

# iPython Notebook and PANDAS Cookbook

More and more of my research involves some degree of 'Big Data' – typically datasets with a million or so tweets. Getting these data prepped for analysis can involve massive amounts of data manipulation – anything from aggregating data to the daily or organizational level, to merging in additional variables, to generating data required for social network analysis. For all such steps I now almost exclusively use Python's [PANDAS](#) library ('Python Data Analysis Library'). In conjunction with the [iPython Notebook](#) interactive computing framework and [NetworkX](#), you will have a powerful set of analysis tools at your disposal.

Given that I am now doing almost all of my dataset manipulation – and much of the analysis – in [PANDAS](#), and how new I am to the framework, I created this page mostly as a handy reference for all those PANDAS commands I tend to forget or find particularly useful. But if it proves helpful to any others, great!

## iPython Notebook Settings

Set width of columns for display:

```
pd.set_option('display.max_colwidth', 200)
```

Set cell width:

```
pd.set_option('max_colwidth', 200)
```

## Analysis

Cross-tab (can be saved as dataframe):

```
pd.crosstab(df['8-K filing_info'], df['8-K Item'])
```

## Generating New Variables, Arrays, etc.

Create list from dataframe column:

```
screen_names = ticker_master['screen_name'].tolist()
```

Create list of unique column values in DF:

```
tickers_in_df = pd.unique(df.ticker.ravel()).tolist()
```

Convert string variable to float:

```
df['count_V2'] = df['count'].convert_objects(convert_numeric=True)
```

Convert float column to int (only works if there are no missing values):

```
df_2014['rt_count'] = df_2014['rt_count'].astype(int)
```

Convert column to string:

```
df.index = df.index.astype(str)
```

Create new variable as a slice of an existing one:

```
df['cusip8'] = df['cusip'].apply(lambda x: x[:8])
```

Replace a word *within* a column with another word:

```
df['8-K'] = df['8-K'].str.replace('Item', 'Section')
```

Fill in missing values for one column with zero:

```
df_2014['rt_count'] = df_2014['rt_count'].fillna(0)
```

Get new list of unique items in a list:

```
screen_names_unique = list(set(screen_names))
```

Create dummy variable based on whether another column contains specific text (values will be 'True' and 'False'):

```
df['retweeted_status_dummy'] =  
df['retweeted_status'].str.contains('THIS', na=False)
```

Then convert to float (will convert 'True' and 'False' categories of above variable into '1' and '0', respectively):

```
df['retweeted_status_dummy'] =  
df['retweeted_status_dummy'].astype(float)
```

Replace values (here, replace 'None' with '0'):

```
df['reply_message'] =  
df['in_reply_to_screen_name'].replace([None], ['0'])
```

Replace values (here, replace np.nan values with '0'):

```
df.ix[df.Number_of_retweets.isnull(),  
'Number_of_retweets'] = 0
```

Switch values of '0' and '1':

```
df['reply_message'] = df['reply_message'].replace([1],  
[99]).replace([0],[1]).replace([99],[0])
```

Create binary variable from count variable (if old var=0, assign value of 0; otherwise 1):

```
df['urls'] = np.where(df['entities_urls_count']==0, 0, 1)
```

Change each unicode element in a list to string:

```
tickers_in_df = [x.encode('UTF8') for x in tickers_in_df]
```

Change column values to upper case:

```
df['Name'] = df['Name'].apply(lambda x: x.upper())
```

Change column values to upper case:

```
sec['8-K Item'] = sec['8-K Item'].str.upper()
```

Find number of unique values:

```
len(pd.unique(compustat_2013.cusip.ravel()))
```

Add leading zeros to string variable (as many as needed to reach 10 characters):

```
df['cik_x'] = df['cik_x'].apply(lambda x: x.zfill(10))
```

Convert column to string and add leading zeros:

```
df['CIK_10'] = df['CIK'].astype(str).apply(lambda x: x.zfill(10))
```

Get a list of values from a DF column:

```
values_list = df['8-K'].value_counts().keys().tolist()
```

