04-19-2023

CS051A

INTRO TO COMPUTER SCIENCE WITH TOPICS IN AI

22: Web Pages

Alexandra Papoutsaki

she/her/hers

Lectures

Zilong Ye

he/him/his

Labs

Lecture 22: Web pages

- Web pages

Web Pages

- what is a web page or more specifically what's in a web page?
  - just a text file with a list of text, formatting information, commands, etc. Usually ends in .html
- Generally made up from three things:
  - HTML (HyperText Markup Language): this is the main backbone of the page
  - CSS (cascading style sheets): contains style and formatting information
  - JavaScript: for handling dynamic content and other non-static functionalities
- This text is then parsed by the web browser to display the content
- You can view the html source of a web page from your browser
  - In Safari: View->View Source
  - In Firefox: View->Page Source
  - In Chrome: View->Developer->View Source

html content

- html consists of tags
  (a tag starts with a '<' and ends with a '>')
- Generally, tags come in pairs, with an opening tag and a closing tag, e.g.
  <html> ... </html>
- Lots of documentation online for html
  - A good tutorial https://www.w3schools.com/html/
- We use URLs (Uniform Resource Locator) as addresses to access webpages.
- If we look at the course webpage (http://www.cs.pomona.edu/classes/cs51a/), we can see the html that generates it.
  - The default webpage for many web servers is index.html

Reading from web pages using `urllib.request`

- Look at the `url_basics.py`. What does the `print_data` function do?
    - looks very similar to other functions we've seen before for reading data
    - key difference: we're reading from a webpage!
- To read from a webpage, we need to open a connection to it (like opening a file)
    - There is a package `urllib.request` that supports various web functionality
        - The main function we'll use is urlopen
        - `from urllib.request import urlopen`
    - once you have a connection open, you can read it a line at a time, like from a file, etc.

`print_data` function in `url_basics.py`

- If we run this on the course webpage we see the following output:
- `>>> print_data(“http://www.cs.pomona.edu/classes/cs51a/")`
  `b’…’`
- Which mirrors roughly the same text we saw through our browser but starts with b.
  - These aren't actually strings. We can check the type by adding an extra print statement
    - `print(type(line))`
  - If we run again with the type information printed out we see:
    - `<class ‘bytes'>`
    - `bytes` is another class that represents raw data
  - Webpages can contain a wide range of characters (e.g., Chinese characters)
  - We need to know how to interpret the raw data to turn it into characters.

`print_url_data` function in `url_basics.py`

- `timeout` is an optional parameter that specifies a timeout in seconds for blocking operations like the connection attempt. It will be useful in the next assignment.
- Often web pages will have as metadata the character encoding to use.
- For our purposes, we'll just make a best *guess* at a common encoding scheme, ISO-8859-1, which handles a fair amount of web pages.
- The `bytes` class has a 'decode' method that will turn the bytes into a string
- If we run `print_url_data`, we'll see that we get the same output, but now as strings:

-
  ```
  >>> print_url_data(“http://www.cs.pomona.edu/classes/cs51a/")
  '...'
  ```

`get_lectures_url` function in `url_extractor.py`

- What does the `get_lectures_urls` function do?
  - opens up the course web page
  - reads a line at a time
  - checks each line to see if it contains a link to lecture slides and if so, keeps track of it in a list

`get_lectures_url` function in `url_extractor.py`

- `str.find(some_string)`:
  - returns the index in `str` where `some_string` occurs, or -1 if it doesn't.
  - starts searching from the beginning of the string
- `str.find(some_string, start_index)`
  - rather than starting at the beginning, start searching at `start_index`.

```
>>> "banana".find("ana")
1
>>> "banana".find("ana",2)
3
```

`get_lectures_urls` function in `url_extractor.py`

- what does `begin_index = line.find(search_line)` do?
  - finds where the lecture strings starts.
- what does `end_index = line.find('"', begin_index)` do?
  - searching for the end of the link.
    - The html syntax for linking to a page is `<a href = “yourlink.com”>link</a>`

`write_list_to_file` function in `url_extractor.py`

- Opens a file, this time with “`w`" mode as a second parameter instead of “`r`".
  - "`w`" stands for write
  - if the file doesn't exist it will create it
  - if the file does exists, it will erase the current contents and overwrite it (be careful!)
- We can also write to a file without overwriting the contents, but instead appending to the end
  - We would use the “`a`" mode which stands for append
- Just like with reading from a file, we get a file object from open
- The "write" method writes an object to the file as a string
- Write does NOT put a line return after the end of it. You will need “\n”!

`write_lectures` function in `url_extractor.py`

- Gets the lecture urls from the course web page
    - `COURSE_PAGE` is written in all caps to indicate a constant, a variable whose value should not be changed by the user.
- Writes them to the outfile.

Revisiting `url_extractor.py`

- Look at the webpage http://cs.pomona.edu/classes/cs51a/
- Now look at the output: do we get **all** of the lecture slides links?
- No! We miss the ones with the notes. Why?
  - The code assumes one lecture per line, but that's not true
- How do we fix this?
  - rather than searching per line, treat the entire webpage as a long string
  - search for the first occurrence of lecture,
  - extract it,
  - then search again starter at the end of that occurrence.

`get_lectures_urls_improved` function in `url_extractor_improved.py`

- Look at the get_lectures_urls_improved function
  - `read()` method reads and returns the entire contents all at once rather than reading a line at a time.
    - This also works on files!
- We then decode this so that `page_text` has all of the webpage text as a string.
- What does `begin_index = page_text.find(search_line)` do?
  - searches for the index of the first occurrence of `lectures/`
- The code will enter the while loop if it finds an occurrence.
- What does `end_index = page_text.find('"', begin_index)` do?
  - searches for the end of the link. We can then extract the url
- What does `begin_index = page_text.find(search_line, end_index)` do?
  - searches again, but now starting at `end_index`, the end of the last link found
- If we run the improved version, we now get the notes links, too.

`get_note_files_only` function in `url_extractor_improved.py`

- Function that allows us to just extract the name of the file (e.g, Lecture1.pdf).
- key change: we want to skip the "lectures/" part when extracting the page.
  - rather than using `begin_index`, we want to skip the length of "lectures/" forward when extracting.

Difference between http and https

- The 's' stands for secure. When you communicate with an https website:
  - you get some reassurance that you're actually communicating with the website (rather than someone pretending to be the website).
  - your communications are encrypted so it's difficult to see what information you're sending back and forth.
  - there is a bit of overhead in setting up this communication properly
  - the right way is to install SSL certificates for python.
  - for simplicity, however, you can also tell python to simply ignore the SSL certificates and connect to an https site without checking.
  - Look at `url_basics_ssl.py` code
    - urlopen has an optional parameter that you can specify that will allow you to connect to an https webpage without checking ssl certificates.

Resources

- url_basics.py
- url_extractor.py
- url_extractor_improved.py
- url_basics_ssl.py

Homework

- Assignment 11 (cont'd)