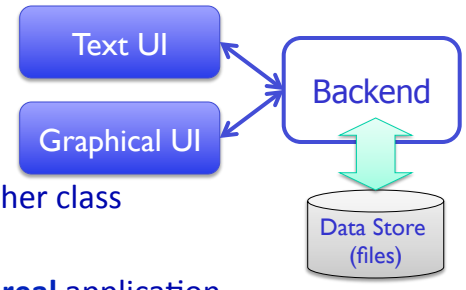
# Reviewing Lab 10




- Created two classes
  - Used one class within another class
  - Tested them
  - Example of a backend to a **real** application
    - Could add a different user interface
- "Good judgment comes from experience"
  - Test methods after writing method
  - Remember your data types
  - Refer to the data type's API
- What could you do to improve your development process?



# Lab 10 Feedback

- Problem solving capstone!
  - Solving lots of different small problems in a variety of ways
- Use methods you've already written
  - Example: use `addPerson` in `addPeople`
  - Who has this functionality? Do I have access to that object in this method?
- Adhere to interface
  - Accepted parameter types
  - Type of what is returned

## Lab 11: Three Parts

- Linux practice:
  - Using the **wc** command
- Social Network extensions
  - Binary search – find people with a certain name
  - UI: add search functionality
- Two-dimensional lists
  - Including Connect Four



## wc Command

- **wc**: Word Count
  - Counts the lines of Social Network code from Lab 10
  - Compare with code for this assignment
- Example:
  - `wc -l ../lab10/*.py`
- Specific directions are in the lab

# Social Network, Extended

- Searching Overview
  - Allows you to search for people by their name– lowercased—for more intuitive results
  - Update `Person` and `SocialNetwork` classes and UI appropriately
    - Specific directions are in the lab



# Extensions to Solution

```
def search(searchlist, key):
   low=0
   high = len(searchlist)-1
   while low <= high :
      mid = (low+high)//2
      if searchlist[mid] == key:
         return mid
      elif key > searchlist[mid]:
         # look in upper half
         low = mid+1
      else:
         # look in lower half
         high = mid-1
   return -1
```

Consider what happens when **`searchlist`** is a list of *`Persons`*, ***`key`*** is a ***`str`*** representing the name

- Goal: find a person with a certain name

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| Person Id: "4" "Ben" | Person Id:"3" "Chris" | Person Id: "1" "Gal" | Person Id:"2" "Natalie" | Person Id:"5" "Samuel" |

## Summary of Modifications to Binary Search

- Add a search method
  - Takes as parameter the name to search for
    - Need to lowercase that name
  - Original binary search function took a list as a parameter; where should we get our list to search?
- Check the *name* of the Person that is at the midpoint, lowercased
- If we have a match, return that Person
- Represent (in method) and handle (in UI) when no person has that name



## SocialNetwork Code

- Fix the major problems in your code first
- Or, use the code in the `handouts/lab10_solution` directory
  - person.py, social.py, facespace.py

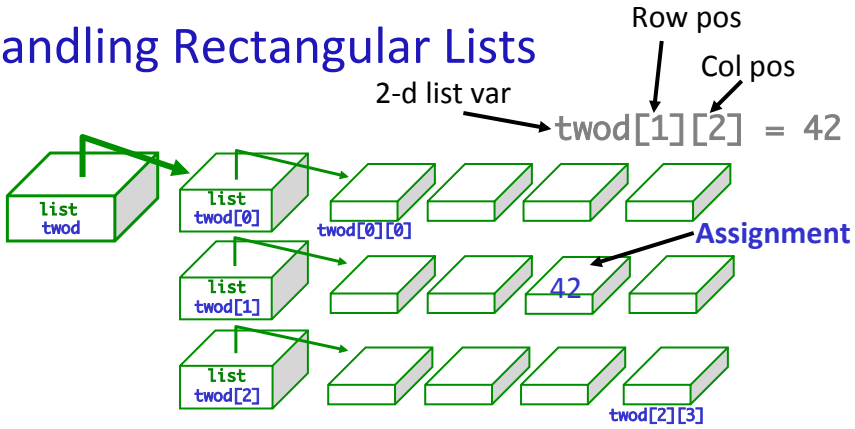
# 2D LISTS



## Review

- How do you create a 2D list?
- How do you get the 2nd element in the 3rd "row" of a list?
- How do you find the number of lists in a 2D list?
- How do you find the number of elements in one of those lists?

# 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])`



# Game Board for Connect Four

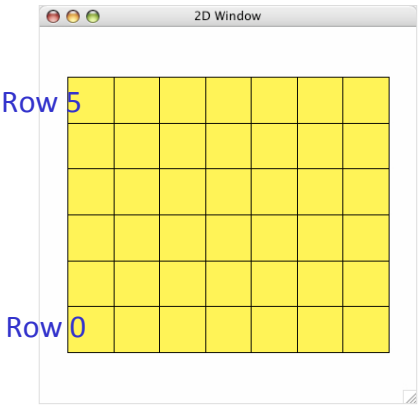
- 6 rows, 7 columns board
- Players alternate dropping red/black checker into slot/column
- Player wins when have four checkers in a row vertically, horizontally, or diagonally

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

## 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 |






## 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()
```

# ConnectFour Class

- Play the game method implementation
  - Repeat:
    - Get input/move f
    - Check if valid mo
    - Make move
    - Display board
    - Check if win
    - Change player

```
won = False
player = ConnectFour.PLAYER1

while not won:
    print("Player {:d}'s move".format(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
```



# 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?

# Looking Ahead

- Bring your final exam envelopes to me by Friday
  - Exam will be taken in Parmly 405
- Bring your final exam questions Friday

Thanks to **Kelly** and **Collin** for their help this semester!