More on Lists, Dictionaries
Introduction to File I/O

CS 8: Introduction to Computer Science
Lecture #10

Ziad Matni
Dept. of Computer Science, UCSB

Administrative

Tutoring/Review Session Available!

- TWO of them again!
- Friday, **5/12** at **1:00 PM & 2:30 PM** in **PHELPS 3525**

- Please note:
  T.A. Sourav's office hours are now
  Thursday: 10AM to 12PM

- **Midterm #2 is next week on Thursday 5/18!**

REVIEW

Lists

- Use square brackets, `[ ]` to define a list

```
fruit = ['apple', 'pear', 'orange',
  'lemon']
```

- And use `[ ]` to access elements too

```
fruit[2]    >>> 'orange'
```

  – Indexing works the same as strings
    - i.e. start with [0]

REVIEW

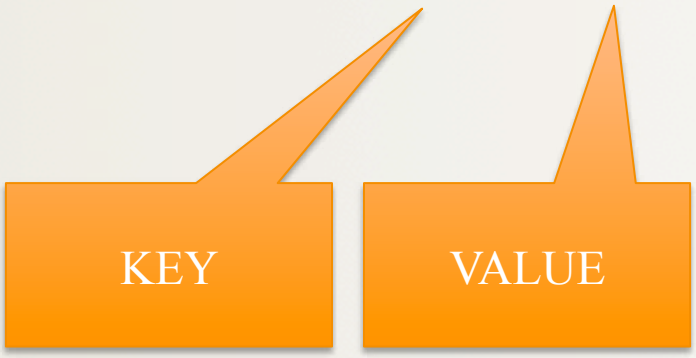
Dictionaries

- Unordered *associative* collections
  - Basically lists,

    but you can access each value by a **key**

    instead of an index position
- Use curly braces, `{ }` to define a dictionary

```
ages = { 'sam':19, 'alice':20 }
```



NOTE THE SYNTAX and the use of the colon

key:value

Measuring Dispersion

- How much do values *vary* from the average?
- Differences from mean: **`x[i] - mean(x)`**
  - Includes positive and negative differences
  - So usually square difference: `(x[i] - mean(x))**2`
- Variance:
  The sum of squared differences (for all **`i`**), divided by **`n - 1`**
- Standard deviation = $sd = \sqrt{\frac{\sum_{i=0}^{n-1}(x[i] - mean(x))^2}{n-1}}$, or square root of variance

In Python

```
import math
def sd(alist):
    theMean = mean(alist)

    total = 0
    for item in alist:
        diff = item - theMean
        diffsq = diff ** 2
        total = total + diffsq

    sdev = math.sqrt(total/
(len(alist)-1))

    return sdev
```

```
def mean(alist):
    return(sum(alist)/len(alist))
```

Histograms

- A very popular statistical analysis tool
- A plot of how often (i.e. frequency) a data point appears in a set

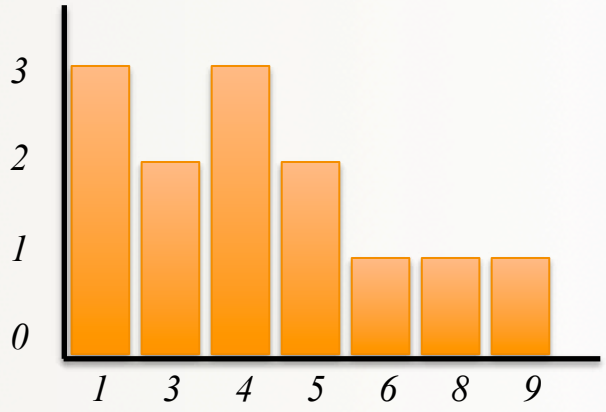
Example

Data set is: **1 3 4 1 1 5 9 8 5 6 4 4 5 3**

Frequency table:

No.	Frequency
1	3
3	2
4	3
5	2
6	1
8	1
9	1

Histogram Plot:



How Can I Do This in Python?!

- **Given**: you start with a list of numbers
- **Goal**: you have to print out a frequency table

- **Hint**: We tackled counting how often numbers occur in lists before!
  - Using dictionaries

Printing a Frequency Table

- Easiest done with a dictionary (see Listing 4.8):

```
countdict = {}
for item in alist:
    if item in countdict:
        countdict[item] = countdict[item] + 1
    else:
        countdict[item] = 1
itemlist = list(countdict.keys())
for item in sorted(itemlist):
    print(item, "\t", countdict[item])
```

The One We Came Up With in Class!

```
def freqTable(alist):
    countD = {}
    for i in alist:
        if i not in countD:
            countD[i] = 1
        else:
            countD[i] = countD[i] + 1
    keylist = list(countD.keys())
    keylist.sort()
    print("Item", "\t", "Frequency")
    for x in keylist:
        print(x, "\t", countD[x])
```

How About Plotting That Table?

- The textbook has an excellent example (listing 4.10).
- I will leave this as an exercise for you to do in an upcoming lab!

Starting Chapter 5

Files

- Mostly handle like any sequential data type
- A sequence of characters if a text file,
  or a sequence of bytes if a binary file
- Can you name some file *types* that are textual?
  Or binary?

Why Use Files?

4 Good Reasons:

- Files allow you to store data permanently and conveniently!
- Data output to a file lasts after the program ends
  - You can usually view them without the need of a Python program
- An input file can be used over and over
  - No typing of data again and again for testing
- Files allow you to deal with larger data sets

Files

- First, **open** file, and state purpose – **r**ead or **w**rite

```
ThatImportantFile = open('mydata.txt', 'r')
MyPrettyOutputs   = open('myresults.txt', 'w')
```

- Often process text files as a sequence of lines

```
for line in inputFile: # process each line as a string
    outputFile.write(line[:5] + '\n')
```

- Best to **close** the files when you're done

```
inputFile.close()
outputFile.close()
```

DEMO!
Let's try it!

More Ways To Read A File

- Already saw: `for line in file` – to process each line as a separate string (inc. `'\n'` at ends)
- To get just a *single* line (as **string**): `file.readline()`
  - Do it again to get the next line, and so on
- Also can get a **list** of lines as strings by `file.readlines()` – including `'\n'` at ends
  - Note readline**s** vs readline

DEMO!
Let's try it!

More Ways To Read A File

- You can also just `file.read()` – to get ***all*** of the file's text as a single string

- Note: use `open` again if want to go back to the beginning of a file and read from start

DEMO!
Let's try it!

YOUR TO-DOs

- We're just starting **Chapter 5**, so read ahead

3 THINGS TO FINISH THIS WEEK!!!

- Finish **Homework5** (due **Thursday 5/11**)
- Finish **Lab4** (due **Tuesday 5/9**)
- Keep working on **Project1** (due **Friday 5/12**)
  - Lab time tomorrow is for working on Project1

- Sing like you mean it

</LECTURE>