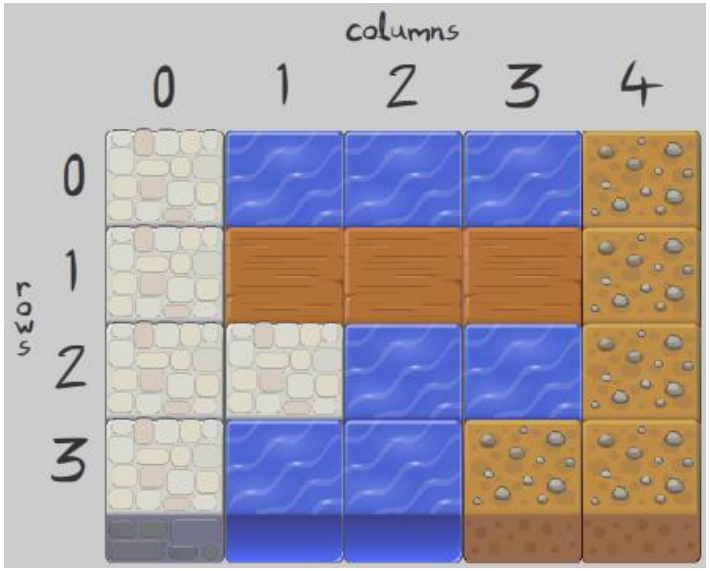
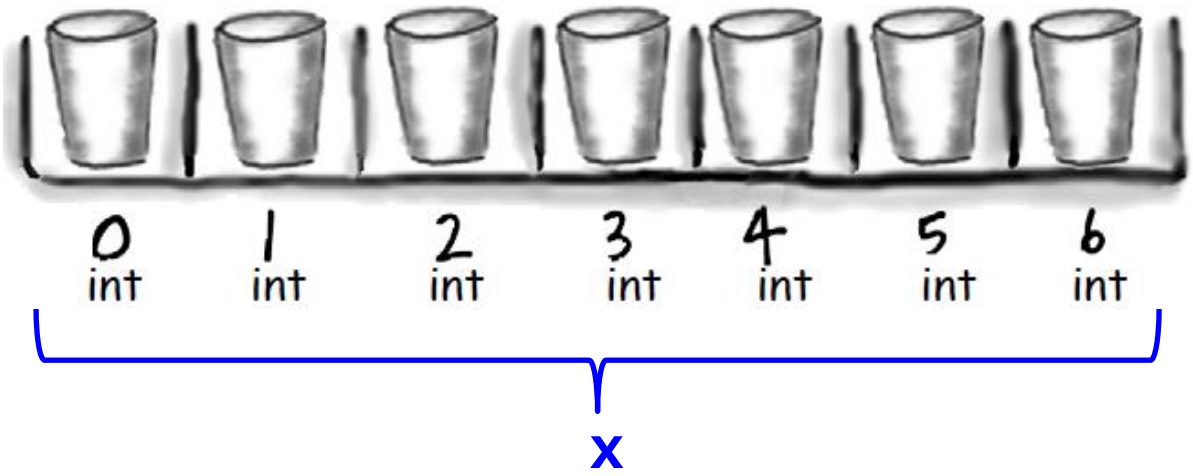
WORKING WITH LISTS



Outline

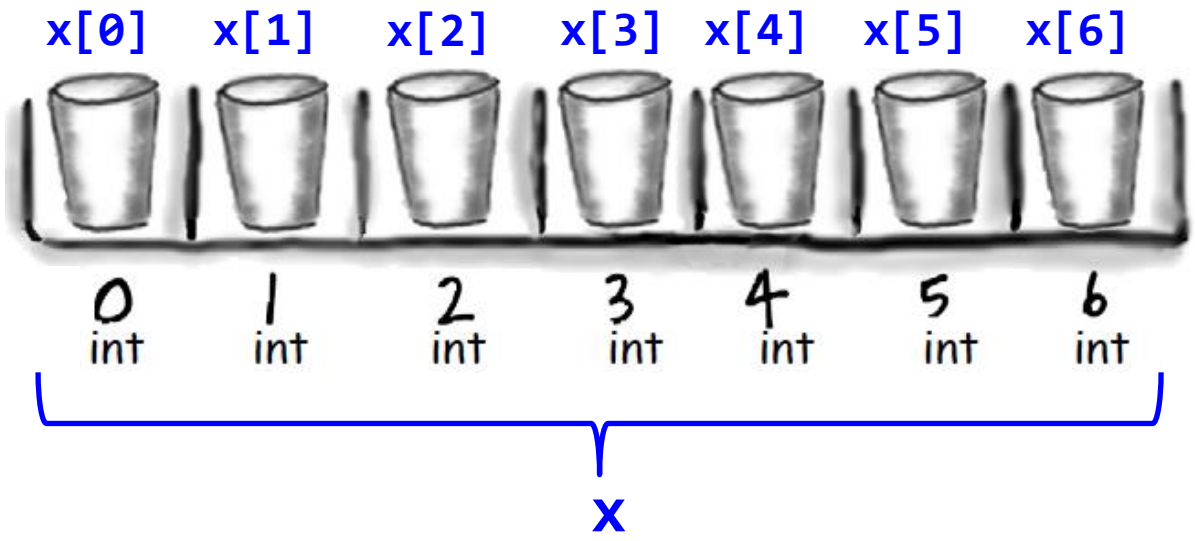
- Operations on Lists
- List Comprehensions
- Slicing a List
- Copying a List
- For Loop Revisited
- Matrices
- Tuples

