*You are currently looking at **version 1.1** of this notebook. To download notebooks and datafiles, as well as get help on Jupyter notebooks in the Coursera platform, visit the Jupyter Notebook FAQ (https://www.coursera.org/learn/python-text-mining/resources/d9pwm) course resource.*

Assignment 1

In this assignment, you'll be working with messy medical data and using regex to extract relevant infromation from the data.

Each line of the `dates.txt` file corresponds to a medical note. Each note has a date that needs to be extracted, but each date is encoded in one of many formats.

The goal of this assignment is to correctly identify all of the different date variants encoded in this dataset and to properly normalize and sort the dates.

Here is a list of some of the variants you might encounter in this dataset:

- 04/20/2009; 04/20/09; 4/20/09; 4/3/09
- Mar-20-2009; Mar 20, 2009; March 20, 2009; Mar. 20, 2009; Mar 20 2009;
- 20 Mar 2009; 20 March 2009; 20 Mar. 2009; 20 March, 2009
- Mar 20th, 2009; Mar 21st, 2009; Mar 22nd, 2009
- Feb 2009; Sep 2009; Oct 2010
- 6/2008; 12/2009
- 2009; 2010

Once you have extracted these date patterns from the text, the next step is to sort them in ascending chronological order accoring to the following rules:

- Assume all dates in xx/xx/xx format are mm/dd/yy
- Assume all dates where year is encoded in only two digits are years from the 1900's (e.g. 1/5/89 is January 5th, 1989)
- If the day is missing (e.g. 9/2009), assume it is the first day of the month (e.g. September 1, 2009).
- If the month is missing (e.g. 2010), assume it is the first of January of that year (e.g. January 1, 2010).
- Watch out for potential typos as this is a raw, real-life derived dataset.

With these rules in mind, find the correct date in each note and return a pandas Series in chronological order of the original Series' indices.

For example if the original series was this:

```
0    1999
1    2010
2    1978
3    2015
4    1985
```

Your function should return this:

```
0    2
1    4
2    0
3    1
4    3
```

Your score will be calculated using Kendall's tau (https://en.wikipedia.org/wiki/Kendall_rank_correlation_coefficient), a correlation measure for ordinal data.

In [4]:
```
import pandas as pd

doc = []
with open('dates.txt') as file:
    for line in file:
        doc.append(line)

df = pd.Series(doc)
df.head(10)
```

Out[4]:
```
0         03/25/93 Total time of visit (in minutes):\n
1                         6/18/85 Primary Care Doctor:\n
2    sshe plans to move as of 7/8/71 In-Home Servic...
3                 7 on 9/27/75 Audit C Score Current:\n
4    2/6/96 sleep studyPain Treatment Pain Level (N...
5                      .Per 7/06/79 Movement D/O note:\n
6    4, 5/18/78 Patient's thoughts about current su...
7    10/24/89 CPT Code: 90801 - Psychiatric Diagnos...
8                           3/7/86 SOS-10 Total Score:\n
9              (4/10/71)Score-1Audit C Score Current:\n
dtype: object
```

In [5]:
```
def date_sorter():

    # Your code here
    #Extract the following, use .extract so that rows in which pattern is not
 found will become NaN.
    ##'Mar 20 2009',Feb 2009; Sep 2009; Oct 2010, #'Mar 20, 2009', 'March 20,
 2009', 'Mar. 20, 2009'
    #Mar-20-2009, 20 Mar 2009; 20 March 2009; 20 Mar. 2009; 20 March, 2009, Ma
r 20th, 2009; Mar 21st, 2009; Mar 22nd, 2009
    first_set = df.str.extract((r'((?:\d{,2}\s)?(?:Jan|Feb|Mar|Apr|May|Jun|Jul
|Aug|Sep|Oct|Nov|Dec)[a-z]*(?:-|\.|\s|,)\s?\d{,2}[a-z]*(?:-|,|\s)?\s?\d{2,4})'
))

    ##'04/20/2009', '04/20/09', '4/20/09', '4/3/09'
    second_set = df.str.extract(r'((?:\d{1,2})(?:(?:\/|-)\d{1,2})(?:(?:\/|-)\d
{2,4}))')

    #6/2008; 12/2009, 2009; 2010
    third_set = df.str.extract(r'((?:\d{1,2}(?:-|\/))?\d{4})')

    #Combine the 3 sets above using fillna.
    full_set = first_set.fillna(second_set).fillna(third_set)

    #Correct spellings of months
    full_set = full_set.replace('Decemeber','December', regex=True)
    full_set = full_set.replace('Janaury','January',regex=True)


    #convert all dates to date time
    full_set = pd.to_datetime(full_set)

    #Sort series
    full_set_sorted = full_set.sort_values()
    #Get index series
    index_series = pd.Series(full_set_sorted.index)

    return index_series

date_sorter()
```

```
Out[5]: 0      9
        1     84
        2      2
        3     53
        4     28
        5    474
        6    153
        7     13
        8    129
        9     98
        10   111
        11   225
        12    31
        13   171
        14   191
        15   486
        16   335
        17   415
        18    36
        19   405
        20   323
        21   422
        22   375
        23   380
        24   345
        25    57
        26   481
        27   436
        28   104
        29   299
             ...
        470  220
        471  208
        472  243
        473  139
        474  320
        475  383
        476  244
        477  286
        478  480
        479  431
        480  279
        481  198
        482  381
        483  463
        484  366
        485  439
        486  255
        487  401
        488  475
        489  257
        490  152
        491  235
        492  464
        493  253
        494  427
        495  231
```

```
496    141
497    186
498    161
499    413
Length: 500, dtype: int64
```