Announcements
- Program 5 due tomorrow.
- No class Wednesday (Veteran’s day).
- Midterm review Friday.
- Midterm 2 Monday.
- Today – when to use what data structure, and a bit about how dictionaries work.

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:
  ```python
  if "sparrow" in petDict:
    print petDict["sparrow"]
  ```

for loop with a list
- shopping = ["milk", "eggs", "tea"]
- for product in shopping:
  - print "We’re out of", product
- Variable product takes on the values of each of the items in the list in turn.
- Block under the for statement is executed three times.
- No index variable.

for loop with a dictionary
- for pet in petDict:
  - print pet, petDict[pet]
- In the first line, the variable pet becomes on each of the keys in turn.
- pet is the index variable.
- We use pet as the key, to look up the values.
- The keys appear in no particular order.
len() function works with both

```python
# Number of items in dictionary D
print len(D)

# Number of items in list L
print len(L)
```

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.

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 SID numbers won’t have a corresponding person
- If this does not cause an error:
  ```python
  L[5302208945]
  ```
  then the length of the list has to be $\geq 5302208945$.
- 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 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.

Why is this?

- Why can’t we sort dictionaries?
- Are lists or dictionaries faster on big data?
- Does one or the other take up less space?
- It might help to understand how dictionaries work.
- Also it is nice to know they are not magic!

How lists work

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

Making a dictionary out of a list

- A classic CompSci trick called “hashing”

```python
# 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 = [] # A list pretending to be a dictionary
# Fill it up with zeros
for i in range(listLen):
    poser = poser+[0]

for name in phoneBook:
    number = phoneBook[name]
    # Compute the index from the number!
    index = number % listLen
    poser[index] = [number, name]
```

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

Putting stuff in the “dictionary”

```python
for name in phoneBook:
    number = phoneBook[name]
    # Compute the index from the number!
    index = number % listLen
    poser[index] = [number, name]
```

A Python list is a whole chunk of memory.
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 \ 
        "The phone number belongs to", dataList[1]
```

Under the hood

- This is how dictionaries really work.
- You don’t need to know this to use them.
- But maybe it helps?
- A good trick you might be able to use someday.

Dictionary vs list, take 2.

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

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.