Lists Revisited



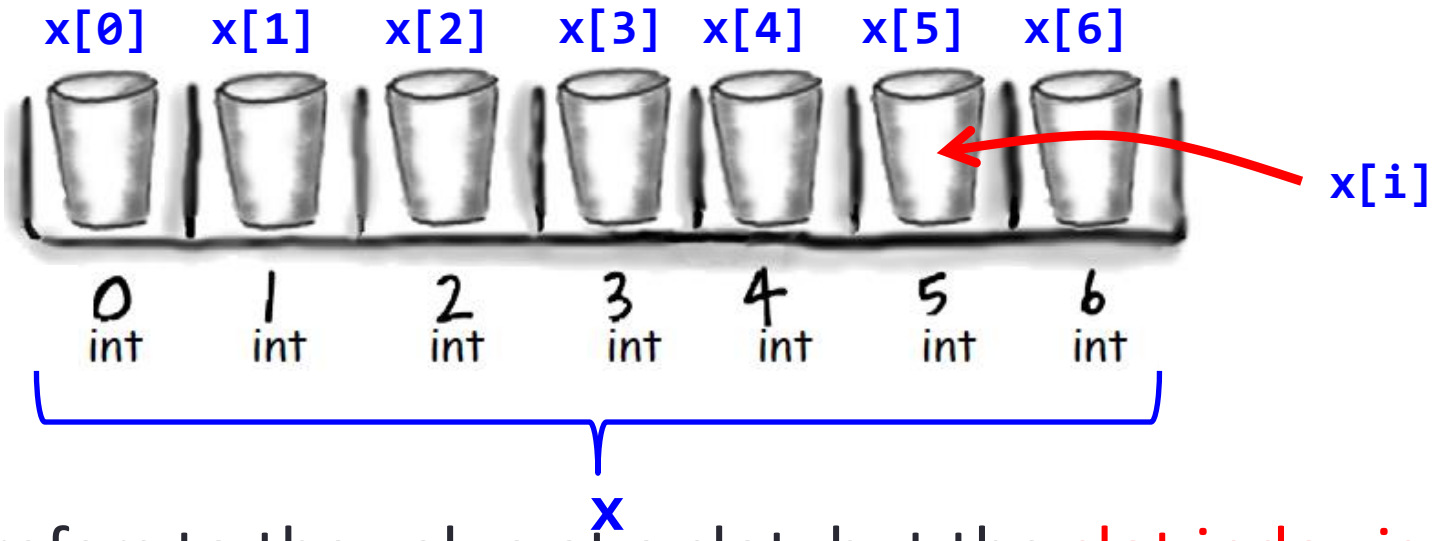
- Variable x refers to the whole set of slots

Lists Revisited



- `x[0],x[1],…,x[6]` refers to value at a particular slot
- `x[7]` = `IndexError`

Lists Revisited



- `x[i]` refers to the value at a slot, but the slot index is determined by variable `i`
  - If i = 0 then `x[0]`, if i = 1 then `x[1]`, etc.
- Whatever inside [] must be an int
- Whatever inside [] must be in 0 to `x.length - 1` (inclusive) OR, in Python, a negative number to start counting from the end of the list

Lists

- Ordered collection of arbitrary objects
- Accessed by offset
- Variable length, heterogeneous, arbitrarily nestable
- Mutable

Slicing a List

- [*start*:*end*+1]
  - [1:4]
  - [:4]
  - [1:]
  - [:]
- Can loop through just a slice (instead of the entire list)

Operations on Lists

- Assignment of Elements
  - L[i] = 3
  - L[i:j] = [4, 5, 6]
- Inserting at a Position
  - append – adds one item to end
  - insert
    - motorcycles.insert(0, ‘ducati’)
- Extend
  - Adds several items
    - L.extend([5, 6, 7])
- Concatenation
  - [1, 2, 3] + [4, 5, 6]
- Repeat
  - [1, 2, 3] * 4

