CIS 122

Lists Within Lists

Logistics

- Entering week 7
  - Last week of new material
  - Nested lists
  - Classes

- Next week is Finals week
  - Review Monday, Tuesday, Wednesday
  - Break Thursday
  - Final Friday

- Final times
  - Friday 3:15 - 5:15
  - Wedesday ??? - ???

Logistics

- Assignment 4 graded
    - Still missing a few assignments
    - Will post grades/solution soon

- Nice job overall

- Very creative guesing games
    - Difficulty levels
    - Impressive insults
    - Ascii art

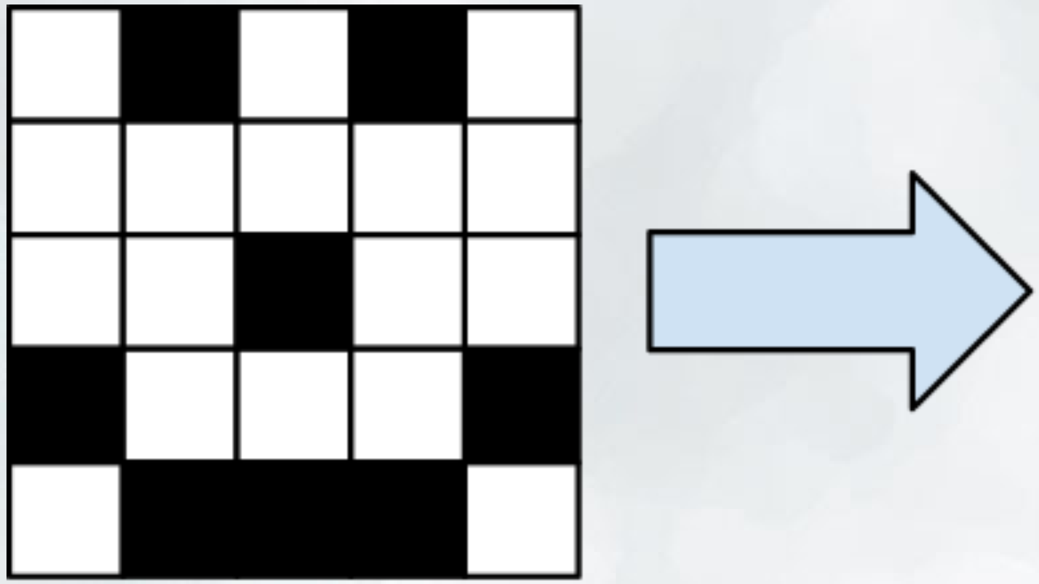
Logistics

- Assignment 5 has been posted
  - Two parts
- Part 0
  - Follows up on last week's concepts
  - No new knowledge required
  - Get it done early
- Part 1
  - Relatively large problem
  - Deals with nested lists / classes
  - Look it over

Lists Within Lists

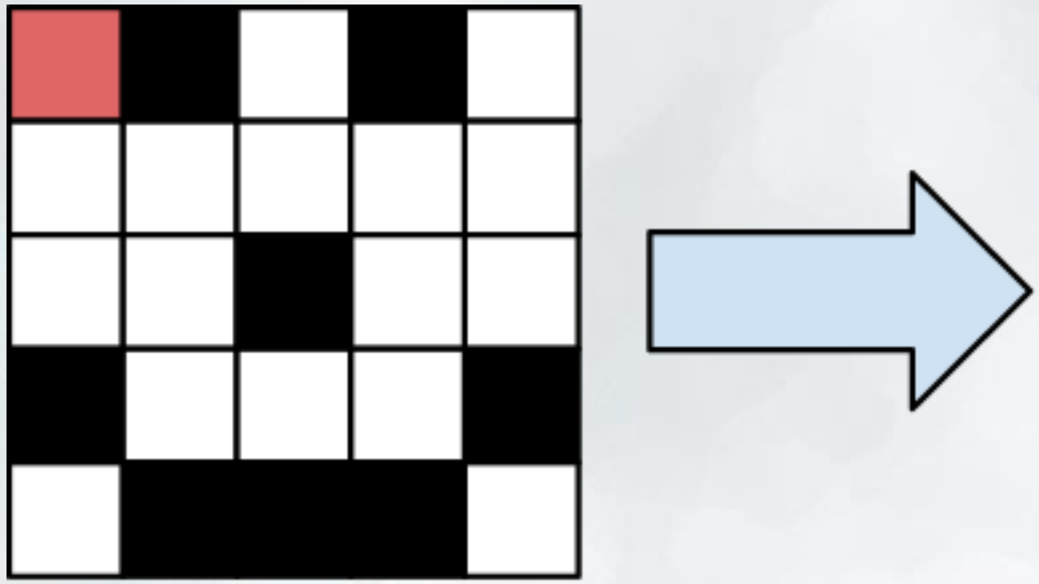
- So far, we've used flat lists
  - Useful for representing a sequence of values
  - Storing a group of things
- What if we want to represent a 2D structure?
  - Pixels in an image
  - Moves in a game of tic tac toe
- Nested lists
  - Represent information on multiple levels

