# **ELE409 SPRING2018 LAB5**

# **Python webpage information fetching and plotting**

### **Objectives:**

* Use python to fetch and parse through a web page for information
* All of lab 5 will be performed on the desktop linux machine except for exercise 3 because it requires the use of a package that needs to be installed. Also this will be to show that since python is a scripting language that needs to run in a python environment it is easily portable.

### **Pre-Lab:**

Make sure you have downloaded the lab 5 zip file from the sakai website.

### **Exercise 1: Sending Data to a Webpage through Query Variable**

1. In the last lab (hardware control through a webpage) you may have noticed that the variable data that we passed from the webpage so that the program knew what to do with the hardware was the Query environment variable. It looked something like this (hexValue=123456&LED2=on&LED5=on) . We are going to do something similar in order to send data to a webpage.
2. We are going to get the time at a particular place from a website called timeanddate.com
3. First you should see how the website finds the time when used normally. Go to timeanddate.com and find the time from a different time zone. Use somewhere in California for example.
4. Take notice of what the URL is after you have found a place.
5. Now you should view the html so you can see how the webpage displays that the data. For most browsers you can simply press the F12 key. Make sure you are in the inspector tab, and search for the time in the search bar. The result should look something like:
   * <span class =”h1” id=”ct”>HH:MM:SS am</span>
6. We are going to write some python code to extract that.
7. Open up a new file in gedit or some other text editor. Add the following lines
   * import urllib
   * city = “ChosenCity” #ie. sacramento
   * page = “https://www.timeanddate.com/wolrldclock/usa/” + city
   * content = urllib.urlopen(page)
   * for line in content:
   * print line
8. Now save as a .py file
9. open up a terminal window and type:
   * >> python filename.py >! output.html
10. This line will run the python script and write the output to a file called output.html. There is an exclamation point there just in case the output.html file already exists. It is not otherwise necessary.
11. Now find the output.html file and open it with a browser to see if you got the output you were expecting. (It should show an html page that is all text and some of the formatting will be wrong because that is handled by something more complicated. However somewhere within that text should be the time of the city you specified)

### **Exercise 2: Parsing through html text**

1. Now we are going to parse through the html text to find the time that we wanted in the first place.
2. Go back to the previous program and add this import line:
   * import re #for regular expressions
3. Above the “for loop” we need to compile a pattern to search for. Add this line:
   * pattern = re.compile(r”\d{1,2}:\d{1,2}:(\d{1,2}) (am|pm)”, 0)
4. the r stands for raw string so that python does not try to interpret the “\” as an escape character. The “\d” with match any digit. The “{1,2}” will match from 1 to 2 of the preceding pattern so in our case that means 1 or 2 digits. The “:” are literal colons because we are looking for time. The “ (am|pm)” will match either the literal string “am” or “pm”. The “()” around the third “\d{1,2}” make it so that this particular piece of matched string will be put into a numbered group that we will use in the next exercise.
5. Now at the bottom of the program comment out the print line and type:
   * match = re.search(pattern, line)
   * if match is not None:
   * print match.group(0)
6. By default the the entire matched string is in group(0).
7. Run this python script and see if it prints out the correct time for your chosen location

### **Exercise 3: Plotting**

1. It won’t be very interesting but to learn how to plot things with python we are going to plot the seconds from the time.
2. First you will have to install matplotlib for python on the boards. Type:
   * >> sudo apt-get install python-matplotlib
3. Next you will have to add these import lines:
   * from time import sleep
   * import matplotlib.pyplot as plt
4. Next put your previous code inside a loop that will run through as many times as you want points in your graph. I would say something like 100 so that you can see something interesting happen when the seconds reset (go from 59 to 0).
   * for i in range(100):
5. You will also want to sleep your loops so that it doesn’t run too fast. (Actually this python program will run much slower than the c code you are used to, so we don’t have to sleep for very long. Also using the sleep function is usually not the correct way to control the speed of code execution, because it needlessly occupies the processor, but we are going to do it anyways. Lastly we are plotting seconds against the loop cycles which is weird but it is easier for now than keeping track of the system time and plotting system time against the time would be a very very boring plot). Add the following at the end of the highest tier loop:
   * sleep(.1)
6. Next we have to initialize an array to hold the seconds that we want to plot. Technically python is dynamically typed so that isn’t necessary but it is good practice for code readability. Above the highest tier “for loop” add this line:
   * secondsArray = []
7. Remember how re.search puts certain matches into a numbered group. We will use this to add the seconds to an array. Underneath the print statement add this line:
   * secondsArray.append(match.group(1))
8. Now we simply have to plot.
9. Add the following at the beginning of the program after the import statements:
   * plt.ion()
     1. This turns on interactive plotting so that a plot doesn’t block execution of the program.
   * fig = plt.figure() # handle to figure
   * ax = fig.add\_subplot(111) # handle to axis
10. Add the following lines after the line that appends a value to secondsArray:
    * secPlot = ax.plot(secondsArray, ‘b-’)
    * fig.canvas.draw()
11. Now save the program and send it to the board. You will need to set the display so that linux knows where to display the plot. From the board run this command:
    * >> export DISPLAY=:0
12. Now run the script using the same command as before. Notice that there was no special compilation (cross-compilation or local compilation) because python runs in an environment.

### **Exercise 4: GUI**

1. Next thing would be to put the whole thing into a GUI so that it looks nice. However doing so would mean a restructuring of how the simple code we have written would work so I have given you a file called L1E4.py
2. Run L1E4.py and try to understand how it works (It’s not at all complicated it just would have taken too long to walk through it step by step in this lab). You will need to use a GUI for the assignment and upcoming project.

### **Assignment: Tracking stock market prices**

For the assignment implement a stock price tracker. You can use any website you want to get the information so long as it frequently updates (ie. every second or so). You can try using the website money.cnn.com/data/markets if you can’t find anything better. Have it show the stock price of some popular stocks (have 3 or 4 selectable from a drop down menu) over the time that the program has been plotting. Depending on how long the exercises take try to have it plot the 3 stocks based on checkboxes so that 1, 2 or all 3 stocks could be on the plot at the same time (look up python tkinter Checkbuttons). Go to the website first and press F12 to see the html code, and find the stock prices that are shown and how they are shown. Then write the regular expression to capture the information. To help with regular expressions try to use regex101.com, they allow you to pretty conveniently test your regular expression on some test-text.