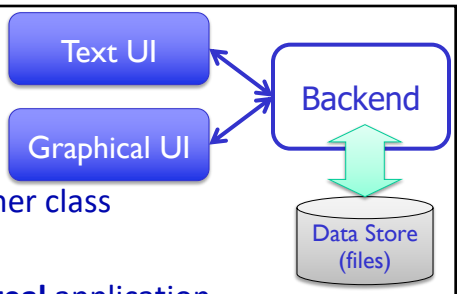
Reviewing Lab 10



- 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
  - Use to count
    - The lines of Social Network code from Lab 10
    - The lines of code for the whole semester
- Example:
  - `wc -l ../lab10/*.py`
- Specific directions are in the lab

Extensions to Search

In InstaFace, we want to find *person* who has a certain name.

Consider what happens when **searchlist** is a list of *Persons* and key is a name (a `str`)

We want to find a `Person` whose name matches the key and return the *Person*

Binary Search Implementation

```
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
```

Extensions to Binary Search

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

Goal: return a Person object with that name (key)

0	1	2	3	4
Person Id: "4" "Brie"	Person Id:"3" "Gal"	Person Id: "1" "Jason"	Person Id:"5" "Samuel"	Person Id:"2" "Scarlett"

Extensions to Solutio

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

Goal: return a Person object with that name (key)

```
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
```

0	1	2	3	4
Person Id: "4" "Brie"	Person Id:"3" "Gal"	Person Id: "1" "Jason"	Person Id:"5" "Samuel"	Person Id:"2" "Scarlett"

List of Person objects

0	1	2	3	4
Person Id: "1" "Jason"	Person Id:"2" "Scarlett"	Person Id: "3" "Gal"	Person Id:"4" "Brie"	Person Id:"5" "Samuel"

Example: looking for a person with the name "Brie"…

List of Person objects

0	1	2	3	4
Person Id: "1" "Jason"	Person Id:"2" "Scarlett"	Person Id: "3" "Gal"	Person Id:"4" "Brie"	Person Id:"5" "Samuel"

List of Person objects sorted by name, e.g.,

0	1	2	3	4
Person Id: "4" "Brie"	Person Id:"3" "Gal"	Person Id: "1" "Jason"	Person Id:"5" "Samuel"	Person Id:"2" "Scarlett"

Extensions to Solutio

```
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 a name

Goal: return a Person object with that name (key)

0	1	2	3	4
Person Id: "4" "Brie"	Person Id:"3" "Gal"	Person Id: "1" "Jason"	Person Id:"5" "Samuel"	Person Id:"2" "Scarlett"

Extensions to Solutio

```
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 a name

Goal: return a Person object with that name (key)

What can we do to make search results more intuitive?

0	1	2	3	4
Person Id: "4" "Brie"	Person Id:"3" "Gal"	Person Id: "1" "Jason"	Person Id:"5" "Samuel"	Person Id:"2" "Scarlett"

Summary of Search Additions

- Add a search method
  - Takes as parameter the name to search for
    - Need to lowercase that name for more intuitive results
  - Original binary search function took a list as a parameter; our method does not
    - Where should we get our list to search?
  - The list to search must be sorted in alphabetical order
- Check the *name* of the Person that is at the midpoint, lowercased
  - If they match, return that Person
  - Otherwise, …
- Represent (in method) and handle (in UI) when no person has that name

Social Network 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

SocialNetwork Code

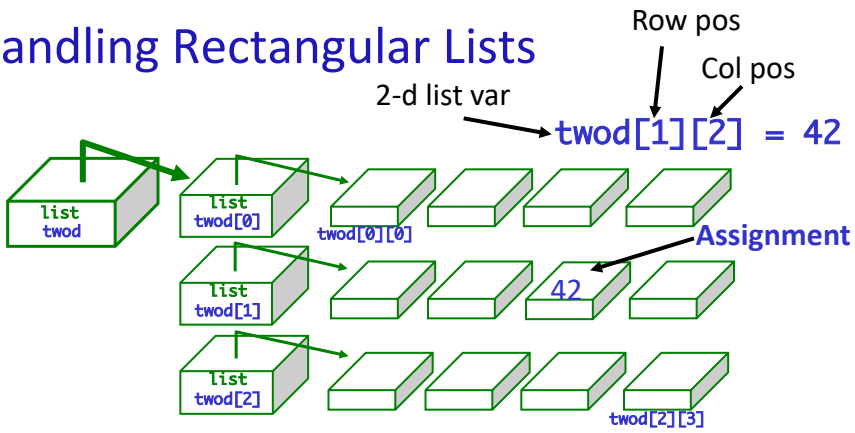
- Fix the major problems in your code first
- Or, use the code in the `handouts/lab10_solution` directory
  - `person.py, social.py, instaface.py`

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

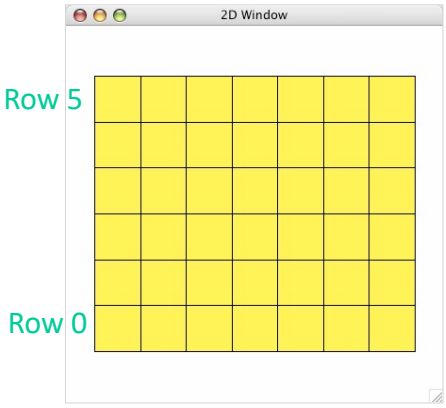
- 6 rows, 7 columns board
- Players alternate dropping red/black checker into slot/column
- Player wins when have four checkers in a row vertically, horizontally, or diagonally

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

Representing Connect Four Game Board

- Using a 2D list

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

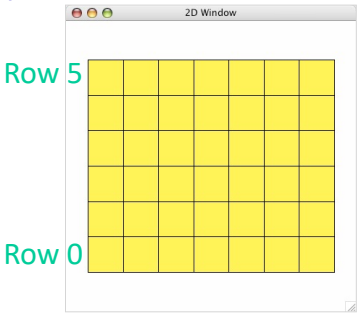


Problem: C4 - Valid move?

- 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?

Problem: C4 - Valid move?

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




```
def _isValidMove(self, col):
    """
    Return True iff the dropping a checker in this col (an int)
    represents a valid move.
    """
    return self._board[ConnectFour.ROWS-1][col] == ConnectFour.FREE
```

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?

Lab 11 Directory

- To start, your directory should look like
  - connectfour.py
  - csplot.py
  - instaface.py instaface.out
  - person.py person.out
  - social.py social.out
  - test.py

Thanks to **Ana** and **Rinn** for their help this semester!