Lists Within Lists



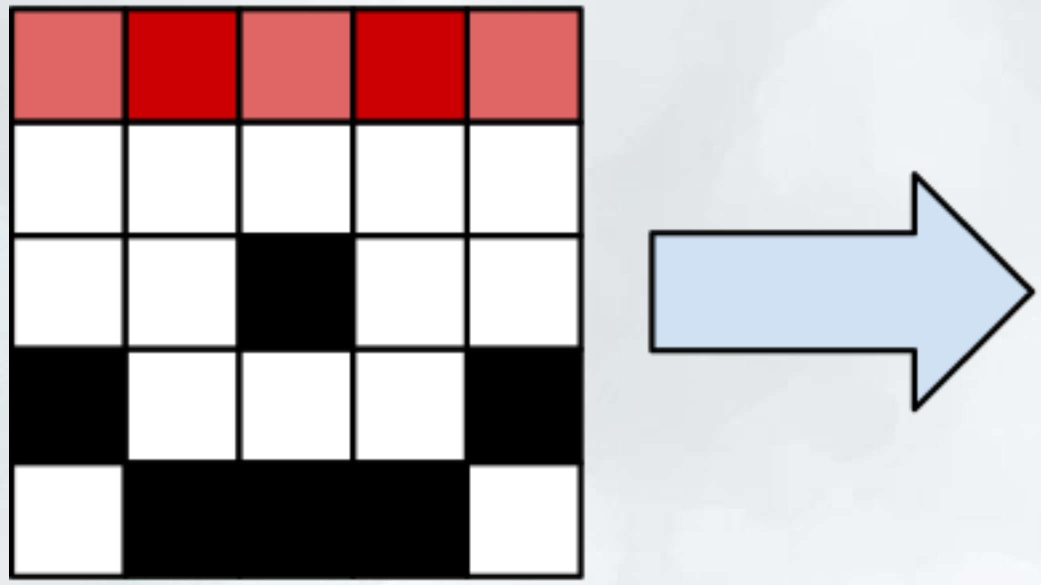
```
[ [ 0, 1, 0, 1, 0 ],
  [ 0, 0, 0, 0, 0 ],
  [ 0, 0, 1, 0, 0 ],
  [ 1, 0, 0, 0, 1 ],
  [ 0, 1, 1, 1, 0 ] ]
```

Lists Within Lists



```
[ [ 0, 1, 0, 1, 0 ],
  [ 0, 0, 0, 0, 0 ],
  [ 0, 0, 1, 0, 0 ],
  [ 1, 0, 0, 0, 1 ],
  [ 0, 1, 1, 1, 0 ] ]
```

Lists Within Lists



```
[ [ 0, 1, 0, 1, 0 ],
  [ 0, 0, 0, 0, 0 ],
  [ 0, 0, 1, 0, 0 ],
  [ 1, 0, 0, 0, 1 ],
  [ 0, 1, 1, 1, 0 ] ]
```

Lists within Lists

- Each element of our nested list is another entire list
    - One row of our picture

- We can access these rows with list indexing

```
bitmap = [ [ 0, 1, 0, 1, 0 ],
           [ 0, 0, 0, 0, 0 ],
           [ 0, 0, 1, 0, 0 ],
           [ 1, 0, 0, 0, 1 ],
           [ 0, 1, 1, 1, 0 ] ]

bitmap[0] → [ 0, 1, 0, 1, 0]
```

Lists within Lists

- Each element of our nested list is another entire list
    - One row of our picture

- We can access individual elements by indexing again

```
bitmap = [ [ 0, 1, 0, 1, 0 ],
           [ 0, 0, 0, 0, 0 ],
           [ 0, 0, 1, 0, 0 ],
           [ 1, 0, 0, 0, 1 ],
           [ 0, 1, 1, 1, 0 ] ]

bitmap[0][2] → 0
```

Lists within Lists

- How large is our nested list?
- How many rows does it have?
- How many columns does it have?
    - Assuming all columns have the same size...

Lists within Lists

- How large is our nested list?
- How many rows does it have?
- How many columns does it have?
    - Assuming all columns have the same size...

```
# Each element in list is a row
numRows = len(nestedList)

# Each row has one element per column
numCols   = len(nestedList[0])
```

Nested List Quiz

```
L=[ [ 1, 2, 3, 4, 5 ], [ 11, 12, 13, 14, 15 ], [ 21, 22, 23, 24, 25 ] ]

print L[0]

print L[2]

print L[0][3]

print L[1][1]

print len(L)

print len(L[1])
```

Looping through Lists

- We can use for loops to iterate through lists
- How would we iterate through a nested list?
    - With nested for loops!
- Iterating by elements:

```
for row in nestedList:
    for element in row:
        < do stuff with element>
```

Looping through Lists

- We can use for loops to iterate through lists
- How would we iterate through a nested list?
  - With nested for loops!
- Iterating by indices:

```
numRows = len(nestedList))
numCols   = len(nestedList[0]))

for row in range(numRows)):
    for col in range(numCols):
        element = nestedList [ row ] [ col ]
        <do stuff with element>
```

Are you in there?

- Let's write a function contains(nestedList, element)
  - Takes a nested list as input
  - Returns True if element is in nestedList
  - False otherwise

Are you in there?

- Let's write a function contains(nestedList, element)
    - Takes a nested list as input
    - Returns True if element is in nestedList
    - False otherwise

```
def contains(nestedList, element):
    """Returns true if nestedList contains element
       False otherwise"""

    for row in nestedList:
        for currElement in row:
            if currElement == element:
                return True
    return False
```

Nested Lists, Assemble!

- Typing out a nested list by hand is tedious

- How might we automatically construct a nested list?
  - Start with an empty list
  - Construct one row
  - Add it to the list
  - Repeat

- How do we construct a row?
  - Start with an empty list
  - Add on element
  - Repeat

- This sounds like a job for nested for loops

Nested Lists, Assemble!

```
def constructNestedList(numRows, numCols):
    """Constructs a nested list containing all 0's
        with given number of rows and columns"""
    nestedList = [ ]                          # Initialize empty nested list

    for row in numRows:
        currRow = [ ]                         # Initialize empty row

        for col in numCols:
            currRow.append(0)                 # Add elements to row

        nestedList.append(currRow)            # Add completed row to list

    return nestedList
```