Session 13: Analyzing Text Data in Pandas

Setting up

```
[1]: import pandas as pd
     courses=pd.read_excel('Marshall_Course_Enrollment_1516_1617.xlsx')\
         [['Course','First Instructor','First End Time','First Days','First Room','Term','Tit
     courses.columns=['course','instructor','end_time','days','room','term','title']
     courses['end_time']=courses['end_time'].astype(str)
     courses.head()
```

```
     course         instructor  end_time days    room   term  \
0  ACCT-370  Hopkins, Merle, W  11:50:00    F  SLH200  20153
1  ACCT-370  Hopkins, Merle, W  09:50:00   MW  ACC303  20153
2  ACCT-370  Hopkins, Merle, W  11:50:00   MW  ACC303  20153
3  ACCT-370  Hopkins, Merle, W  13:50:00   MW  ACC303  20153
4  ACCT-371                NaN  11:50:00    F  SLH200  20153

                                title
0  External Financial Reporting Issues
1  External Financial Reporting Issues
2  External Financial Reporting Issues
3  External Financial Reporting Issues
4   Introduction to Accounting Systems
```

```
[2]: instructors=courses['instructor'].drop_duplicates().head()
     instructors
```

```
0          Hopkins, Merle, W
4                        NaN
5     Porter, Leslie, Robert
9            Karuna, Christo
13           Layton, Rose, M
Name: instructor, dtype: object
```

1. Applying a Function to an Entire Series

```
[3]: s=instructors[0]
     s
```

```
'Hopkins, Merle, W'
```

```
[4]: s.split(',')
```

```
['Hopkins', ' Merle', ' W']
```

```
[5]: l=s.split(',')
     l[1].strip()
```

```
'Merle'
```

```
[6]: from numpy import nan
     def getFirst(s):
         if type(s)!=str:
```

```
            return nan
        l=s.split(',')
        if len(l)<=1:
            return nan
        return l[1].strip()
    getFirst('Shi, Peng')
```

```
'Peng'
```

```
[7]: getFirst(nan)
```

```
nan
```

```
[8]: getFirst('Shi')
```

```
nan
```

```
[9]: instructors.apply(getFirst)
```

```
0       Merle
4         NaN
5      Leslie
9     Christo
13       Rose
Name: instructor, dtype: object
```

Q1: Create the following columns in the DataFrame "courses":

- first_name: the first name of the instructor. Use NaN if not available.
- last_name: the last name of the instructor. Use Nan if not available.

```
[13]: courses[['instructor','first_name','last_name']].head()
```

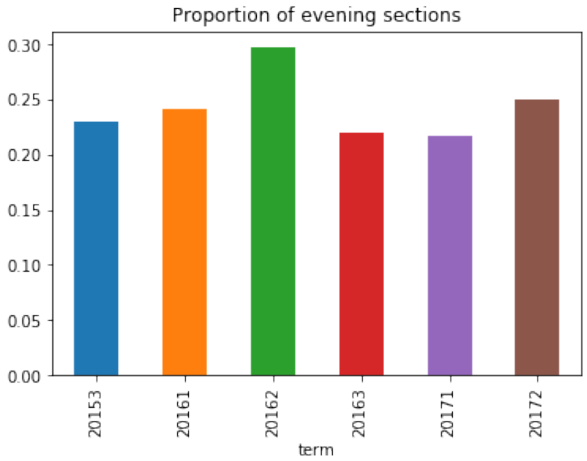
```
          instructor first_name last_name
0  Hopkins, Merle, W      Merle   Hopkins
1  Hopkins, Merle, W      Merle   Hopkins
2  Hopkins, Merle, W      Merle   Hopkins
3  Hopkins, Merle, W      Merle   Hopkins
4                NaN        NaN       NaN
```

Q2: Extract the hour from the column "end_time", and convert it to integers using "pd.to_numeric" (with errors='coerce'). Then create a columns called "evening" in the DataFrame "courses", corresponding to whether the hour is greater than or equal to 18.

Once you have created the column, you can compute the proportion of evening courses for each term.

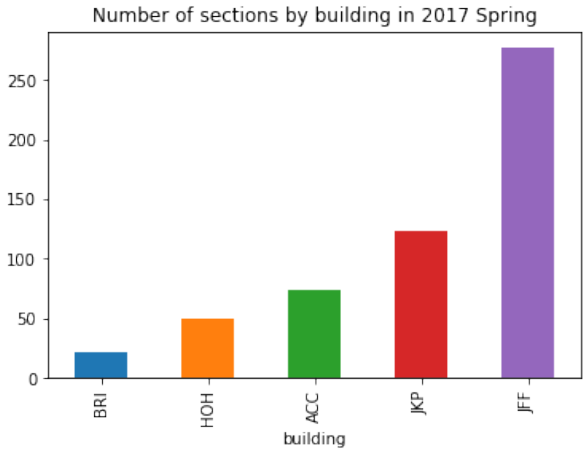
```
[38]: courses.groupby('term')['evening'].mean()\
          .plot(kind='bar',title='Proportion of evening sections')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f19128585c0>
```



2. Vectorized String Methods in Pandas

```
[19]: room=courses['room'].drop_duplicates()
      room.head()

0       SLH200
1       ACC303
8     HOH EDI
9       ACC310
22      HOH305
Name: room, dtype: object

[20]: room.str[:3].head()

0     SLH
1     ACC
8     HOH
9     ACC
22    HOH
Name: room, dtype: object

[21]: courses['building']=courses['room'].str[:3]
      courses[courses['building'].isin(['ACC','HOH','BRI','JKP','JFF'])]\
          .query('term==20171')\
          .groupby('building')['course'].count().sort_values()\
          .plot(kind='bar',title='Number of sections by building in 2017 Spring')

<matplotlib.axes._subplots.AxesSubplot at 0x7f1912ab1908>
```



