New York University
School of Continuing and Professional Studies
Division of Programs in Information Technology

Advanced Python

Homework, Session 1

In this class employing good coding practices and style are as important as learning new language features. Accordingly, please begin by reviewing the **Advanced Python Code Quality** handout, and then the exercises, homework assignments and solutions for **Sessions 1-4** on the Introduction to Python course website. You may wish to warm up by completing some of these assignments and/or the exercises given for each of these sessions. The solutions can be found under homework solutions (to come) on our own class website.

Please note that since we are focusing on the exercise and review of specific features in this first session, you must not use more advanced features in your solution (such dictionaries), or external modules (the **math** module, **pandas**, etc.), except as may be specified in the assignment.

1.1 Write a program that dynamically downloads weather data for a particular city and year from the **Weather Underground** website (see below), then computes average temp, range (min and max) temps and standard deviation (optional, explained below).

```
$ python homework_1.1.py KNYC 2006
```

```
Daily temperature data for KNYC in year 2006:
Average temp: 56.9616438356
Max temp:  90
Min temp:  23
Standard deviation:  15.1257254339
```

This solution requires that you retrieve CSV data from the web, loop through the CSV data, split out the needed value from each line, convert the value to the proper type, and sum up the values. It also requires that you keep a running count of values encountered.

Data source: the Weather Underground is a website devoted to reporting weather data. It identifies reporting sources by a string code representing an airport reporting station. Some sample IDs:

New York	KNYC
Chicago	KMDW
San Francisco	KSFO
Los Angeles	KCQT
Honolulu	PHNL

The data can be obtained from the web using the following URL pattern (this one returns daily meteorological data for Los Angeles from 1/1/2013 - 12/31/2013 -- note the bold values, which will be dynamically added to the URL for each request):

```
https://www.wunderground.com/history/airport/KCQT/2016/1/1/Custo
mHistory.html?dayend=31&monthend=12&yearend=2016&format=1
```

You can view this data visually (by pasting this URL into a browser) as well as obtain it programmatically (by using the **urllib2** library), as below:

```
import urllib2

dh = urllib2.urlopen(url)  # url is string like example above

for line in dh:   # line is a string, assigned each line in CSV text
    print line
```

This script loops through each line in the web response (in this case, the CSV file) and assigns the line, as a string, to the variable **line**. Note that this uses identical syntax to opening and looping through a file.

Looking at the returned dataset, note the column headings in the first printed line. The temperatures we will use for this analysis are the **Mean TemperatureF** values.

As mentioned above, **please do not use any import libraries except as specified**. This restriction is to ensure that you are making use of the features highlighted in the session, instead of going at it in some other way with which you are more familiar (for example by using the **pandas** or **math** libraries).

Hints:

1. To take arguments at the command line, use the **sys** module and **sys.argv** list. See slides.
2. Use the string **format()** method to build the url -- i.e., to insert the dynamic values (date, station symbol) into a string template of the url.
3. **Mean (average)** temperature is calculated by dividing the sum of values by the count of values (use **sum()** and **len()** from the list).
4. Please note that **max** and **min** are the maximum and minimum temperature from the **MeanTemperatureF** column, i.e. not the absolute max and min for the year.
5. The **standard deviation** indicates average deviation of all temperatures from the average of temperatures. It is calculated by comparing each value to the mean.

   Standard Deviation is calculated in the following manner:

   a. For each value in the dataset, find the difference from the mean and square it.
   b. Calculate the average (sum / count) of these squared values (the *variance*).
   c. Find the square root of the variance (**sd = v ** (0.5)**).

Example runs:

```
$ python homework_1.1.py KNYC 2006

Daily average temperature data for KNYC in year 2006:
Average temp:  56.9616438356
Max temp:  90
Min temp:  23
Standard deviation:  15.1257254339


$ python homework_1.1.py KMDW 2006

Daily average temperature data for KMDW in year 2006:
Average temp:  53.2849315068
Max temp:  89
Min temp:  2
Standard deviation:  17.3818017827


$ python homework_1.1.py PHNL 2010

Daily average temperature data for PHNL in year 2010:
Average temp:  77.7643835616
Max temp:  83
Min temp:  69
Standard deviation:  3.0330313903
```

1.2 Write a program that takes an integer as user input and uses a **while** loop to generate a *fibonacci series* (in which each number in the series is the sum of the previous two numbers in the series (the series begins 1, 1, 2, 3, 5, 8, 13, etc.)) up to the user's number.

Example run:

```
$ python homework_1.2.py 100
1 1 2 3 5 8 13 21 34 55 89
```

Hint: to print the numbers horizontally, i.e. without introducing a newline, use a comma after the value, i.e. `print var,` where **var** is the value to be printed.

1.3 (Extra credit, from Introduction to Python) Find prime numbers in a range of numbers. Ask the user for an integer, and display all prime numbers (i.e., numbers that have no even divisor above 1) between 2 and the user's number.

Sample program runs (user input in bold):

```
$ python homework1.3.py
* Prime Numbers *
Please enter max integer:  5
2
3
5
```

```
$ python homework1.3.py
* Prime Numbers *
Please enter max integer:  20
2
3
5
7
11
13
17
19
```

Hints:

1. The modulus operator **%** can be used to see whether one number divides evenly into another.
2. The standard approach is to use an outer loop to step through the range of numbers between 2 and the user's number,

1.4 (Extra credit, from Introduction to Python) Write the classic "number guessing" program in which you think of a number and the computer program attempts to guess it. See suggested output for behavioral details.

Hint: some challenges of precision may occur when narrowing down some guesses. Use the **round()** function to round any fractional guesses to the nearest integer value.

Speculative extra credit: have the program detect when you've changed your number (e.g., when you say "lower" at a point when you would previously have said "higher")! (Note that I haven't solved this one so it may or may not be easy!)

Example runs:

```
*Number Guessor*


Think of a number between 100 and 0, and I will try to guess it.  Hit
[Enter] to start.


is it 50 (yes/no/quit)?  no
is it higher or lower than 50?  lower


is it 25 (yes/no/quit)?  no
is it higher or lower than 25?  lower


is it 12 (yes/no/quit)?  no
is it higher or lower than 12?  higher


is it 19 (yes/no/quit)?  no
is it higher or lower than 19?  higher


is it 22 (yes/no/quit)?  no
is it higher or lower than 22?  higher


is it 24 (yes/no/quit)?  no
is it higher or lower than 24?  lower


is it 23 (yes/no/quit)?  yes
I knew it!
```