 4, 2011

Sprenkle - CSCI111

3

Validating User Input

- What happened when the user entered something like "B6"?
 - Threw an **Exception** and the program exited

Python interpreter's message:

```
Enter your age: B6
Traceback (most recent call last):
  File "currentAge.py", line 22, in <module>
    main()
  File "currentAge.py", line 9, in main
    age=input("Enter your age: ")
  File "<string>", line 1, in <module>
NameError: name 'B6' is not defined
```

Apr 4, 2011

Sprenkle - CSCI111

4

Handling Exceptions

- Using try/except statements
- Syntax:

```
try:
    <body>
except [<errorType>] :
    <handler>
```

Optional: use this to handle specific error types appropriately
- Example:

```
try:
    age = input("Enter your age: ")
    currentyear = input("Enter the current year: ")
except:
    print "ERROR: Your input was not in the correct form."
    print "Enter integers for your age and the current year"
    return
```

Apr 4, 2011

Sprenkle - CSCI111 `yearborn.py`

5

Handling Exceptions

- Could put **try/except** statements in a loop to make sure user enters valid input
 - Example: `birthyear3.py`
- Other types of exceptions
 - File exceptions:
 - File doesn't exist
 - Don't have permission to read/write file

Apr 4, 2011

Sprenkle - CSCI111 `file_handle.py`

6

Handling Exceptions

- Using try/except statements

- Syntax:

```
try:
    <body>
except [<errorType>] :
    <handler>
```

Optional: use this to handle specific error types appropriately

- Example:

```
try:
    age = input("Enter your age: ")
    currentyear = input("Enter the current year: ")
except:
    print "ERROR: Your input was not in the correct form."
    print "Enter integers for your age and the current year"
    return
```

Apr 4, 2011

Sprenkle - CSCI1111

yearborn.py

7

Handling Exceptions

- Could put try/except statements in a loop to make sure user enters valid input
 - Example: birthyear3.py

- Other types of exceptions

File exceptions:

- File doesn't exist
- Don't have permission to read/write file

Apr 4, 2011

Sprenkle - CSCI1111

file_handle.py

8

2D LISTS

Apr 4, 2011

Sprenkle - CSCI1111

9

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 4, 2011

Sprenkle - CSCI1111

10

Review of Regular (1D) Lists

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

Elements in the list

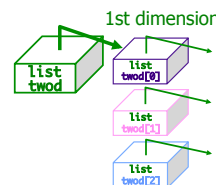
Apr 4, 2011

Sprenkle - CSCI1111

11

A List of Lists: 2-dimensional List

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



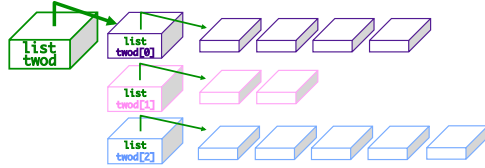
Apr 4, 2011

Sprenkle - CSCI1111

12

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 same length
- However, it's often easier if they're the same length!
➤ We'll focus on “rectangular” 2-d lists

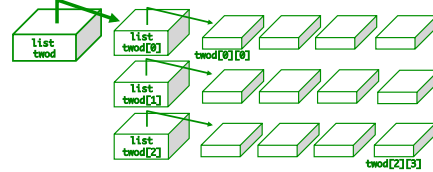
Apr 4, 2011

Sprenkle - CSCI111

13

Handling Rectangular Lists

```
twod[1][2] = 42
```



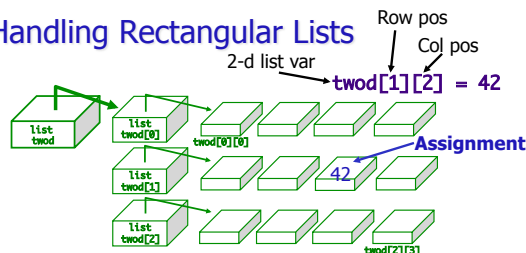
- What does each component of `twod[1][2]` mean?
- How many rows does `twod` have, in general?
- How many columns does `twod` have, in general?