List of Available Series.str.XXX methods: https://pandas.pydata.org/pandas-docs/version/0.21/api.html#string-handling

```
[22]: courses['days'].head()

0     F
1    MW
2    MW
3    MW
4     F
Name: days, dtype: object

[23]: courses['days'].str.contains('M').head()

0    False
1     True
2     True
3     True
4    False
Name: days, dtype: object

[24]: courses.query('term==20171')['days'].str.contains('M').mean()

0.38487394957983195

[25]: import re
      courses['days'].str.contains('m',flags=re.IGNORECASE).head()

0    False
1     True
2     True
3     True
4    False
Name: days, dtype: object

[26]: courses['course'].head()

0    ACCT-370
1    ACCT-370
```

```
2    ACCT-370
3    ACCT-370
4    ACCT-371
Name: course, dtype: object
```

```
[27]: courses['course'].str.split('-').head()
```

```
0    [ACCT, 370]
1    [ACCT, 370]
2    [ACCT, 370]
3    [ACCT, 370]
4    [ACCT, 371]
Name: course, dtype: object
```

```
[28]: courses['course'].str.split('-').str[0].head()
```

```
0    ACCT
1    ACCT
2    ACCT
3    ACCT
4    ACCT
Name: course, dtype: object
```

```
[29]: courses['course'].str.split('-').str[1].head()
```

```
0    370
1    370
2    370
3    370
4    371
Name: course, dtype: object
```

Q3: Redo Q1 and Q2 using vectorized string methods already in Pandas, rather than writing your own function and calling Series.apply.

```
[31]: # Checking outputs for Q1
      courses[['first_name','last_name']].head()
```

```
  first_name last_name
0      Merle   Hopkins
1      Merle   Hopkins
2      Merle   Hopkins
3      Merle   Hopkins
4        NaN       NaN
```

```
[32]: courses['first_name'][0]
```

```
'Merle'
```

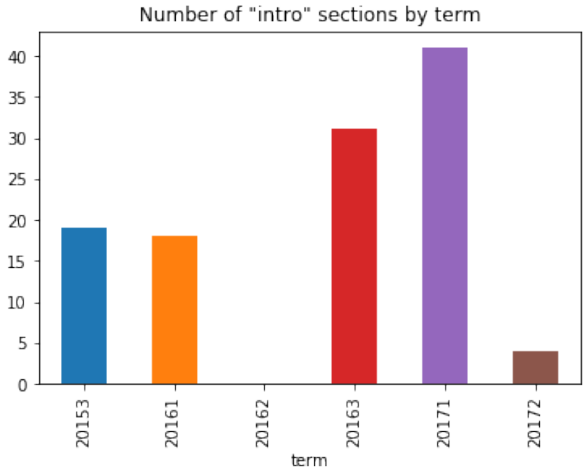
```
[34]: # Checking outputs for Q2
      courses.groupby('term')['evening'].mean()
```

```
term
20153    0.229814
20161    0.241379
20162    0.297297
20163    0.219911
20171    0.216590
20172    0.250000
Name: evening, dtype: float64
```

Q4: Plot the number of sections by term whose title contains the string "intro" (ignoring cases).

[35]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f191279eb70>
```



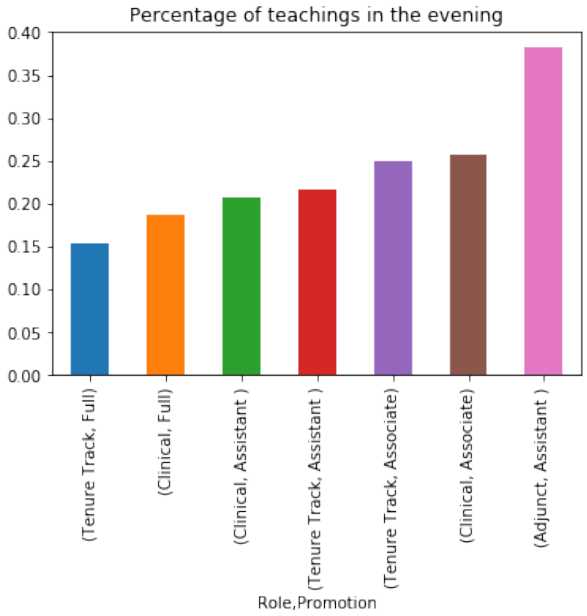
(Optional) Q5: Run the following code to load in professor information, and merge it with the "courses" DataFrame using the "first_name" and "last_name" columns you have created. Then group by the "Role" and "Promotion" of professors and plot the percentage of sections taught by each group scheduled in the evening, as follows.

[36]:
```
professors=pd.read_csv('Professor_information.csv',encoding='latin1')\
    [['First_Name','Last_Name','Role','Promotion']]
professors.head()
```

	First_Name	Last_Name	Role	Promotion
0	Arif	Ansari	Clinical	Full
1	Yehuda	Bassok	Tenure Track	Full
2	Murat	Bayiz	Clinical	Associate
3	Jacob	Bien	Tenure Track	Assistant
4	Sriram	Dasu	Tenure Track	Associate

[37]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f191290e0b8>
```


Percentage of teachings in the evening
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
(Tenure Track, Full)
(Clinical, Full)
(Clinical, Assistant)
(Tenure Track, Assistant)
(Tenure Track, Associate)
(Clinical, Associate)
(Adjunct, Assistant)
Role,Promotion