*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.

This function should return a Series of length 500 and dtype int.

In [1]:

```
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[1]:

```
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 [3]:

```
def date_sorter():

    # Your code here
    dates_extracted = df.str.extractall(r'(?P<origin>(?P<month>\d?\d)[/|-](?P<day>\d?\d)[/|
    index_left = ~df.index.isin([x[0] for x in dates_extracted.index])
    dates_extracted = dates_extracted.append(df[index_left].str.extractall(r'(?P<origin>(?P
    index_left = ~df.index.isin([x[0] for x in dates_extracted.index])
    del dates_extracted[3]
    del dates_extracted[4]
    dates_extracted = dates_extracted.append(df[index_left].str.extractall(r'(?P<origin>(?P
    index_left = ~df.index.isin([x[0] for x in dates_extracted.index])
    dates_extracted = dates_extracted.append(df[index_left].str.extractall(r'(?P<origin>(?P
    del dates_extracted[3]
    index_left = ~df.index.isin([x[0] for x in dates_extracted.index])

    # Without day
    dates_without_day = df[index_left].str.extractall('(?P<origin>(?P<month>[A-Z][a-z]{2,})
    dates_without_day = dates_without_day.append(df[index_left].str.extractall(r'(?P<origir
    dates_without_day['day'] = 1
    dates_extracted = dates_extracted.append(dates_without_day)
    index_left = ~df.index.isin([x[0] for x in dates_extracted.index])

    # Only year
    dates_only_year = df[index_left].str.extractall(r'(?P<origin>(?P<year>\d{4}))')
    dates_only_year['day'] = 1
    dates_only_year['month'] = 1
    dates_extracted = dates_extracted.append(dates_only_year)
    index_left = ~df.index.isin([x[0] for x in dates_extracted.index])

    # Year
    dates_extracted['year'] = dates_extracted['year'].apply(lambda x: '19' + x if len(x) ==
    dates_extracted['year'] = dates_extracted['year'].apply(lambda x: str(x))

    # Month
    dates_extracted['month'] = dates_extracted['month'].apply(lambda x: x[1:] if type(x) is
    month_dict = dict({'September': 9, 'Mar': 3, 'November': 11, 'Jul': 7, 'January': 1, 'D
                       'Feb': 2, 'May': 5, 'Aug': 8, 'Jun': 6, 'Sep': 9, 'Oct': 10, 'June':
                       'February': 2, 'Dec': 12, 'Apr': 4, 'Jan': 1, 'Janaury': 1,'August':
                       'July': 7, 'Since': 1, 'Nov': 11, 'April': 4, 'Decemeber': 12, 'Age'
    dates_extracted.replace({"month": month_dict}, inplace=True)
    dates_extracted['month'] = dates_extracted['month'].apply(lambda x: str(x))

    # Day
    dates_extracted['day'] = dates_extracted['day'].apply(lambda x: str(x))

    # Cleaned date
    dates_extracted['date'] = dates_extracted['month'] + '/' + dates_extracted['day'] + '/'
    dates_extracted['date'] = pd.to_datetime(dates_extracted['date'])

    dates_extracted.sort_values(by='date', inplace=True)
    df1 = pd.Series(list(dates_extracted.index.labels[0]))

    return df1# Your answer here
```