Apr 4, 2011

Sprenkle - CSCI111

14

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 4, 2011

Sprenkle - CSCI111

15

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 xrange( len(twod) ):
        for col in xrange( len(twod[0]) ):
            if row == col:
                twod[row][col] = 42
            else:
                twod[row][col] += 1
```

twod After

Apr 4, 2011

Sprenkle - CSCI111

mystery.py

16

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 xrange( len(twod) ):
        for col in xrange( 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 4, 2011

Sprenkle - CSCI111

mystery.py

17

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]]`

Apr 4, 2011

Sprenkle - CSCI111

18

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

Apr 4, 2011

Sprengle - CSCI1111

19

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

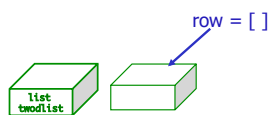
```
def create2DList(rows, cols):
    twodlist = [ ]
    # for each row
    for row in xrange( rows ):
        row = [ ]
        # for each column, in each row
        for col in xrange( cols ):
            row.append(0)
        twodlist.append(row)
    return twodlist
```

Apr 4, 2011

Sprengle - CSCI1111

20

How Does This Work?

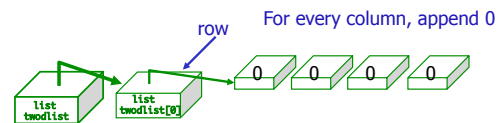


Apr 4, 2011

Sprengle - CSCI1111

21

How Does This Work?



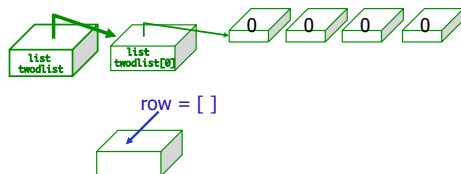
Append row to twodlist

Apr 4, 2011

Sprengle - CSCI1111

22

How Does This Work?

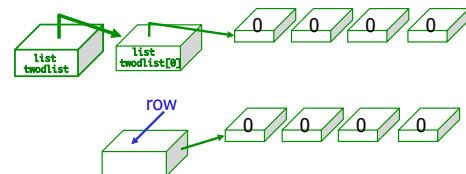


Apr 4, 2011

Sprengle - CSCI1111

23

How Does This Work?

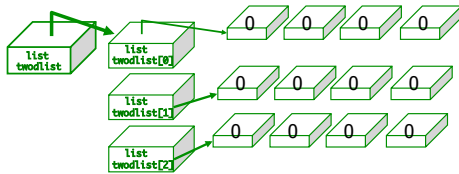


Apr 4, 2011

Sprengle - CSCI1111

24

How Does This Work?



Apr 4, 2011

Sprenkle - CSCI1111

25

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
```

twod_exercises.py

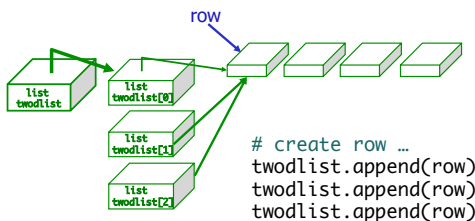
Apr 4, 2011

Sprenkle - CSCI1111

26

All Rows Pointing at Same Block of Memory

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



Apr 4, 2011

Sprenkle - CSCI1111

27

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?

Apr 4, 2011

Sprenkle - CSCI1111

28

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."

Apr 4, 2011

Sprenkle - CSCI1111

29

Problem: Print a Tic-Tac-Toe Board

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

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

Apr 4, 2011

Sprenkle - CSCI1111

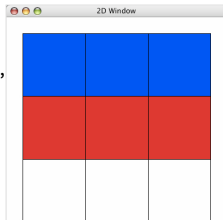
tictactoe.py

30

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)
```



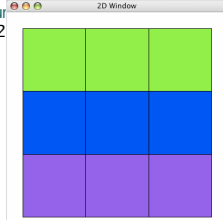
Apr 4, 2011

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
cspLOT.show(twodlist, numToColor)
```



Apr 4, 2011

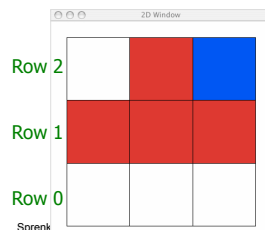
Sprenkle - CSCI111

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?



Apr 4, 2011

Sprenkle

33

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 4, 2011

Sprenkle - CSCI111

34

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 4, 2011

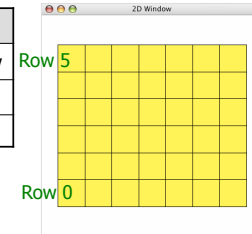
Sprenkle - CSCI111

35

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 4, 2011

Sprenkle - CSCI111

36

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()
```

Apr 4, 2011

Sprenkle - CSCI111

37

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 4, 2011

Sprenkle - CSCI111

38

Problem: C4 - Valid move?

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

Apr 4, 2011

Sprenkle - CSCI111

39

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

Apr 4, 2011

Sprenkle - CSCI111

40

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 4, 2011

Sprenkle - CSCI111

41

Plan for This Week

- Tomorrow: Lab 11
 - SocialNetwork - binary search, exceptions
 - 2D list practice
- Wednesday:
 - Security vulnerability
 - Course evaluations: completed by Sunday at midnight
- Friday
 - Programs in other programming languages
 - Broader Issue – One Laptop Per Child

If 70% of class responds,
1% of possible lab points added (~12 pts)
For each additional 10%,
additional 1% off.
Max ~50 pts

Apr 4, 2011

Sprenkle - CSCI111

42