CS151 Fall 2012 Lecture 29

Stephanie R Taylor

November 12, 2012

1 Administrative Topics

- Return graded quizzes
- Stephanie's office hours Monday and Tuesday must be revised. I will be available from 1 to 4 Monday and from 1 to 3 on Tuesday.
- You will get your project grades by Tuesday.

2 Advice about Project Write-ups

Part of the purpose of the write-up is to give you an opportunity to gain a deeper understanding of the overarching structure and purpose of the code. Explaining the project to a peer (not in the course) is a good way to force yourself to step back and think about the project. So we want you to write for that audience. That means you should continue to include code snippets and images, and L-system rules as you have been doing. But be sure to provide context for those details. Don't just dive into the details.

For this week, you should spend a short(ish) amount of time explaining L-systems. Remind your reader what they are (you can assume the reader has read the last few write-ups) and then go on to explain the new feature (stochasticity). Then you can spend more time on the design of the Shapes class and the classes derived from it. The Tree class is particularly interesting

because it is derived from the Shape class and it contains and Lsystem object. It brings it all together!

3 Python is Powerful when it comes to Parsing

Something inevitable in computer science is file-parsing. We have done some (e.g. when we read in the L-system description files). Fortunately, Python provides some powerful data types to help us with that. For example, file.readlines(), str.split(), str.strip(), str.lower(), and dictionaries. My ultimate goal for today's lecture is to develop a class that reads in a text file and counts the frequency of all the words in it. The class's methods should allow us to get information about which words appear the most, about the total number of unique words, and about the frequency of any particular word.

And since today is Veterans' Day, I think it would be fun to study the Constitution of the U.S.A.

3.1 List Sorting

But before we can begin, we need to learn about another powerful method – list.sort(). There is a method that will let us sort the items in a list. And it is super flexible – it will let us do so using any "rule" we want. In other words, we can supply the function that determines the relative order of any two elements.

But before I get ahead of myself, let's look at the sort method on a list of numbers:

```
nums = [3,2,5,6]
print nums
nums.sort()
print nums
```

This makes sense. The items in the array have a natural ordering, and Python just uses that.

What about lists of more complicated data? e.g. tuples with a string and a number.

```
mixed_list = [ ('a',4), ('c',2), ('b',1)]
```

How do we put this list in order? Can we simply say mixed_list.sort()? We could, but how does Python decide which duple should be first? We need to tell Python how to order them they way we want them ordered. Sort can take as input a comparison function. This comparison function should take two inputs and return -1 if the first element belongs earlier in the list than the second element, 0 if they are the same, and 1 if the first element belongs later in the list. Let's write a comparison function that orders the elements according to the first element in the tuple

```
# a comparison function that uses the second item in a duple.
# returns -1 if a comes before b
# returns 1 if a comes after b
# returns 0 if they are the same
def compare_duples2(a,b):
    if a[1] < b[1]:
        return -1
    elif a[1] == b[1]:
        return 0
    else:
        return 1
```

Then, let's sort the list, supplying the comparison function.

```
mixed_list.sort(compare_duples2)
print mixed_list
```

Now, let's reverse the order:

```
mixed_list.sort(compare_duples2,reverse=True)
print mixed_list
```

3.2 Counting Words

Let's make a class called WordCounter that counts the words in a document and which supplies methods that will allow us to examine or print that information. E.g. It should have this structure

```
# Count the word frequency of the words in the given text document
```

```
class WordCounter:
    # Create a word counter for the given document
    def __init__(self, filename):
        ~~~~~~~~

    # dump out to the screen the counts for all words
    # appearing more than once
    def dump(self):
        ~~~~~~~~

    # print out the words and counts for the numItems
    # most frequently occurring words
    def seeTop(self,numItems=10):
        ~~~~~~~~

    # return the number of unique words in the document
    def getNumWords(self):
        ~~~~~~~~

    # return the number of times the given word appears
    # or None if it doesn't appear
    def getCount(self, word):
        ~~~~~~~~
```

To count the words in a text document, we need to

- Read the lines from the file
- For each line
  - Separate it into words
  - For each word
    - Update the counter for that word

How do we store counters for each word? We can use a dictionary where the key is the word and the value is the number of times it occurs in the document.

So our updated algorithm is:

- Read the lines from the file

- Make an empty dictionary
- For each line
  - Separate it into words
  - For each word
    - Update the counter for that word (i.e. add an entry if this is the first time we have seen it, or update its value if it is already there)

We write the code together. There are a few additional details which our code must take care of:

- We want to be case insensitive, so we should put every word in the same case (e.g. lower case) before we update the dictionary.
- We want to remove punctuation (e.g. commas, periods, exclamation points, etc.) replace any of those characters with spaces.

```
# Count the word frequency of the words in the given text document
class WordCounter:
    def __init__(self, filename):
        # we are going to store the counts in a dictionary
        # where the key is the word and the value
        # is the number of times that word appears in the text
        self.word_count = {}

        # read the file into a list of strings
        f = file( filename, 'r' )
        lines = f.readlines()
        f.close()

        # go thru line by line, adding words (and counts) to
        # the self.word_count dictionary
        for line in lines:
            # prettify it
            line = line.lower()
            clean_line = ''
            for c in line:
                if c in [',','.','/','"',"'",'(',')',':',';','?','!','@','#','$',\
                    '%','^','&','*','[',']','{','}','<','>','-','`']:
```

```
                c = ' '
            clean_line += c
        clean_line = clean_line.strip()
        words = clean_line.split()
        #print words

        for word in words:
            if word not in self.word_count:
                self.word_count[word] = 0
            self.word_count[word] += 1
        #print self.word_count

# dump out to the screen the counts for all words
# appearing more than once
def dump(self):
    for key in self.word_count.keys():
        if self.word_count[key] > 1:
            print key + " " + str(self.word_count[key])

# print out the words and counts for the numItems
# most frequently occurring words
def seeTop(self, numItems=10):
    duple_list = self.word_count.items()
    duple_list.sort( compare_duples2 )
    print duple_list[-1:-numItems:-1]
```

Note that in class we used negative indexing (in seeTop) to access the last elements of the duple list. We could have told the sort method to sort in reverse order, but we didn't. :)

In class, we did not get to the final two methods. But here is code for them:

```
# return the number of unique words in the document
def getNumWords(self):
    return len(self.word_count)

# return the number of times the given word appears
# or None if it doesn't appear
def getCount(self, word):
    return self.word_count.get(word.lower())
```

Here is the testing code:

```
wc = WordCounter('constitution.txt')
wc.seeTop()
print "there are "+str(wc.getNumWords())+" unique words"
```

```
print "congress appears "+str(wc.getCount('congress'))+" times"
```