Data Abstraction

UW CSE 190p

Summer 2012

Recap of the Design Exercise

- You were asked to design a module – a set of related functions.
- Some of these functions operated on the same data structure
  - a list of tuples of measurements
  - a dictionary associating words with a frequency count
- Both modules had a common general form
  - One function to create the data structure from some external source
  - Multiple functions to query the data structure in various ways
  - This kind of situation is very common

Text Analysis

```
def read_words(filename):
    “””Return a dictionary mapping each word in
filename to its frequency in the file”””
    wordfile = open(filename)
    worddata = wordfile.read()
    words = worddata.split()
    wordcounts = {}
    for w in words:
      cnt = wordcounts.setdefault(w, 0)
      wordcounts[w] = cnt + 1
    return wordcounts

def wordcount(wordcounts, word):
    “””Return the count of the given word”””
    return wordcounts[word]

def topk(wordcounts, k=10):
    “””Return top 10 most frequent words”””
    scores_with_words = [(s,w) for (w,s) in wordcounts.items()]
    scores_with_words.sort()
    return scores_with_words[0:k]

def totalwords(wordcounts):
    “””Return the total number of words in the file”””
    return sum([s for (w,s) in wordcounts])
```

Quantitative Analysis

```
import matplotlib.pyplot as plt

def read_measurements(filename):
    “””Return a dictionary mapping column names to data. Assumes 
the first line of the file is column names.”””
    datafile = open(filename)
    rawcolumns = zip(*[row.split() for row in datafile])
    columns = dict([(col[0], col[1:]) for col in rawcolumn
    return columns

def tofloat(measurements, columnname):
    “””Convert each value in the given iterable to a float”””
    return [float(x) for x in measurements[columnname]]

def STplot(measurements):
    “””Generate a scatter plot comparing salinity and temperature”””
    xs = tofloat(measurements, “salt”)
    ys = tofloat(measurements, “temp”)
    plt.plot(xs, ys)
    plt.show()

def minimumO2(measurements):
    “””Return the minimum value of the oxygen measurement”””
    return min(tofloat(measurements, “o2”))
```

What we’ve learned so far

- data structure
  - a collection of related data
  - the relevant functions are provided for you
  - Ex: list allows append, sort, etc.
  - What if we want to make our own kind of “list,” with its own special operations?

- module
  - a named collection of related functions
  - but shared data must be passed around explicitly
  - What if we want to be sure that only our own special kind of list is passed to each function?

Data abstraction

- Data structures can get complicated
- We don’t want to have to say “a dictionary mapping strings to lists, where each list has the same length and each key corresponds to one of the fields in the file.”
- We want to say “FieldMeasurements”
- Why?