ENGL 516X:
Methods of Formal Linguistic Analysis

Semester: Spring '18

Instructor: Sowmya Vajjala

Iowa State University, USA

1 March 2018

Outline

- Announcements etc.
- Solution to yesterday's question
- Writing small web-applications in Python: An Introduction

Announcements etc

- Questions about assignment 4: I updated the description on Canvas to clarify things.

Announcements etc

- Questions about assignment 4: I updated the description on Canvas to clarify things.
- Order of assignments: A5–A7 changed (deadlines remain same. A5 became A6. A6 became A7 and A7 became A5).
- Reason: I forgot about LARC being right after spring break - that may not give enough time for original A5.

What is a web application?

First things first: What according to you is a web-application?
What should it do?

What is a web application?

First things first: What according to you is a web-application?
What should it do?
What should we know to develop programs that work on the browser, over the web, in python?

Terminology

1. Server - a computer device or program that provides some functionality to several programs called clients. In web programming, clients are typically browsers on your users' machines.
2. Client - devices or software that access the services of a server.
3. Server side programming: programs that you write to make your server do its job.
4. Client side programming: programs you write on the clients side, so that the client-server communication happens, and so that the client does not go to server for things that can be done there itself.
5. In this class, we will talk about server side web programming.

Why is this here?

1. Since this is a first computer programming course, I care about breadth, not depth (in 520, it is depth)
2. Developing a web-application is not something you can totally rule out as impossible in future.
3. So, understanding basic workings will only help you (atleast to communicate with whoever is doing the web-application for you.

Why is this here?

1. Since this is a first computer programming course, I care about breadth, not depth (in 520, it is depth)
2. Developing a web-application is not something you can totally rule out as impossible in future.
3. So, understanding basic workings will only help you (atleast to communicate with whoever is doing the web-application for you.
4. It is one of those "You'll thank me later" things, as far as I am concerned.

Web application development in Python

- There are some collections of packages and modules in Python, that allow us to write web application code. These are called "Web Frameworks".
- They take care of low level details and allow us to focus only on our application.
- Several web frameworks exist in python, and their primary differences lie in the diversity of functionalities they provide.
- For a detailed overview: `https://wiki.python.org/moin/WebFrameworks/`
- For a general idea about using python in the web: `https://docs.python.org/2/howto/webservers.html`

Bottle: Introduction

- Bottle is a light weight, easy to setup web framework for Python. It is distributed as a single module, and does not have any external dependencies.
- Provides support for basic data access, file uploads, display of web pages and so on.
- installation:
  1. Download bottle.py from `http://bottlepy.org/bottle.py` and save it into your project folder. Then, just import bottle in your program, and you are ready to go.
  2. "install" bottle in PyCharm (`https://goo.gl/uTTPwp`)

A Hello World Web Application

```
from bottle import route, run

#route is to give a path
@route("/")
def whatever():
    return 'Hello world!'

#route is to give a path
@route("/hello")
def anotherfunction():
    return 'Hello world in hello path!'

#run is to run your program on the server
#localhost - your computer.
#When you deploy the application somewhere,
#you will put that IP instead of localhost
run(host='localhost', port=8080)
```

Terminology

1. @route, @get, @post - these are "descriptions" before a function definition, used to tell us what the purpose of the function is.
2. @route: route takes you to a path on the website and executes that function below it.
3. @post: this is used typically in cases where we "submit" some data to the server, and the server receives whatever we sent (does some processing if needed)
4. @get: this is used to get some information from the server to the client browser.

HTML

- HTML - Hyper Text Markup Language. It is a markup used to display formatted data on webpages. We can think of it as a protocol between the browser and the webpage author on how the content should be displayed.
- HTML - is full of various tags. So, key to writing good HTML is to know what these tags are and what functionalities they have.
- You can do a lot of things with HTML, but we don't have to know all that for your final projects.