Operations on Lists

- Removing an Element
  - del motorcycles[0]
  - pop
    - motorcycles.pop()
    - motorcycles.pop(0)
  - Remove by value
    - motorcycles.remove(‘ducati’)
- Remove a range of elements
  - L[i:j] = []
  - del L.[i:j]
- Removing all elements
  - L.clear()

Operations on Lists

- Sort
  - cars.sort()
  - cars.sort(reverse = True)
  - sortedCars = sorted(cars)
- Reverse
  - cars.reverse()
  - list(reversed(L))
- Copy – creates a new (separate) copy
  - cars.copy()

Operations on Lists

- Searching
  - L.index(x)
  - L.count(x)
- Membership
  - 3 in L

List Comprehensions

- Generate an operation on every element in a list with a single line of code
  - L = [x**2 for x in range(5)]

For Loop Revisited

- Looping is for more than just working with lists
- We only talked about for loops with numbers
  - They also work with any data type:
    - for magician in magicians:
- Indentation
- Additional lines of code in the block
  - for x in [1, 2, 3]:
    - # do one statement
    - # do another statement
  - Indentation is important – shows how many statements go with the for loop

Tuples

- Tuple looks like a list, except with () instead of []
- ***Immutable***
  - But you can reassign the variable to a new tuple
- Can loop through values in a tuple just like those in a list

Matrices

- Nested lists
  - matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

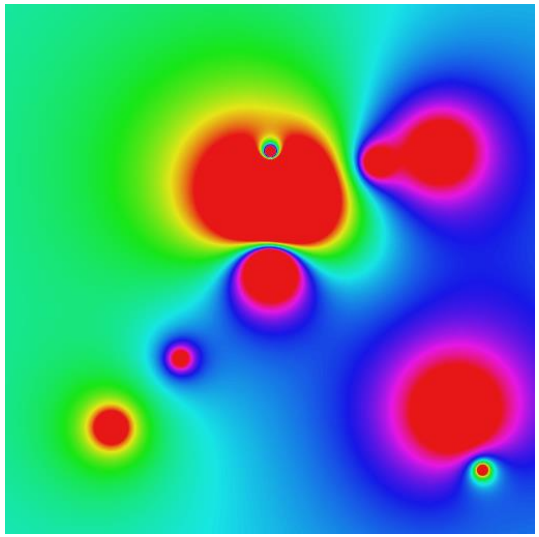
  - # Creates a list containing 5 lists, each of 8 items, all set to 0
  - w, h = 8, 5
  - matrix = [[0 for x in range(w)] for y in range(h)]

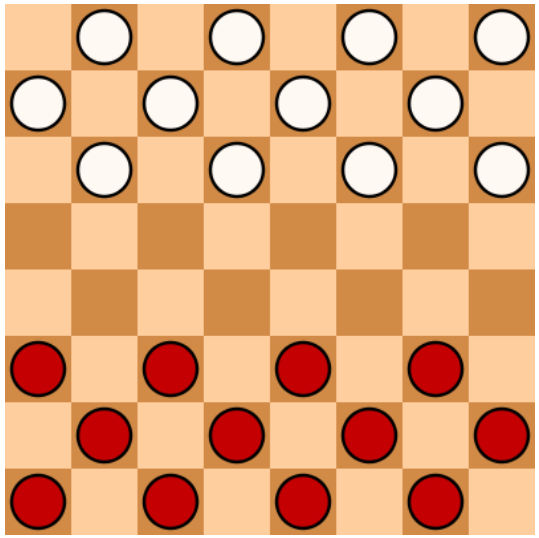
  - import random
  - w, h = 8, 5
  - matrix = [[random() for x in range(w)] for y in range(h)]

Two dimensional list examples

- Two dimensional lists
  - Tables of hourly temps for last week
  - Table of colors for each pixel of a 2D image
  - Table storing piece at each position on a checkerboard

