KMCI-935-A: Programming for Analytics  
Winter 2015

Tuesdays 6:30-9:30pm

Jennifer Farver, Adjunct Professor of Data Analytics

Director of Engineering, Civis Analytics

jennifer.farver@kellogg.northwestern.edu

# Assignment 1

Due: Tuesday, January 13, Noon.

## Part 0: Submission

Please submit the two .py files to Canvas.

Note: for all functions where arguments are specified, functions must be defined with arguments in THE SAME ORDER.

## Part 1: Hello Data!

In this exercise, we are going to explore a dataset of “Beach Water Quality” – a dataset of hourly measurements of several beach water metrics from the Chicago Park District’s automated sensors at Lake Michigan beaches. This data was obtained from the Chicago Data Portal.

Download the dataset from course website named “Sensors.csv”.

Write a python executable named: hello\_data.py

Place all the following 5 functions for Part 1 in hello\_data.py

Ensure that the following is at the top of the file:

#!/usr/bin/env python

1. Hello Data!

Write a function called hello\_data(), that takes no arguments and prints the phrase “Hello Data!”

Note: the printout should be without the quotes.

1. Read in the dataset and remove the header.

Write a function called read\_input(path), that takes a string path to a file.

Your function should return an array of all the rows in the dataset.

DO NOT return the header row

Each row should be an array of length 10 and should be of the form:

['Foster Weather Station', '12/08/2014 10:00:00 AM', '', '', '', '', '', '15.2', '12/08/2014 10:00 AM', 'FosterWeatherStation201412081000']

**From here on, we’ll call this a ‘reading.’**

1. Find low battery readings.

Using a list comprehension, write a function called:

find\_low\_battery\_readings(readings, threshold)

that takes (in this order):

* an array of readings, *readings*
* a float, *threshold*

Your function should return an array of readings such that each reading has a battery life lower than *threshold*.

1. Using a list comprehension, write another function:

battery\_dict(readings)

that takes an array of readings.

Your function should return readings in an array of dictionaries. Each dictionary should have two keys, *battery\_life* and *id* corresponding to the fields ‘Battery Life” and “Measurement ID” respectively.

{'battery\_life': '5.3', 'id': '63rdStreetBeach201408042000'}

1. Using a list comprehension, write another function:

month\_readings(readings, month\_number)

that takes

- an array of readings.

- an integer month number (i.e. 5 for May).

Your function should return an array of readings for that month.

Your function must use the datetime library to parse the datetime data. Hint: use the strptime() function.

## Part 2: Hello pandas!

Housing prices vary over time in different cities. In this exercise, we are going to see sales prices from four different cities in the United States including Atlanta, Chicago, Dallas and Oakland. This data is published by economist Robert Shiller.

There are four columns in each of these CSV files. They are “the first sale price” (purchase price), “second sale price”(sale price), “first sale date”(purchase date) and “second sale date”(sale date). The date in the last two columns is indicated in quarters, which will be further explained in the following instructions. There are no headers in these documents. More information on this data can be found through this link:

<http://www.econ.yale.edu/~shiller/data.htm>

Write a python executable named: hello\_pandas.py

Ensure that the following appears at the top of the file:

#!/usr/bin/env python

Place all functions for Part 2 in hello\_pandas.py

1. Read all files in a directory.

Write a function called:

read\_all\_files(files, column\_names, new\_column)

that takes three arguments:

* A string *files* which (using file system wildcards) points to all files you would like to read. Example: /my\_dir/\* points to all files in *my\_dir*
* An array of string that will be used as column names
* A string name of a new column

Your function should read each file specified by *files*.

Your function should return a single DataFrame with the column names as specified in the *column\_names* argument. The resulting data frame should be indexed with unique integers starting at 0.

For now, you can ignore the third argument.

Hint: Import and use the glob package to read multiple files.

Hint: Use pandas read\_csv and concat functions.

When you test your function, you can use an array of strings for column names such as:

columns = ['purchase\_price', 'sale\_price', 'purchase\_q\_since\_1970', 'sale\_q\_since\_1970']

These parameters should be passed when you call your program. DO NOT hard code them into your functions (we will later relax this constraint).

At this point, each row of your data frame should look like the following with the header row (Here “0” is the row index):

0 2550 3300 38 54

1. Now we’ll use the third argument in our function.

First, create a new function called:

add\_source\_column(frame, col, path).

This function should take three arguments:

* A data frame, *frame*
* The string, name of a column, *col*
* A string file path, *path*

Your function should populate another column, *col,* in your dataframe *f*, based on *path.* Strip the directory path and file extension to give just the city name. Do this with string manipulation.

Now, for each CSV file, if you call function add\_source\_column(f, col, path), you should get a new data frame with each row like:

0 2550 3300 38 54 atlanta

**Now**, update read\_all\_files() to call add\_source\_column().

Note: on Windows machines, file system paths are typically separated by backslashes (i.e. \) instead of forward slashes (i.e. /). Your function should work for one convention or the other. It does not need to work for both.

1. Add a year column.

This dataset contains data columns that are the number of the quarter since 1970. We want to know which year the quarter number indicates. For instance, “54” means the 54th quarter starting from Q1 1970, i.e. year 1983.

Write a new function called:

append\_year(frame, name)

Your function should take a DataFrame, *frame*, and an *existing* column name, *name,* and return the frame with a new column. The values in the new column should be the year for the column indicated by the column name, *name*. The new column should be called *name*\_year, (note: here the *“name*” should depend on the *name* from the argument passed in). For example, if *name* is ‘quarter\_num’ then the new column name should be ‘quarter\_num\_year’.

If we pass in the DataFrame above and *name* refers to the fourth column in our data, it should return a DataFrame with each row like:

0 2550 3300 38 54 atlanta 1983

**From here on, assume your DataFrames use the following column names:**

['purchase\_price', 'sale\_price', 'purchase\_q\_since\_1970', 'sale\_q\_since\_1970']

1. Find fast sales.

Write a function fast\_sales(frame, period) that takes a DataFrame, *frame,* and an integer, *period,* and returns all rows for homes that were sold within *period* quarters of their purchase.

Your function can add columns to *frame* if you wish.

1. Compute profit on fast sales.

Write a function profit\_from\_fast\_sales(frame, period) that takes a DataFrame of sales records *frame* and an integer, *period.*

Your function should return the mean profit (a number) for homes that were sold within *period* quarters of their purchase.

Your function can add columns to *frame* if you wish*.* You can call the function from (9).

1. Filter out sales by city.

Write a function sales\_by\_city(frame, city\_column)

Your function should take a DataFrame of sale records, *frame* and a column name, *city\_column,* which is the name of the column containing the city.

Your function should return the count of sales by city. For our datasets, the function should return four rows with each row like:

chicago 15530

1. Find the most profitable sales.

Write a function most\_profitable\_sales(frame)that takes a DataFrame, *frame,* of sale records.

Your function should return a DataFrame, which contains the 10 most profitable home sales. Your output DataFrame should contain at least two columns: *city\_name* and *profit*.