Homework Week 4:

Problem 1:

You are given a string such as `"temp/babynames/baby2004.html"`. Obviously, this string represents a file name that you obtain if you unzip the babynames.zip folder. You need to write a function `get_year(file_name)` that retrieves the year (the four digits before the dot). The only thing that you can rely on is that the year is always in the four digits before the dot.

Problem 2:

Use the module os and the function os.listdir to write a function that takes the name of a directory and returns a list of all files that are `.html` files. (You can use endswith).

Problem 3:

In the zip folder babynames.zip, you will find a number of html files downloaded from the Social Security website with information on the popularity of baby-names in Social Security applications. Normally, you would use an agent to automatically obtain these pages, usually at a slow rate so that you will not be penalized by throttling your requests.

If you look at the files in a text editor (clicking on it will just show them to you in your standard browser), you will find them full of html code that you might have difficulties in understanding. However, if you move down, you will soon see the gist of the page, namely the list of ranks and names, one for each gender. Here is an example from 1988.

```
<td>1</td> <td>Michael</td> <td>Jessica</td>
```

As you can see, all interesting lines have this pattern. After stripping white spaces, the lines of interest and only they start with the string “<td>”. You can therefore read all lines, and if they start with “<td>”, then you can process them. Notice that the white spaces between the column markers “<td>” and “<\td>” make it easy to break this string up into three different parts.

Your first task is to write a function `process(file_name)` that takes the file name of one of the babyxxxx.html files as input and prints out the rankings of the baby-names in that year. Here is a sample for 2004.

```
1 Jacob Emily
2 Michael Emma
3 Joshua Madison
4 Matthew Olivia
5 Ethan Hannah
6 Andrew Abigail
7 Daniel Isabella
```

Problem 4:

We want to be able to give statistics on the popularity of names over the years. We process the files in alphabetical order, which is also the chronological order. When you process a file, you obtain a ranking “1” and two names, “Michael” and “Jessica”. You convert the ranking from a string to an integer. You then add to two dictionaries, one for male and one for female names. The keys to the dictionaries are the names and the values is a list of rankings. Therefore, processing this entry gives us

```
dmale[“Michael”].append((1988,1))
dfemale[“Jessica”].append((1988,1)).
```

Write a function that creates the dictionaries by processing all the file-names in the babyname directory.

After you create a dictionary with the lists, you can now answer user questions. The user types in a name and your program consults the dictionaries and gives the rankings of the name per year. (You might find it handy to sort the list of tuples and find them ordered by the first item).