SI206 Discussion 10

Midterm 2 Review and XML

Midterm 2 Review

Unit Tests

- Import unittest
- Make a subclass of unittest.TestCase
- Define methods for test cases
  - Start each method with test_
  - Use assertion methods to check the results. Ex:
    - assertEqual (first,second): Test that *first* and *second* are equal.
    - assertAlmostEqual(first,second,places): Test that *first* and *second* are approximately (or not approximately) equal by computing the difference, rounding to the given number of decimal *places* (default 7), and comparing to zero
  - Run the tests using unittest.main
- Test cases
  - Test usual values (expected values, length, type, etc.)
  - Test edge cases (negative, empty, etc)

Reading Files

- TXT files
  - file_obj = open(filepath, ‘r’)
  - file_obj.read() : reads entire file as string
  - file_obj.readlines() : reads entire file as list of strings
  - file_obj.close()
- CSV files
  - reader = csv.reader(f)
  - Iterate through reader to read lines of csv as lists
  - writer = csv.writer(f, delimiter=’,’)
  - writer.writerow(list) : write list to row of csv file
- with statement
  - Closes file automatically

Regex

- re.findall('re_string', string) : returns a list of strings that match the regex
- When using the \b character, make sure your string is a raw string
- Special characters:
  - \w - Alphanumeric characters and underscore
  - [...] - set of characters
  - [^...] - Any character not in the brackets
  - \s - Any whitespace character
  - . - Any character
  - * - Repeat 0 or more times
  - + - Repeat 1 or more times
  - \b - Boundary between alphanumeric characters and whitespace
  - ^ - start of a string
  - $ - End of a string
  - (...) - Returns only the expression inside the parenthesis
  - (?:...) - Treats the characters in the parens as a whole expression. The ?: negates the effect of grouping and returns the full matched string.
  - See regex cheat sheet for more special characters

BeautifulSoup

- 3 steps
  - Store the url of website in a variable
  - Get the data from the url
    - r = requests.get(url)
  - Create a BeautifulSoup object using the data
    - soup = BeautifulSoup(r.content, ‘html.parser’)
- Soup object methods
  - soup.find(‘<tag>’, attribute=’value’) : returns the first tag that matches
  - soup.find_all(‘<tag>’, attribute=’value’) : returns a list of all tags that match
  - tag.attrs : returns a dictionary of the attributes in the tag object
  - tag.get(‘attribute’) : returns the value of a specified attribute or None if the attribute does not exist

Discussion 9 Exercise

Your goal:

Read XML from a live website, parse it in Python, and use it to programmatically answer these questions:

1. What's the price of a California poppy plant?
2. What plants from this list can I grow in zone 5?

(here's more info on USDA zones, if you happen to be curious

Steps to take:

1. Import the packages you'll need
2. Make a request to the site using requests.get()
3. parse your XML
4. Answer the questions & pass the tests

Extra Midterm Review Exercise

Tasks

- Task 1
  - Implement the get_profs() function. This function should read in `umsi_faculty.csv` and parse it to return a list of lists. Each list should contain the name, title(s), and email address of each professor in the csv file.

- Task 2
  - Implement the get_valid_emails() function. This function should accept the list from Task 1 and return a dictionary. The keys should be the names of professors and the values should be their email addresses. Some of the email addresses were entered erroneously. Use a regular expression to filter out invalid email addresses.
  - A valid email address, for this task, should:
    - Only have lowercase letters (no numbers or uppercase letters)
    - End with @umich.edu