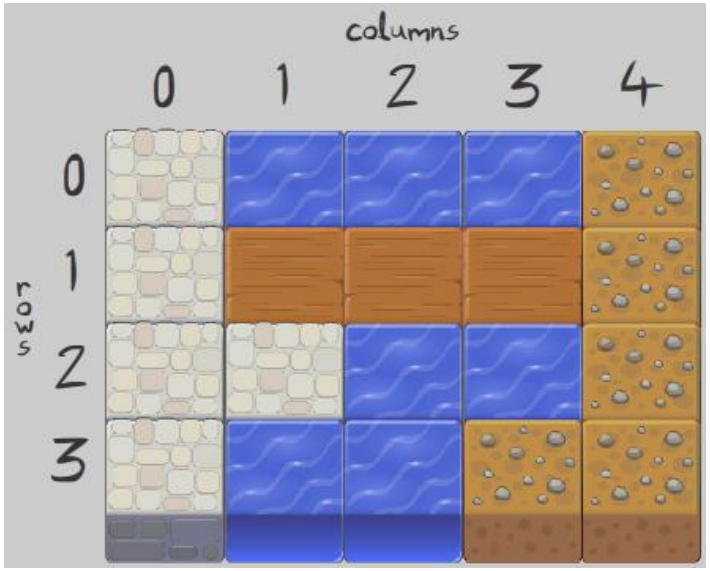
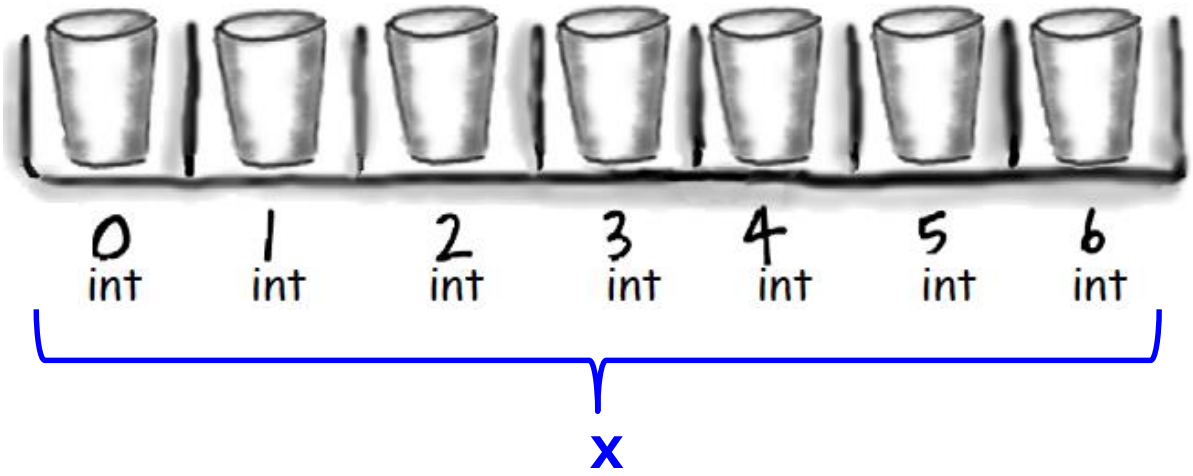
WORKING WITH LISTS



