ECS 10

3/4

Announcements

- Midterm grades available probably by end of the week.
- Final will cover similar material.
- Assignments:
  - Makeup program due 3/12
  - Program 6 due 3/17 (Sunday night)
- if you have done OK on all previous programs, you only need to do one of these two assignments. Pick the Makeup if you need more practice with files and dictionaries.

Makeup program

- Popularity of top 1000 baby names
  - Most popular name has rank 1
  - 2nd most popular has rank 2
  - etc.
- Use dictionary to store a list of ranks for each name
- Use canvasPlot module to draw the graph

Today - Under the hood

- How lists and dictionaries really work.
- You don't need to know this to use them.
- But maybe it helps?
- Introduce a good trick you might be able to use someday.

Lists

```
shopping = ["milk", "eggs", "tea"]
print shopping[2]
```

- Items in a list are stored in order.
- Look up an item by its position.
- A list is always indexed by integers, starting with zero.
- An index which is >= the length of the list causes an error.

Dictionaries

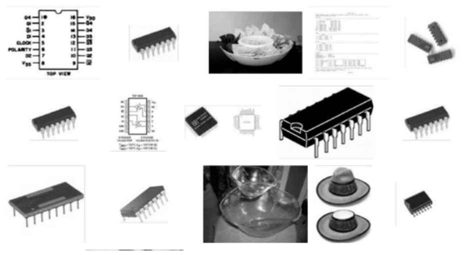
- Values in a dictionary are indexed by keys, which can be anything.
- Items in a dictionary are not in any particular order.
- Looking up a key that is not in the dictionary causes an error; check first:

```
if  "rice" in foodDict:
     print foodDict["rice"]
```

Variable - a labeled memory spot

- The computer memory is like a single big list.
- A variable is a name for one item.

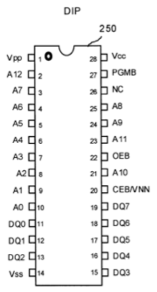
Memory chip



Google image search for "memory chip dip14"

Memory chip

- Has pins for address (number of memory cell where data is stored) A0-A12
- Has pins for data in/out DQ0-DQ7



How lists work

- A Python list is a whole chunk of memory.

But dictionaries?

- Memory is sort of like "list hardware"
- There is no dictionary hardware
- Need to construct a dictionary out of a list
- Trick involving the mod operator
- Phonebook data

Mod operator

```
>>> 8%3
2
>>> 17%6
5
```

- Integer operator
- Produces the remainder when int on left is divided by int on right

Example Problem

- Say I have a file of phone numbers and names.

```
5302204728, "Oswald,  Astrid"
5307547821, "Ortiz, Esteban"
```

- I want to write a program that will let me enter a number, and get back the name.
- Phone number is an integer, name is a string.

Dictionary trick

- A classic CompSci trick called "hashing"

```
# I pick a prime number, larger than the number
# of things I want to store.
# This will be the length of my list.
listLen = 7

poser = [0]*7 # A list pretending to be a dictionary
# Fill it up with zeros
```

Putting stuff in the "dictionary"

```
for name in phoneBook:
    number = phoneBook[name]
    index = number % listlen
    poser[index] = [number,name]
```

- Key idea: compute the index from the key, somehow.

Looking up a number

```
index = number % listLen
if poser[index] == 0:
    print("The phone number is not here.")
else:
    dataList = poser[index]
    if number != dataList[0]:
        print("The phone number is not here.")
    else:
        print("Name is",dataList[1])
```

Strings as Keys?

- This works for integer keys, but how about strings?
- Turn string into a big integer…
- which you use as an index!
- Basic idea: ord() function turns one character into an integer. Compute the index from these integers.

Possible data structures

- Which would be a better data structure:
  - A list of names, indexed by number
  - A dictionary, using the numbers as keys
  - A list of lists [number,name]

Problem with list

- Many possible phone numbers won't have a corresponding person
- If this does not cause an error:
  `L[5302208945]`
  then the length of the list has to be >= 5302208945.
- **Takes up a huge amount of memory.**

Problem with list of lists

- Might have to read the whole list to find the phone number we want.
- So it is slow at answering queries (if there is a lot of data).

Dictionary is best

…for this problem, anyway.

- Length of dictionary is the number of items in it, not the size of the biggest key.
- You can access items using the key, not by looking through the whole data structure.
- Even though the keys are integers, if lots of possible keys are not used, then a dictionary still works best.

When is a list a better choice?

- When order is important.
- We can sort lists, but not dictionaries.
- Dictionaries are always in some weird arbitrary order.

Dictonary vs list, take 2.

- Dictionaries are a little slower, but not much.
- Dictionaries are a litte bigger, but not much.
- Dictonaries have to be in "random" order to work properly.