Computer Science CS134 (Spring 2020)

Shikha Singh & Iris Howley

Laboratory 7

Setting Precedent in the Supreme Court (due Thursday April 16 at 11pm EST)

Objective. Creating and using a class.

Lab overview video. The following videos provide an overview to this lab assignment:
`https://williams.hosted.panopto.com/Panopto/Pages/Viewer.aspx?pid=556f2a40-62d4-4d7d-8b50-ab960141af3c`.

This week, we'll look at one of the most carefully curated social networks—the network of Supreme Court majority decisions. In this network, the majority decision of a case is founded in the law by *citing* past decisions. Past decisions establish *precedent* by being cited by later cases.

A similar network is the network of academic publications. Publications cite prior work to establish legitimacy, and older publications become influential when they're cited by future work.

In academic circles, an author's influence can be estimated by *academic impact scores*. In this lab we'll attempt to apply academic impact scores to dockets of Supreme Court decisions.[1]

Background: The h-Index. A commonly used metric for determining the impact of authors is the *h-index*. The index is computed by counting the number of times each of the author's papers have been cited. Suppose Rita DeCoder has has written ten papers. Her list of citation counts might look something like this: `(0, 2, 15, 9, 7, 48, 4, 82, 14, 6)`. In particular, Rita's first paper has been cited zero times, her second paper has been cited twice, etc. Rita's h-index is *the maximum* n *where her top* n *papers have been cited at least* n *times*. Looking at Rita's citation counts we see that her top 6 papers have been cited at least 6 times.

The process is more obvious when you sort the citation count list in descending order: `(82, 48, 15, 14, 9, 7, 6, 3, 2, 0)`. It's easy to see that the first six elements are greater than or equal to 6, but the 7th is less than 7. We'll leave it to you to determine the relationship between the *values* of the list and their list *indices*. Authors with a high h-index have written many highly-cited papers and, over time, their impact score may improve as more and more people cite their works.

Getting Started. As usual, you should clone the starter repository for this week's lab, similar to the process you followed for last week's `Lab06 Remote Set-up` lab. We typically do this by going to `https://evolene.cs.williams.edu` and clicking `Clone` and selecting the `Copy URL to clipboard` button under the `Clone with HTTPS` text. Return to the `Terminal`, navigate to your cs134 directory, and type `git clone` followed by the URL you just copied, followed by the name of the lab, e.g.:

```
git clone https://evolene.cs.williams.edu/cs134-s20/lab07/22xyz3.git lab07/
```

where your CS username replaces 22xyz3. We hope to plot data, so you will need to install matplotlib, click on this link for instructions on how to install it.

You will the find the file, `scotus.py`, along with several CSV files in the lab folder. The primary CSV file we will be using is `decisions.csv`. This file contains data that has been collected from Fowler and Jeon's interesting analysis of the 30,288 majority US Supreme Court decisions on dockets through 2002.[2]

[1] Thanks to Peter Tianlun Zhang for performing this initial research and to Carl Rustad helping develop this lab.

[2] J. H. Fowler, S. Jeon, "The authority of Supreme Court precedent", *Social Networks*, 30(1), January 2008, pp. 16-30. See also `fowler.ucsd.edu/judicial.htm`. The raw data used for their work is in `judicial.csv`.

Each row in this file corresponds to an annual docket of Supreme Court decisions. The first item on each line is the year the decisions were made, which is followed by a sequence of *citation counts* of cases heard during that year (in the order they were heard by the court). For example, line 18: `1784,1,0,1,0,0,0,0,0,1,0` represents the *docket* of cases heard by the Supreme Court in the year 1784 during which the first, third and ninth were cited once, while the rest were cited zero times.

This week's tasks. Before we get to the main tasks for this week, we need you to do a one-time `git config` setup on your personal machines.

Task 0. `git config` **setup.** Execute the following commands (if you have not done so already) in the Terminal/Command Prompt application. They will ensure that your `git commits` and pushes have your `user name` and `email` associated with it.

```
git config --global user.name "22xyz3"
git config --global user.email "xyz@williams.edu"
```

where your CS username replaces 22xyz3 and the email is your Williams email.