HTML Tutorial

- HTML is a collection of tags. Most the tags also have an end tag. For example, a tag XX starts with $< XX >$ and ends with $< /XX >$.
- Two top level tags are : head and body. Head contains information about the title of the page and other meta data. Body contains the actual body of your html page.
- Inside body, you can have "forms" where the user can enter some input (choose from a list of options, enter some text, enter passwords etc.) and submit it to your server to do some additional processing.

A basic HTML page

```
<html>
<head><title>A HTML Page </title></head>
<body> This is an example html page </body>
</html>
```

Some text formatting tags: b is for bold, i is for italics, $< br >$ is for a new line (without an end tag), $< p >$ and $< /p >$ cover a paragraph..and so on.

@get and @post

```
from bottle import get, post, request, route,run

def check_login(u,p):
   if u == p == "dummy":
         return True
   else:
         return False

@route('/')
@get('/login') # or @route('/login')
def login():
    return '''
        <form action="/login" method="post">
            Username: <input name="username" type="text" />
            Password: <input name="password" type="password" />
            <input value="Login" type="submit" />
        </form>
    '''

@post('/login') # or @route('/login', method='POST')
def do_login():
    username = request.forms.get('username')
    password = request.forms.get('password')
    if check_login(username, password):
        return "<p>Your login information was correct.</p>"
    else:
        return "<p>Login failed.</p>"

run()
```

Having text areas and submit buttons

```
 <form>
  First name: <input type="text" name="first"> <br>
  Last name:  <input type="text" name="last"><br>
  Tell me something about yourself:
<textarea name="message" rows="10" cols="30"> </textarea>
<br><input type="submit" value="Submit">
</form>
```

That is not enough in real-life

1. What should happen after submit?

That is not enough in real-life

1. What should happen after submit?
2. Our data should go somewhere, something has to happen. What is the point otherwise?

That is not enough in real-life

1. What should happen after submit?
2. Our data should go somewhere, something has to happen. What is the point otherwise?
3. So, we use "action" and "method" attributes in form. "action" tells us which function in the code gets activated after submitting the form. "method" tells us if it is a get or post request.

Example HTML with action and method

```
 <form action="/tokenize" method="post" id="formid">
 <textarea form ="formid" name="taname" cols="35">
</textarea>
  <input value="Tokenize" type="submit" />
        </form>
```

Some Example Web applications using Bottle

(uploaded on canvas - go through the code)

1. Tokenizer application
2. Login page application
3. Uploading a file

For more details on using Bottle

Look at the tutorial and other additional resources in:
`http://bottlepy.org/docs/dev/index.html`

To learn more about HTML

`http://www.w3schools.com/html/default.asp`

Today's Exercise

1. Do some lessons in the HTML tutorial from W3 schools
2. After that, write a Python program using Bottle, that shows two text fields and a button in the initial screen:
   First Name:
   Last Name:
3. In the program, have a function, that gets called when the user submits this information, which takes these two values, and returns - "Hello" + FirstName + LastName on the screen.
4. You can post your solutions in Today's forum if you finish this.

Extra Exercise

1. Write a Python program using Bottle, that shows two text fields and a button in the initial screen to take two strings as input
2. In the program, have a function, that gets called when the user submits this information, which takes these two values, and does the following:
   - 2.1 checks if the two words are permutations of each other (i.e., god-dog; spam-maps etc.)
   - 2.2 Shows a message: "The words are permutations" or "The words are not permutations" depending on the result of the check.
   - 2.3 Note: Inputs don't have to be valid words. Any word strings are okay, and you can ignore punctuation, spaces, numbers etc.
3. You can post your solutions in Today's forum if you finish this.

Next Week

- Tuesday:
  - Continue on this topic (but with more hands on work)
- Thursday:
  - General review
  - Discussion about final projects (some ideas uploaded on canvas)
  - Assignment 5 description
- To do for you: See the final project descriptions, discuss among your team mates
- It is okay to come up with a new idea. But remember: you don't have all time in the world.