Outline

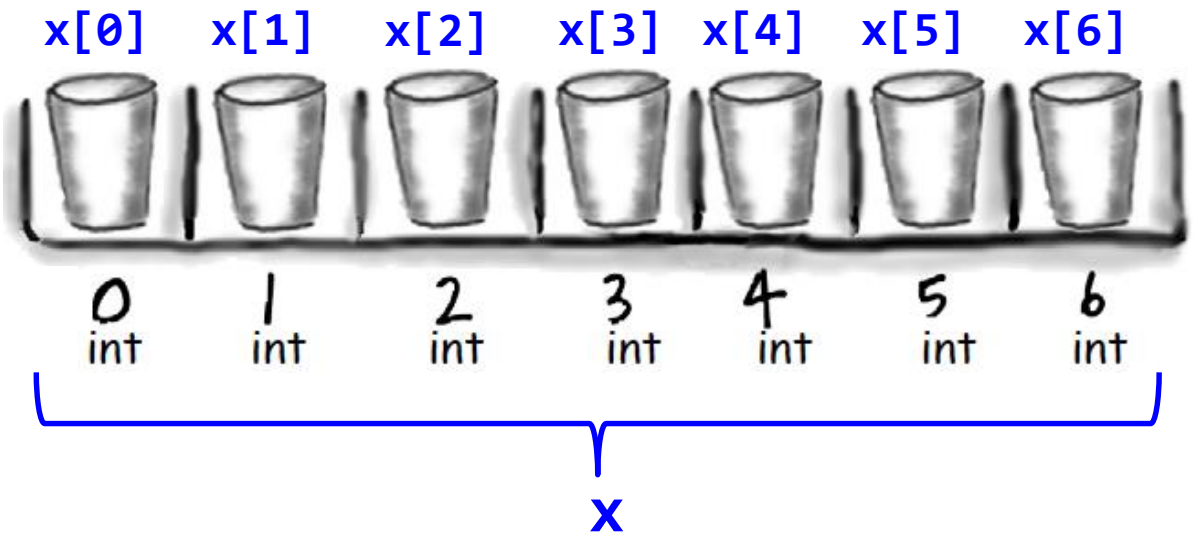
- Operations on Lists
- List Comprehensions
- Slicing a List
- Copying a List
- For Loop Revisited
- Matrices

Lists Revisited



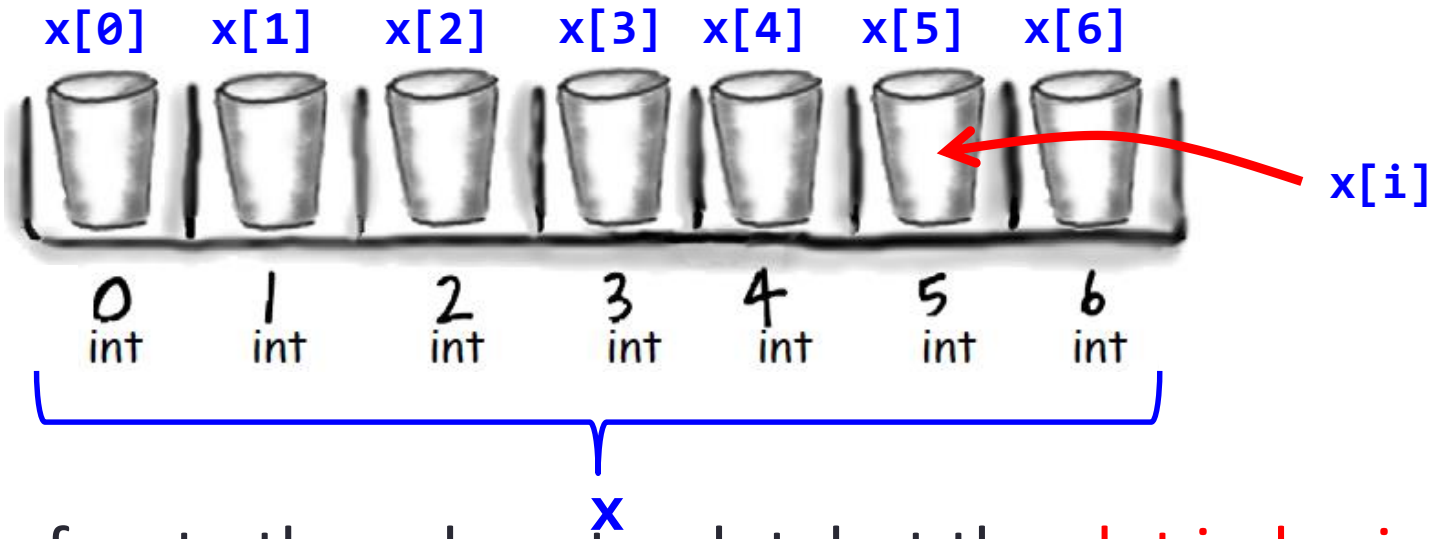
- Variable x refers to the whole set of slots

