CS 3430: Python & Perl
Assignment 5: Temperature Log Analysis of an Electronic Beehive Monitoring System

Vladimir Kulyukin
Department of Computer Science
Utah State University

1. Learning Objectives

1. Py & PL Strings
2. Py & PL String Splits
3. Py List Comprehension
4. Py & PL Sorting

2. Solar-Powered Electronic Beehive Monitoring

The Apis millifera, also known as the common honeybee, is responsible for one out of every three daily mouthfuls that the average U.S. resident eats. Since 2006 honeybees have been disappearing from amateur and commercial bee yards. This trend has been called the colony collapse disorder (CCD). While the beekeepers can rebuild colonies over time, the high rates of colony loss continue to put significant pressure on agriculture and threaten food supply chains. The July 2015 issue of the American Bee Journal reports that beekeepers across the U.S. lost more than 40 percent of their bee colonies from April 2014 to April 2015. What is unusual about 2015 is that the summer losses were greater than the winter losses. The CCD is not the only malady affecting the honeybee. Other growing threats include Varroa mites, American and European foulbrood, and nausema.

There appears to be an emerging consensus among some researchers and practitioners in entomology, apiculture, computer science, and engineering that a possible solution that will help both the beekeeper and the honeybee is to transform traditional bee yards into data processing clusters. These clusters will monitor their environments through multiple sensors, detect and recognize behavior patterns, and notify beekeepers and other interested third parties (e.g., climate change and environmental science researchers) about significant deviations and anomalies. A major objective of my current research project is to transform bee yards into distributed solar-powered multi-sensor data clusters that collect and analyze large volumes of live data in real time. This type of monitoring system will, in my opinion, enable researchers and practitioners to collect objective data on different bee races, different hive designs, and different climates.

In this assignment, you will get a glimpse of what it is like to process sensor data collected by an electronic beehive monitor. Figure 1 shows three beehives with multiple sensor monitors powered by solar panels placed on top of each hive.

Figure 1. Three Beehives Equipped with Solar-Powered Bee Hive Monitors at the USU Organic Farm

One of the monitor's sensors is a temperature sensor. Readings are taken every 10 minutes so long as there is sufficient power. When there is no charge left in the battery the system hibernates until there is enough power to continue its normal operation. The readings of the temperature sensor are saved in a txt file on a Raspberry PI computer in the top box of the hive.

```
2015-04-08_16-53-14 22.25
2015-04-11_16-56-58 22.312
2015-04-11_17-06-58 22.312
2015-04-11_17-16-58 23.125
2015-04-11_17-26-58 23.0
```

Figure 2. Excerpt from a Celsius temperature log

The format of each line is as follows: **YEAR-MONTH-DAY_HOUR-MINS-SECS<SPACE>FLOAT**. For example, in the first line **2015-04-08** refers to **4/8/2015**. Then, after the underscore character _, we have **16:53:14**, which, in the 24-hour notation, means 16 hours, 53 minutes, and 14 seconds, or **4:53:14** p.m. Then, there is a space followed by **22.25**, which is the Celsius temperature reading at that moment in time on the temperature sensor.

3. Analyzing Temperature Logs

I have provided two files - **celsius_temp_log.py** and **celsius_temp_log.pl** – that you can use as templates for this assignment. The files have two string constants defined: **short_temp_log** and **long_temp_log**. Yes, I know, I know, we should have used real files. We will learn how to do it after we cover I/O. For now, we will deal with strings.

Besides, the string processing logic of your programs will not change if the input is taken from a file. So, to complete this assignment, you should use **short_temp_log** to implement and debug the functions described below and **long_temp_log** for the final tests.

Step 1: Implement the Py function and PL subroutine **process_temp_log_record** that take a string that represents one temperature record and returns a 7-tuple whose 0th element is the year, the 1st element – the month, the 2nd element – the day, the 3rd element – the hour, the 4th - the minutes, the 5th - the seconds, and the 6th – the Celsius temperature reading. Here is the Py shell test. The PL output should be the same.

>>> process_temp_log_record('2015-04-08_16-53-14 22.25')
(2015, 4, 8, 16, 53, 14, 22.25)

Step 2: Implement the Py function and PL subroutine **process_temp_log** that takes the entire temp log represented as a string, i.e., **short_temp_log** or **long_temp_log**, and returns a list of 7-tuples, each of which is computed by **process_temp_log_record()**. Here is the Py shell test. The PL output should be a list of 7-tuple references.

>>> process_temp_log(short_temp_log)
[(2015, 4, 8, 16, 53, 14, 22.25), (2015, 4, 11, 16, 56, 58, 22.312), (2015, 4, 11, 17, 6, 58, 22.312), (2015, 4, 11, 17, 16, 58, 23.125), (2015, 4, 11, 17, 26, 58, 23.0)]

Step 3: Implement the Py functions and similar PL subroutines **display_max_celsius_temp(temp_log)**, **display_min_celsius_temp(temp_log)**, **display_average_celsius_temp(temp_log)** that find and print the messages about the maximum, minimum, and average temperatures, respectively, in a specific temperature log. Here are the Py shell tests. The PL equivalents must produce the same outputs in the command line window.

>>> display_max_celsius_temp(short_temp_log)
Maximum Celsius temperature of 23.125 was at 17:16:58 on 4/11/2015
>>> display_max_celsius_temp(long_temp_log)
Maximum Celsius temperature of 23.312 was at 18:16:58 on 4/11/2015
>>> display_min_celsius_temp(short_temp_log)
Minimum Celsius temperature of 22.25 was at 16:53:14 on 4/8/2015
>>> display_max_celsius_temp(long_temp_log)
Maximum Celsius temperature of 23.312 was at 18:16:58 on 4/11/2015
>>> display_average_celsius_temp(short_temp_log)
Average Celsius temperature from 16:53:14 on 4/8/2015 to 17:26:58 on 4/11/2015 is 22.5998
>>> display_average_celsius_temp(long_temp_log)
Average Celsius temperature from 16:53:14 on 4/8/2015 to 21:56:58 on 4/11/2015 is 22.03096875

4. What to Submit

Save your Py solution in **celsius_temp_log.py** and your PL solution in **celsius_temp_log.pl** and submit both files in Canvas.

Happy Hacking!