Plan for LWoC

● Power of Regular Expressions
  ➢ From theoretical computer science to scraping webpages
  ➢ Using documentation, understanding language

● Surveys and providing Feedback
● Review Recommender Assignment
  ➢ Pending Questions
● APTs and APT-Quiz
  ➢ Labs and APTs

APTs

● Final APT quiz starts tonight
  ➢ See Sakai for grades on previous APT quizzes
  ➢ 100 points max on APT quizzes

● Final APTs part of lab or challenge
  ➢ Both are challenges, can be used as APT points
  ➢ Completed by Friday

● Will update grades in Sakai ASAP

Course Evaluations: 10 minutes

● Please go to ACES and complete evaluation for course
  ➢ Very important!

● Use Sakai for UTA evaluation if there's time

What is Computer Science?

● ... "it is the study of automating algorithmic processes that scale."

● If you need to find one email address on a webpage, you don't need computer science
  ➢ If you need to scrape every email address, that number in the 10's to 100's, you could use help
Contributions from The Web

- Randall Munroe
  - https://xkcd.com/208/
  - https://xkcd.com/thing-explainer/

- Regex "joke"
  - Some people, when confronted with a problem, think “I know, I’ll use regular expressions.” Now they have two problems.

- Regular expressions can be tough to write and debug, but are often very useful

How do you solve a problem like …

- How many words end in "aria"?
  - Start with "aria"? Contain "aria"?
  - Why would you care about this?

- Can you find ola@cs.duke.edu, susan.rodger@duke.edu, and andrew.douglas.hilton@gmail.com when searching through a webpage source?
  - What is the format of a "real" email address?

Examples of regex’s at work

- What do aria$ and ^aria and aria share?
  - Answers to previous question

- What about the regex .+@.+?
  - Turns out that . has special meaning in regex, so does +, so do many characters

- We’ll use a module RegexDemo.py to check
  - Uses the re Python library
  - Details won’t be tested, regex knowledge will

Regex expressions

- Regex parts combined in powerful ways
  - Each part of a regex "matches" text, can extract matches using programs and regex library
  - ^ is start of word/line, $ is end

- Expressions that match single characters:

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, a, 9 or ...</td>
<td>Any character matches itself</td>
</tr>
<tr>
<td>.</td>
<td>Matches any character</td>
</tr>
<tr>
<td>\w</td>
<td>Matches alphanumeric and _</td>
</tr>
<tr>
<td>\d</td>
<td>Matches digit</td>
</tr>
<tr>
<td>\s</td>
<td>Matches whitespace</td>
</tr>
</tbody>
</table>
Regex expressions

- Repeat and combine regex parts
  - * means 0 or more occurrences/repeats
  - + means 1 or more occurrences/repeats
  - ? Means (after * or +) to be non-greedy
- Expressions match more than one character
  - [a-zA-Z] Brackets create character class
  - (regex) Tag or group a regex
  - \1 or \2 Matches previously grouped regex
  - (1) or (n) Repeat regex 1 or n times

Regex examples tried and explained

- Five letter words ending in p? Starts 'd'?
  - ^\w\w\w\wp$ but not ...p$
- Seven letter words, or seven ending with 'z'
  - Difference between ^\w{7}$ and ^\w{7}$
- Words that start with a consonant:
  - ^[^aeiou]$ double meaning of ^

Summary of Regular Expressions

<table>
<thead>
<tr>
<th>regex</th>
<th>purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>any character</td>
</tr>
<tr>
<td>*</td>
<td>zero or more of preceding regex</td>
</tr>
<tr>
<td>+</td>
<td>one or more of preceding regex</td>
</tr>
<tr>
<td>?</td>
<td>non-greedy version of either * or +</td>
</tr>
<tr>
<td>(regex)</td>
<td>tag or group a regex</td>
</tr>
<tr>
<td>\1, \2</td>
<td>match numbered tag or group</td>
</tr>
<tr>
<td>(\1)</td>
<td>repeat regex \1</td>
</tr>
<tr>
<td>(\1)</td>
<td>repeat regex \1</td>
</tr>
<tr>
<td>^[a-zA-Z]</td>
<td>character class</td>
</tr>
<tr>
<td>$</td>
<td>end of line/string</td>
</tr>
<tr>
<td>^</td>
<td>beginning of line/string</td>
</tr>
</tbody>
</table>
Answer Questions


NCWIT survey

- See course website for URL for survey

Scraping email address from websites

- Suppose we want to send email to all Duke Faculty to let them know ...
  - Visit Departmental website, people, faculty
  - View (HTML) Source
  - Develop regex to access email – if possible!
- RegexScraper.py
  - Python makes this simple
  - Ethical hacking?

Scraping math.duke.edu faculty

- Pattern:
  - r'math/faculty/(.*?)"\>(.+?)\'<
- URL
  - http://fds.duke.edu/db/aas/math/faculty/
- Matches:
  ...
  - ('motta', 'Francis C. Motta')
  - ('jmmza', 'James Murphy')
  - ('ryser', 'Marc D. Ryser')
  - ('sv113', 'Stefano Vigogna')
  - ('haizhao', 'Haizhao Yang')
Scraping Sanford/PubPol faculty

- **Pattern:**
  - `r'(\w*\.\w+)*@((\w*\.\w+\w+)\')`
- **URL**
  - `https://sanford.duke.edu/people/`
- **Matches (call 16 times with different URL)**

  ```
  ('schanzer', 'duke.edu')
  ('steveschewel', 'gmail.com')
  ('michael.schoenfeld', 'duke.edu')
  ('schroeder', 'law.duke.edu')
  ```

Scraping Biology faculty

- **Pattern:**
  - `r'mailto:\(\w*\.\w+\w+)@((\w*\.\w+\w+)\')`
- **URL**
  - `https://biology.duke.edu/people/all-faculty/`
- **Matches (call 26 times with different URL)**

  ```
  ('emily.bernhardt', 'duke.edu')
  ('emily.bernhardt', 'duke.edu')
  ('bhandawat', 'gmail.com')
  ('bhandawat', 'gmail.com')
  ('jboynton66', 'gmail.com')
  ('jboynton66', 'gmail.com')
  ```