In the file `scotus.py`, you will find an incomplete definition of a class, `ScotusDocket`. The objects of this class represent a docket of the US Supreme Court cases with two private attributes: `_label` (which can refer to a particular year as `int` or the name of a Chief Justice as `str`) and `_citations` (sequence of integer citation counts for that year or Justice as a tuple). First, we will implement several methods for this class, and then use the `ScotusDocket` objects to read in and analyze the decisions data.

Task 1. Implementing methods. Complete these method definitions in the `ScotusDocket` class:

(a) `__init__(self, label=None, citations=(0,))`. This method is invoked when we create an object of the class `ScotusDocket` and initializes the object's attributes `_label` and `_citations` with the arguments `label` and `citations` passed as arguments during the creation. The default label is set to `None` and the default citation sequence is the singleton-tuple `(0,)`.

(b) `label(self)`. This is a public accessor/getter method which when invoked returns the calling object's `_label` attribute. Notice the `@property` annotation preceding the method definition. This ensures that we can invoke the method as if it is a data attribute, e.g.:

```
>>> d1 = ScotusDocket(1600, (1, 2, 3))
>>> d1.label
1600
>>> ScotusDocket('R.G.B', (0, 2, 15, 9, 7, 48, 4, 82, 14, 6)).label
'R.G.B'
```

(c) `citations(self)`. This is a public accessor/getter method which when invoked returns the calling object's `_citations` attribute. Notice the `@property` annotation preceding the method definition.

```
>>> ScotusDocket('John Jay', (4, 82, 14, 6)).citations
(4, 82, 14, 6)
>>> ScotusDocket(1754, (0,)).citations
(0,)
```

(d) `_hIndex(self)`. This private method when invoked returns the *h-Index* of the calling object's citation counts. Review the **Background: The h-Index** section to learn more about hIndex and how to compute it. Here are some important examples describing how this method behaves:

```
>>> d1 = ScotusDocket(1600, () )
>>> d1._hIndex()
0
>>> ScotusDocket(1762, (0, 0))._hIndex()
0
>>> ScotusDocket(1800, (1, 2, 3))._hIndex()
2
>>> ScotusDocket('Earl Warren', (0, 2, 15, 9, 7, 48, 4, 82, 14, 6))._hIndex()
6
```

(e) `impact(self)`. This is a public accessor/getter method which when invoked returns the calling object's h-Index. This method is also annotated with a `@property`. (*Hint.* Don't overthink it, use the private method `_hIndex`.)

```
>>> ScotusDocket(1800, (1, 2, 3)).impact
2
>>> ScotusDocket(1913, (0, 2, 15, 9, 7, 48, 4, 82, 14, 6)).impact
6
```

Task 2. Instantiating the class.

For this task, implement the function `readDecisions(filename='decisions.csv')` which takes the name of the decisions CSV file, reads in the file line-by-line and for each line, it creates a docket object (of the class `ScotusDocket`) where the attribute `label` is the year as an `int`, and the attribute `citations` is the corresponding citation sequence in the file as a tuple of `int`s. The function must accumulate these objects in a list, `docketList`, and return it.

Task 3. Plotting the impacts.

For this task, implement the procedure `plotImpacts(docketList, plotfilename)` which takes as input a list of docket objects and the name of the plot as a string. The procedure must plot the dockets (h-index vs year) in chronological order using line-style plotting to suggest trends.

To arrange the docket objects in chronological order (in ascending order by year), use sorting with lambda leveraging the `label` method. If the variable `years` is the list of years in chronological order and `h` is the list of corresponding impact scores (h indices), we can plot them as a line-plot using the command `plt.plot(years, h, 'b-')`, where the `matplotlib.pyplot` module has been imported as `plt`.

Remember to save the plot with the filename `plotfilename`. When plotted correctly, the trend looks like the plot in Figure 1.

Submitting your work. When you're finished, stage, commit, and push your work to the server as you did in previous labs. Remember that you must certify that your work is your own, by typing out the Honor Code statement in the `honorcode.txt` file, committing and pushing it along with your work.