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 `len(x) - 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 or look at just a slice (instead of the entire list)

Operations on Lists

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

Operations on Lists

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

Operations on Lists

- Searching
  - `index = myList.index(x)`
  - `count = myList.count(x)`
- Membership
  - `isThere = 3 in myList`

List Comprehensions

- Generate an operation on every element in a list with a single line of code
  - `myList = [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

Matrices

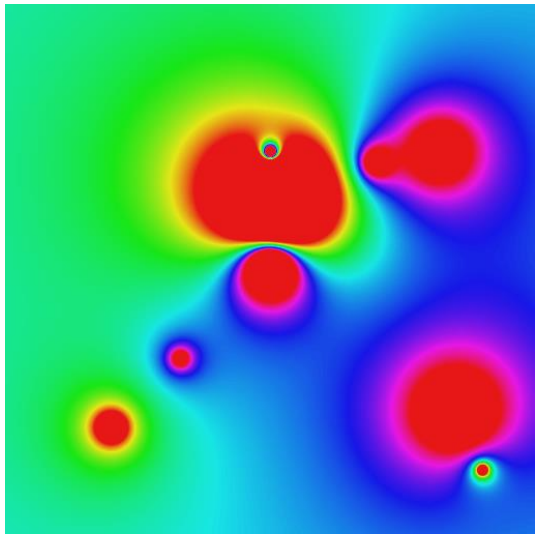
- Nested lists

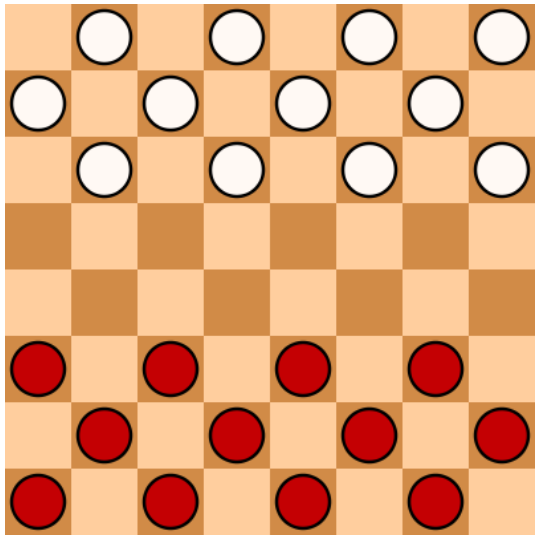
```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

# 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





Summary

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

Let's Try One!

- You have been asked to write two programs – one that will take a single word and an integer as command line input, and encrypt that word using a rotation substitution cipher, and a second program that will take an encrypted word and the number used to encrypt it, and output the decrypted word.

A Caesar cipher, also known as a rotation or shift cipher, is one of the earliest methods of encoding a word or phrase to hide the message from prying eyes. Each letter in the *plaintext* (or original) message is replaced with the letter a certain number of places away in the alphabet to form the *ciphertext*. For example, A becomes Z, B becomes A, C becomes B, and so forth.

Original: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Shifted: Z A B C D E F G H I J K L M N O P Q R S T U V W X Y

Using a shift of one character as above, "HELLO" would become "GDKKN." Caesar and other substitution ciphers are not particularly strong

- For our problem we won't worry about rotating at the end of the alphabet.
- But we do need to know that `ord(<chr>)` returns a number and `chr(<int>)` returns a character