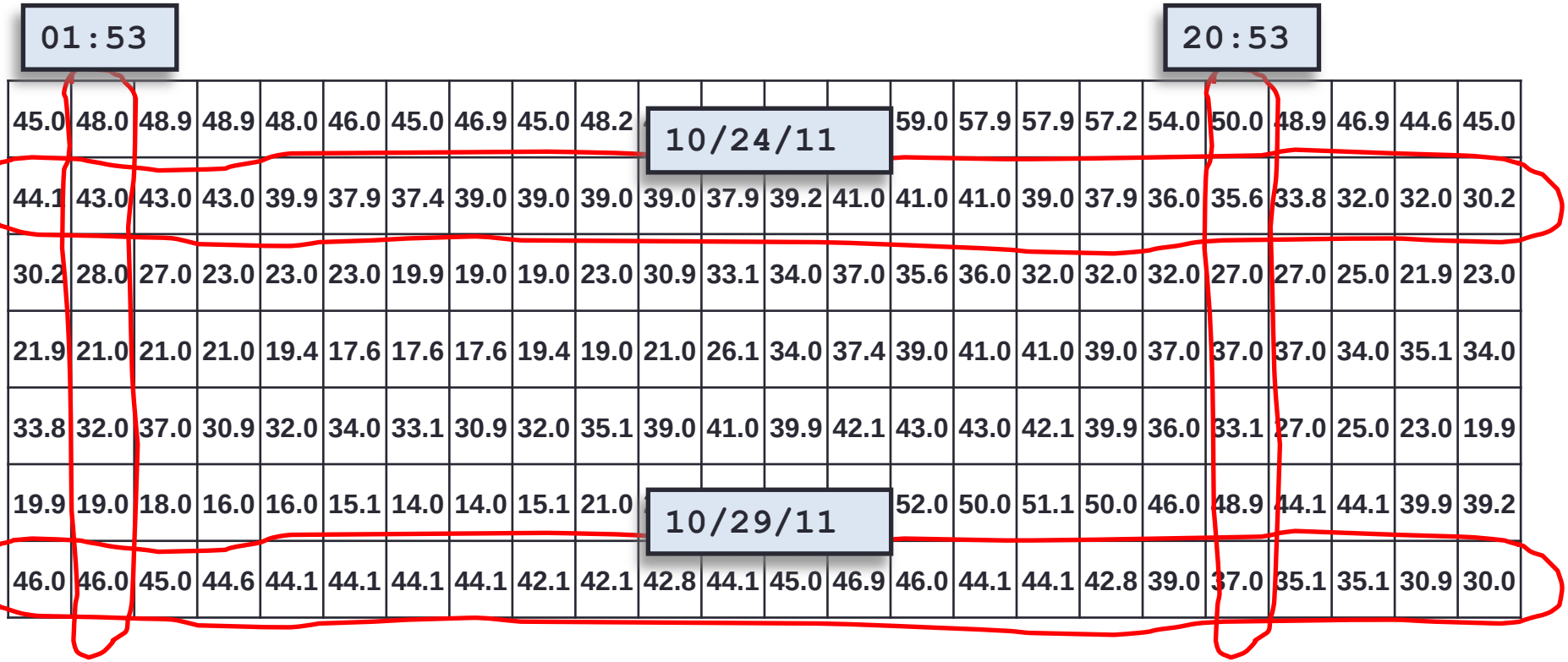
0h	1h	…	23h
32.5	30.0		45.6
…			
59.5	62.1	…	60.0
60.7	61.8	…	70.5
62.6	62.0	…	68.0





Weather data

- Goal: Read in hourly temp data for last week
  - Each row is a day of the week
  - Each column is a particular hour of the day



01:53																	20:53				
45.0	48.0	48.9	48.9	48.0	46.0	45.0	46.9	45.0	48.2					59.0	57.9	57.9	57.2	54.0	50.0	48.9	46.9	44.6	45.0
44.1	43.0	43.0	43.0	39.9	37.9	37.4	39.0	39.0	39.0	39.0	37.9	39.2	41.0	41.0	41.0	39.0	37.9	36.0	35.6	33.8	32.0	32.0	30.2
30.2	28.0	27.0	23.0	23.0	23.0	19.9	19.0	19.0	23.0	30.9	33.1	34.0	37.0	35.6	36.0	32.0	32.0	32.0	27.0	27.0	25.0	21.9	23.0
21.9	21.0	21.0	21.0	19.4	17.6	17.6	17.6	19.4	19.0	21.0	26.1	34.0	37.4	39.0	41.0	41.0	39.0	37.0	37.0	37.0	34.0	35.1	34.0
33.8	32.0	37.0	30.9	32.0	34.0	33.1	30.9	32.0	35.1	39.0	41.0	39.9	42.1	43.0	43.0	42.1	39.9	36.0	33.1	27.0	25.0	23.0	19.9
19.9	19.0	18.0	16.0	16.0	15.1	14.0	14.0	15.1	21.0					52.0	50.0	51.1	50.0	46.0	48.9	44.1	44.1	39.9	39.2
46.0	46.0	45.0	44.6	44.1	44.1	44.1	44.1	42.1	42.1	42.8	44.1	45.0	46.9	46.0	44.1	44.1	42.8	39.0	37.0	35.1	35.1	30.9	30.0

10/24/11

10/29/11

Summary

- Operations on Lists
- List Comprehensions
- Slicing a List
- Copying a List
- For Loop Revisited
- Matrices
- Tuples

Your Turn

- Write a program that creates a two-dimensional list of random floating point numbers between 0 and 1. The dimensions of the list should be 5x6. Print out the values in the 2D list.

- Name your program List2D.py and submit it to the Activity02 dropbox on Moodle. 1 point for turning something in, 2 points for turning in something that is correct.