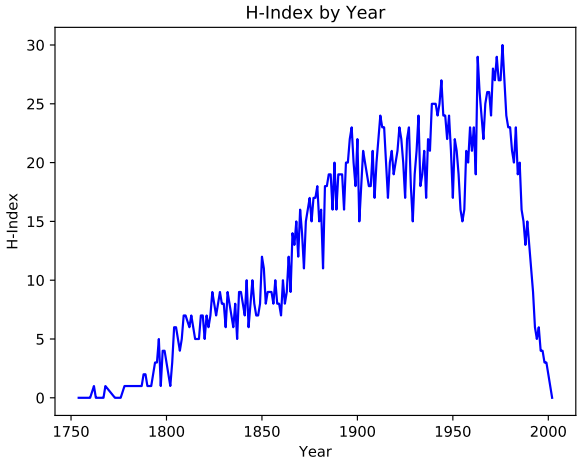

Figure 1: Plotting impact of Supreme Court dockets by year.

Late days. You are allowed a total of 3 late days over the semester, with at most 2 late days towards any one lab. You must request a late day in advance on the form, here: `http://bit.ly/s20late`.

Grading Guidelines. Both functionality and programming style are important when writing code, just as both the content and the writing style are important when writing an essay. In this program, some of the specific functional requirements we will test for include:

- Your class definition must pass all the provided doctests. You do not need to create doctests of your own, but you can if you like.
- You must stage, commit, and push the plot `lab7.pdf` generated by the following code in the `if __name__ == '__main__':` block:

```
dl = readDecisions()
plotImpacts(dl, 'lab7.pdf')
```

- Just like previous labs, we require that the functions defined in `scotus.py` follow our specifications. Do not modify the function names, their parameters, nor what is returned. The examples above in **This week's tasks** provide details about these constraints.

Stylistically, we expect to see programs that exhibit the following: meaningful names used in declarations, informative comments, good and consistent formatting, and good choice of Python commands. There is some subjectivity to what makes good style, but the basic goal is to make your ideas as clear and easy to follow as possible.

Grading scale. Programming labs will be graded on the following scale:

A+	A submission that exceeds our standard expectations for the assignment. The program must reflect additional work beyond the requirements or get the job done in a particularly elegant way.
A	A submission that satisfies all the requirements for the assignment—a job well done.
A-	Submission meets the requirements for the assignment, possibly with a few small problems.
B	A submission that has problems serious enough to fall short of the requirements for the assignment.
C	A submission that has extremely serious problems, but nonetheless shows some effort & understanding.
D	A submission that shows little effort and does not represent passing work.

Extra credit—expand your knowledge. If you'd like to push this analysis a little bit further, consider the file `chiefJustices.csv`, in which we have listed all the chief justices along with the start and end year of each justice's term.

Determining impacts of specific courts. Because these powerful US Supreme Court Chief Justices determine the cases to be heard, they often determine the "personality" of the court. Thus, for example, the *Rehnquist Court* was a conservative court that spanned the years 1986 through 2005. For the purposes of this assignment, let's consider a majority decision to be part of a particular justice's court if it was decided in a year of their term.

Write a Python procedure, `courtInfluence(docketList)` that takes as input `docketList` which is the list of docket objects returned by the `readDecisions` method. The method reads the terms of each chief justice from the file `chiefJustices.csv`, accumulations a sequence of citation counts for each justice that spans their entire term[3] and computes the impact of each of their court's decisions based on the citation sequence over their term. You may write helper functions to read in the `chiefJustices.csv` file. Note that the first line in `chiefJustices.csv` is a header row and you must handle it separately. You can either do this using code or by deleting the line manually from the file. The procedure should sort and print the names of the courts and their h-index in decreasing order of impact.

Hint. For this part of the lab, `_label` attribute of objects of the class `Scotus Docket` correspond to a Chief Justice's name rather than a particular year.

⋆

[3] In particular, if a Chief Justice's term spanned from year x to year y, their citation sequence is a concatenation of citation sequences of all years j, $x \leq j \leq y$.