Lab 03 – pandas I

Submission: `lab03.ipynb`, pushed to your Git repo on `master`.
Points: 10
Due: Monday, January 27, 2pm

(NOTE: Lab 04 - pandas II coming Friday! Also due on Monday)

Objectives

- `pandas`

Introduction

Numpy is a core, foundational Python library for use in many domains. And, it works well. It is efficient, having largely been written in the C language. And, it works quite well when your data is clean, structured, and uniform types. However, most real-world data is large, messy, heterogeneous, and often incomplete, with missing values. Pandas has been developed for the data science community to aid in dealing with real world data. Indeed, it represents the vast majority of the data prepping, cleaning, filtering, munging, etc. that you will be doing. Therefore, you MUST become confident with `pandas`. This is the first part of a 2-part lab designed to get you going with `pandas`.

Create a **`lab03.ipynb`** file. Create your header cell as usual. Then, create your code first cell to set up your imports:

```
import sys
import numpy as np
import pandas as pd
```

A large portion of this lab is taken from snippets scattered throughout the enormous documentation and tutorials from the `pandas` website at http://pandas.pydata.org/pandas-docs/stable/index.html . In particular, this short intro will get you up to speed: http://pandas.pydata.org/pandas-docs/stable/10min.html pretty quickly with the most common tasks you'll be doing. It doesn't do a great job really explaining the "how" and "why" that's happening behind the scenes, but you'll learn those through experience. I strongly recommend that you work through the 10 minute introduction first, and don't just skim it, but try out the exercises presented there. It will make the work here so much easier to follow through. Then come back here to work through the exercises. More advanced `pandas` exercises are coming in part 2!

Exercises

1) [P] Report the Python, Numpy and Pandas version numbers.

2) [P] Now, show the result of `pd.show_versions()`. Write a comment in the cell about what this shows.

The start of Chapter 5 in McKinney's book, "Python for Data Analysis", has a nice explanation about the relationship between `numpy` and `pandas`. (See https://learning.oreilly.com/library/view/python-for-data/9781491957653/ch05.html . Also, don't forget the `pandas` user guide!) Let's go through some interview questions about `pandas`.

3) Explain the relationship between `numpy` and `pandas`. How are they tied together? How are they different? What core functionality does `pandas` add to `numpy`?

There are two key data structures in pandas: `Series` and `DataFrame`. It is critical that you take the time to understand these. A good source to understand the following questions is right from the user guide: https://pandas.pydata.org/pandas-docs/stable/getting_started/dsintro.html

4) [M] Compare and contrast `Series` and `DataFrame`.

5) [M] What are the data types that can be used to create a `Series` object in `pandas`?

6) [M] What are the data structures that can be used to create a `DataFrame` object in `pandas`?

7) [M] In the user guide, when creating a Series, the What role does the `index` parameter play when creating a `Series` object? Does the index always need to be specified? If not, what happens?

Pay close attention to this NOTE from their Intro to Data Structures documentation.

> **Note:** When the data is a dict, and an index is not passed, the `Series` index will be ordered by the dict's insertion order, if you're using Python version >= 3.6 and Pandas version >= 0.23.
>
> If you're using Python < 3.6 or Pandas < 0.23, and an index is not passed, the `Series` index will be the lexically ordered list of dict keys.

We're going to work with a very simple set of data, just to get you started. Enter the following Python lists in a cell in your notebook. They will represent some fictitious daily quiz scores for a couple of weeks of some course you are taking:

```
days = ["Mon","Tue","Wed","Thu","Fri"]
scores_1 = [9.5, 8.75, 8, 10, 7.75]
scores_2 = [9, 8, 10, 8.75, 7.25]
```

8) [P] Convert `scores_1` and `scores_2` into two `Series` objects. Use `days` as your index. You should name each Series object using the name parameter as "week_1", and "week_2". Do not show the result, only store the variables.

9) [P] Create a pandas `DataFrame` called `scores`, that represents the above data. Show your data frame. Your results should be arranged as shown below. You don't have the right answer until you can have the last line of your data from be scores, with a nicely formatted data frame rendered in your notebook. For example:

```
scores
```

	Mon	Tue	Wed	Thu	Fri
week_1	9.5	8.75	8.0	10.00	7.75
week_2	9.0	8.00	10.0	8.75	7.25

10) [P] Append the following data to the end of `scores`. `[8.5,8,9.75,9,6]`. The row label on the new week is `"week_3"`. Show your updated data frame. It should look like the following:

```
scores
```

	Mon	Tue	Wed	Thu	Fri
week_1	9.5	8.75	8.00	10.00	7.75
week_2	9.0	8.00	10.00	8.75	7.25
week_3	8.5	8.00	9.75	9.00	6.00

You are about to practice a lot of data selection and manipulation techniques. Developing the ability to quickly select data you are looking for is an important skill.

11) [M] `numpy` and `pandas` have an ENORMOUS number of ways for selecting data. At first, the flexibility will confuse and drive you nuts. In time, it because amazing and intuitive (with less nuts.)

    From their 10 minute tutorial:

    > **Note:** While standard Python / Numpy expressions for selecting and setting are intuitive and come in handy for interactive work, for production code, we recommend the optimized pandas data access methods, `.at`, `.iat`, `.loc` and `.iloc`.

    Write yourself a quick one sentence reference for each access method listed above, with one example for each using the `scores` DataFrame.

You are going to see so many ways to select data in a `DataFrame`. It will be confusing at first. For example, in addition to the above access methods, you can also use the `[ ]` operator directly to access your data. For example, try

`scores[0:1]` or `scores["Mon"]`

and understand how they are different. In addition, `DataFrames` dynamically create attributes for each column of data you have. For example, you could do either:

`scores["Mon"]` or `scores.Mon`

to access a single variable in your dataset. You'll see all the above techniques used in practice. I strongly recommend that you go through the 10 minute tutorial carefully, and briefly summarize what you can in your own notes. And, find some good cheat sheets, or make your own. AND – PAY CLOSE ATTENTION TO THE RETURN TYPES! Some return `DataFrame` objects, and other techniques return `Series` objects.

Let's continue.

12) [P] Show at least two different techniques to select scores for Tuesday using the string `"Tue"`

13) [P] Show how to retrieve the scores for Tuesday using the named attribute `Tue`

14) [P] Show at least three techniques to select scores for Wednesday using an integer. The return type can be either a DataFrame or a Series.

15) [P] Select the data for the first week using the string `"week_1"`. Your result should return a `Series` representing the scores for week 1.

16) [P] Select the data for the first week using the string `"week_1"`. Your result should return a `DataFrame`, representing the subset of the `scores DataFrame` for week 1 only.

17) [P] Select the data for the first week using a slice (HINT: `0:1`)

18) [P] Show at least two techniques to select the data for the second week using an integer

19) [P] Select Monday and Friday of the first and third week.

20) [P] Report the mean for each week

21) [P] For each week, report how much each day's score for that week differed from the mean for the week

22) [P] Report the maximum score for each week

23) [P] For each week, report which day had the largest score

24) [P] Report the week that had the highest total quiz score. Your answer should only be the name of the week from the index.

25) [P] Report the number of days of each week that had a score >= 9.25. Your answer should be a Series that looks like the following:

```
week_1    2
week_2    1
week_3    1
```

26) [P] Report the scores rescaled to fall between 0 and 100, instead of 0 to 10 as they are now.

Deliverables

Commit and push `lab03.ipynb`. Be sure you have every cell run, and output generated. Verify that your file is pushed properly on Gitlab.