Find number of cases with fewer than the mean value:

```
sum(df['followers_count'] < df['followers_count'].mean())
```

## I/O

Read in a pickled dataframe:

```
ticker_master = pd.read_pickle('valid_tickers_317.pkl')
```

Read in JSON file:

```
f = open('valid_tickers_list_317.json', 'r')
valid_tickers = simplejson.load(f)
```

Read in JSON file – method 2:

```
with open('my_list.json', 'r') as fp:
    my_list = json.load(fp)
```

Save a list as JSON file:

```
f = open('all_valid_screen_names.json', 'w')
simplejson.dump(valid_twitter_accounts, f)
f.close()
```

Save a list as JSON file – method 2:

```
import json
```

```
with open('my_list.json', 'w') as fp:
    json.dump(my_list, fp)
```

Read in Excel file:

```
twitter_accounts = pd.read_excel('Twitter Account-level
Database – non-excluded tickers and accounts only.xlsx',
'accounts', header=0)
```

Write Excel file:

```
df.to_excel('df.xls', sheet_name='Sheet1')
```

## Looping

Looping over rows (note how to select a slice of rows):

```
for index, row in df[:10].iterrows():
    print index, row['content']
```

Loop over rows and update existing variable:

```
for index, row in df.iterrows():
    df['count'] = count
```

Loop over rows and create new variable, method 1:

```
for index, row in df.iterrows():
    #df.ix[df.index==index, 'count'] = count #LONGER
VERSION
    df.ix[index, 'count'] = count #SHORTER VERSION
```

Loop over rows and create new variable, method 2:

```
for index, row in df.iterrows():
    df.loc[index, 'count'] = count
```

## Time Series

Weekdays:

```
1]
df2
```

Create version of dataframe with conditions on two variables (for removing a duplicate firm):

```
merged = merged[~((merged['fyear']==2012) &
(merged['gvkey']=='176103'))]
```

Select partial dataframe – complex conditions:

```
sec[(sec['Form Type'] == '8-K') & (sec['8-K Item']==") &
(sec['8-K Item - V2']==") | (sec['8-K Item']=='Item fo') &
(sec['8-K Item - V2']==") | (sec['8-K Item']=='Item that')
& (sec['8-K Item - V2']==")]
```

Merge two dataframes:

```
merged = pd.merge(fb, org_data, left_on='feed_id',
right_on='Org_ID')
```

Deep copy of DF:

```
df2 = df.copy(deep=True)
```

Get a slice of dataframe (here, the two rows with the given indexes):

```
df.loc[11516:11517]
```

# Custom Twitter Variables

Time on Twitter:

```
import datetime
```

```
df['start']=np.datetime64('2016-04-01')
```

```
df['created_at'] = pd.to_datetime(df['created_at'])
```

```
df['time_on_twitter'] = df['start'] - df['created_at']
```

```
df['time_on_twitter_days'] =  
df['time_on_twitter'].astype('timedelta64[D]')
```

```
df['time_on_twitter_days'] =  
df.time_on_twitter_days.astype('int')
```

---

## Random Python Recipes

This page is mostly for me as a handy reference for all those Python commands I tend to forget. That said, if it proves helpful to any others, all the better!

### Lists

Create list by slicing items of another list:

Combine two lists into a dictionary:

Get list of all files in a directory:

Find items in one list that are not in another:

# Dictionary

Merge two dictionaries (same keys in both dictionaries):

Same as above, but values as list instead of tuple:

For converting to a PANDAS dataframe, you would normally want something like this – create a dictionary of dictionaries and add keys:

To convert the above dictionary to a dataframe:

And you would want this if you are building a dictionary with more than one month:

To merge the above nested dictionary, you have to do something more complicated to convert to a DF:

Sort dictionary by keys:

Sort dictionary by numerical values:

# Regular Expressions

Find all instances of text in string starting with 'Item ':

Find all instances of text that start with 'Item ' plus any number(s) then '.' then any number(s):

# Miscellaneous

Loop over directory and find and open specific file:



Rename a file: