Jupyter Notebook Viewer

In [1]:

```
%load_ext watermark
%watermark -a 'Sebastian Raschka' -v -d -p pandas
```

```
Sebastian Raschka 28/01/2015

CPython 3.4.2
IPython 2.3.1

pandas 0.15.2
```

[More information](http://nbviewer.ipython.org/github/rasbt/python_reference/blob/master/ipython_magic/watermark.ipynb) about the `watermark` magic command extension.

This is just a small but growing collection of pandas snippets that I find occasionally and particularly useful -- consider it as my personal notebook. Suggestions, tips, and contributions are very, very welcome!

I am heavily into sports prediction (via a machine learning approach) these days. So, let us use a (very) small subset of the soccer data that I am just working with.

In [2]:

```
import pandas as pd

df = pd.read_csv('https://raw.githubusercontent.com/rasbt/python_reference/master/Data/some_soccer_data.csv')
df
```

Out[2]:

	PLAYER	SALARY	GP	G	A	SOT	PPG	P
0	Sergio Agüero\n Forward — Manchester City	$19.2m	16	14	3	34	13.12	209.98
1	Eden Hazard\n Midfield — Chelsea	$18.9m	21	8	4	17	13.05	274.04
2	Alexis Sánchez\n Forward — Arsenal	$17.6m	NaN	12	7	29	11.19	223.86
3	Yaya Touré\n Midfield — Manchester City	$16.6m	18	7	1	19	10.99	197.91
4	Ángel Di María\n Midfield — Manchester United	$15.0m	13	3	NaN	13	10.17	132.23
5	Santiago Cazorla\n Midfield — Arsenal	$14.8m	20	4	NaN	20	9.97	NaN
6	David Silva\n Midfield — Manchester City	$14.3m	15	6	2	11	10.35	155.26
7	Cesc Fàbregas\n Midfield — Chelsea	$14.0m	20	2	14	10	10.47	209.49
8	Saido Berahino\n Forward — West Brom	$13.8m	21	9	0	20	7.02	147.43
9	Steven Gerrard\n Midfield — Liverpool	$13.8m	20	5	1	11	7.50	150.01

Converting Column Names to Lowercase¶

In [3]:

```
# Converting column names to lowercase

df.columns = [c.lower() for c in df.columns]

# or
# df.rename(columns=lambda x : x.lower())

df.tail(3)
```

Out[3]:

	player	salary	gp	g	a	sot	ppg	p

	player	salary	gp	g	a	sot	ppg	p
7	Cesc Fàbregas\n Midfield — Chelsea	$14.0m	20	2	14	10	10.47	209.49
8	Saido Berahino\n Forward — West Brom	$13.8m	21	9	0	20	7.02	147.43
9	Steven Gerrard\n Midfield — Liverpool	$13.8m	20	5	1	11	7.50	150.01

Renaming Particular Columns¶

In [4]:

```
df = df.rename(columns={'p': 'points',
                        'gp': 'games',
                        'sot': 'shots_on_target',
                        'g': 'goals',
                        'ppg': 'points_per_game',
                        'a': 'assists',})

df.tail(3)
```

Out[4]:

	player	salary	games	goals	assists	shots_on_target	points_per_game	points
7	Cesc Fàbregas\n Midfield — Chelsea	$14.0m	20	2	14	10	10.47	209.49
8	Saido Berahino\n Forward — West Brom	$13.8m	21	9	0	20	7.02	147.43
9	Steven Gerrard\n Midfield — Liverpool	$13.8m	20	5	1	11	7.50	150.01

Changing Values in a Column¶

In [5]:

```
# Processing `salary` column

df['salary'] = df['salary'].apply(lambda x: x.strip('$m'))
df.tail()
```

Out[5]:

	player	salary	games	goals	assists	shots_on_target	points_per_game	points
5	Santiago Cazorla\n Midfield — Arsenal	14.8	20	4	NaN	20	9.97	NaN
6	David Silva\n Midfield — Manchester City	14.3	15	6	2	11	10.35	155.26
7	Cesc Fàbregas\n Midfield — Chelsea	14.0	20	2	14	10	10.47	209.49
8	Saido Berahino\n Forward — West Brom	13.8	21	9	0	20	7.02	147.43
9	Steven Gerrard\n Midfield — Liverpool	13.8	20	5	1	11	7.50	150.01

In [6]:

```
df['team'] = pd.Series('', index=df.index)

# or
df.insert(loc=8, column='position', value='')

df.tail(3)
```

Out[6]:

	player	salary	games	goals	assists	shots_on_target	points_per_game	points	position	team
7	Cesc Fàbregas\n Midfield — Chelsea	14.0	20	2	14	10	10.47	209.49		
8	Saido Berahino\n Forward — West Brom	13.8	21	9	0	20	7.02	147.43		
9	Steven Gerrard\n Midfield — Liverpool	13.8	20	5	1	11	7.50	150.01		

In [7]:

```
# Processing `player` column

def process_player_col(text):
    name, rest = text.split('\n')
    position, team = [x.strip() for x in rest.split(' — ')]
    return pd.Series([name, team, position])

df[['player', 'team', 'position']] = df.player.apply(process_player_col)

# modified after tip from reddit.com/user/hharison
#
# Alternative (inferior) approach:
#
#for idx,row in df.iterrows():
#    name, position, team = process_player_col(row['player'])
#    df.ix[idx, 'player'], df.ix[idx, 'position'], df.ix[idx, 'team'] = name, position, team

df.tail(3)
```

Out[7]:

	player	salary	games	goals	assists	shots_on_target	points_per_game	points	position	team
7	Cesc Fàbregas	14.0	20	2	14	10	10.47	209.49	Midfield	Chelsea
8	Saido Berahino	13.8	21	9	0	20	7.02	147.43	Forward	West Brom
9	Steven Gerrard	13.8	20	5	1	11	7.50	150.01	Midfield	Liverpool

Applying Functions to Multiple Columns¶

In [8]:

```
cols = ['player', 'position', 'team']
df[cols] = df[cols].applymap(lambda x: x.lower())
df.head()
```

Out[8]:

	player	salary	games	goals	assists	shots_on_target	points_per_game	points	position	team
0	sergio agüero	19.2	16	14	3	34	13.12	209.98	forward	manchester city
1	eden hazard	18.9	21	8	4	17	13.05	274.04	midfield	chelsea
2	alexis sánchez	17.6	NaN	12	7	29	11.19	223.86	forward	arsenal
3	yaya touré	16.6	18	7	1	19	10.99	197.91	midfield	manchester city
4	ángel di maría	15.0	13	3	NaN	13	10.17	132.23	midfield	manchester united

In [9]:

```
nans = df.shape[0] - df.dropna().shape[0]

print('%d rows have missing values' % nans)
```

```
3 rows have missing values
```

In [10]:

```
# Selecting all rows that have NaNs in the `assists` column

df[df['assists'].isnull()]
```

Out[10]:

	player	salary	games	goals	assists	shots_on_target	points_per_game	points	position	team
4	ángel di maría	15.0	13	3	NaN	13	10.17	132.23	midfield	manchester united

	player	salary	games	goals	assists	shots_on_target	points_per_game	points	position	team
5	santiago cazorla	14.8	20	4	NaN	20	9.97	NaN	midfield	arsenal

In [11]:

```
df[df['assists'].notnull()]
```

Out[11]:

	player	salary	games	goals	assists	shots_on_target	points_per_game	points	position	team
0	sergio agüero	19.2	16	14	3	34	13.12	209.98	forward	manchester city
1	eden hazard	18.9	21	8	4	17	13.05	274.04	midfield	chelsea
2	alexis sánchez	17.6	NaN	12	7	29	11.19	223.86	forward	arsenal
3	yaya touré	16.6	18	7	1	19	10.99	197.91	midfield	manchester city
6	david silva	14.3	15	6	2	11	10.35	155.26	midfield	manchester city
7	cesc fàbregas	14.0	20	2	14	10	10.47	209.49	midfield	chelsea
8	saido berahino	13.8	21	9	0	20	7.02	147.43	forward	west brom
9	steven gerrard	13.8	20	5	1	11	7.50	150.01	midfield	liverpool

In [12]:

```
# Filling NaN cells with default value 0


df.fillna(value=0, inplace=True)
df
```

Out[12]:

	player	salary	games	goals	assists	shots_on_target	points_per_game	points	position	team
0	sergio agüero	19.2	16	14	3	34	13.12	209.98	forward	manchester city
1	eden hazard	18.9	21	8	4	17	13.05	274.04	midfield	chelsea
2	alexis sánchez	17.6	0	12	7	29	11.19	223.86	forward	arsenal
3	yaya touré	16.6	18	7	1	19	10.99	197.91	midfield	manchester city
4	ángel di maría	15.0	13	3	0	13	10.17	132.23	midfield	manchester united
5	santiago cazorla	14.8	20	4	0	20	9.97	0.00	midfield	arsenal
6	david silva	14.3	15	6	2	11	10.35	155.26	midfield	manchester city
7	cesc fàbregas	14.0	20	2	14	10	10.47	209.49	midfield	chelsea
8	saido berahino	13.8	21	9	0	20	7.02	147.43	forward	west brom
9	steven gerrard	13.8	20	5	1	11	7.50	150.01	midfield	liverpool

In [13]:

```
# Adding an "empty" row to the DataFrame


import numpy as np


df = df.append(pd.Series(
                [np.nan]*len(df.columns), # Fill cells with NaNs
                index=df.columns),
                ignore_index=True)


df.tail(3)
```

Out[13]:

	player	salary	games	goals	assists	shots_on_target	points_per_game	points	position	team
8	saido berahino	13.8	21	9	0	20	7.02	147.43	forward	west brom
9	steven gerrard	13.8	20	5	1	11	7.50	150.01	midfield	liverpool
10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [14]:

```
# Filling cells with data


df.loc[df.index[-1], 'player'] = 'new player'
df.loc[df.index[-1], 'salary'] = 12.3
df.tail(3)
```

Out[14]:

	player	salary	games	goals	assists	shots_on_target	points_per_game	points	position	team
8	saido berahino	13.8	21	9	0	20	7.02	147.43	forward	west brom
9	steven gerrard	13.8	20	5	1	11	7.50	150.01	midfield	liverpool
10	new player	12.3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [15]:

```
# Sorting the DataFrame by a certain column (from highest to lowest)


df.sort('goals', ascending=False, inplace=True)
df.head()
```

Out[15]:

	player	salary	games	goals	assists	shots_on_target	points_per_game	points	position	team
0	sergio agüero	19.2	16	14	3	34	13.12	209.98	forward	manchester city
2	alexis sánchez	17.6	0	12	7	29	11.19	223.86	forward	arsenal
8	saido berahino	13.8	21	9	0	20	7.02	147.43	forward	west brom
1	eden hazard	18.9	21	8	4	17	13.05	274.04	midfield	chelsea
3	yaya touré	16.6	18	7	1	19	10.99	197.91	midfield	manchester city

In [16]:

```
# Optional reindexing of the DataFrame after sorting


df.index = range(1,len(df.index)+1)
df.head()
```

Out[16]:

	player	salary	games	goals	assists	shots_on_target	points_per_game	points	position	team
1	sergio agüero	19.2	16	14	3	34	13.12	209.98	forward	manchester city
2	alexis sánchez	17.6	0	12	7	29	11.19	223.86	forward	arsenal
3	saido berahino	13.8	21	9	0	20	7.02	147.43	forward	west brom
4	eden hazard	18.9	21	8	4	17	13.05	274.04	midfield	chelsea
5	yaya touré	16.6	18	7	1	19	10.99	197.91	midfield	manchester city

In [17]:

```
# Creating a dummy DataFrame with changes in the `salary` column


df_2 = df.copy()
df_2.loc[0:2, 'salary'] = [20.0, 15.0]
df_2.head(3)
```

Out[17]:

	player	salary	games	goals	assists	shots_on_target	points_per_game	points	position	team
1	sergio agüero	20	16	14	3	34	13.12	209.98	forward	manchester city
2	alexis sánchez	15	0	12	7	29	11.19	223.86	forward	arsenal
3	saido berahino	13.8	21	9	0	20	7.02	147.43	forward	west brom

In [18]:

```
# Temporarily use the `player` columns as indices to
# apply the update functions
```

```
df.set_index('player', inplace=True)
df_2.set_index('player', inplace=True)
df.head(3)
```

Out[18]:

	salary	games	goals	assists	shots_on_target	points_per_game	points	position	team
player									
sergio agüero	19.2	16	14	3	34	13.12	209.98	forward	manchester city
alexis sánchez	17.6	0	12	7	29	11.19	223.86	forward	arsenal
saido berahino	13.8	21	9	0	20	7.02	147.43	forward	west brom

In [19]:

```
# Update the `salary` column
df.update(other=df_2['salary'], overwrite=True)
df.head(3)
```

Out[19]:

	salary	games	goals	assists	shots_on_target	points_per_game	points	position	team
player									
sergio agüero	20	16	14	3	34	13.12	209.98	forward	manchester city
alexis sánchez	15	0	12	7	29	11.19	223.86	forward	arsenal
saido berahino	13.8	21	9	0	20	7.02	147.43	forward	west brom

In [20]:

```
# Reset the indices
df.reset_index(inplace=True)
df.head(3)
```

Out[20]:

	player	salary	games	goals	assists	shots_on_target	points_per_game	points	position	team
0	sergio agüero	20	16	14	3	34	13.12	209.98	forward	manchester city
1	alexis sánchez	15	0	12	7	29	11.19	223.86	forward	arsenal
2	saido berahino	13.8	21	9	0	20	7.02	147.43	forward	west brom

In [21]:

```
# Selecting only those players that either playing for Arsenal or Chelsea

df[ (df['team'] == 'arsenal') | (df['team'] == 'chelsea') ]
```

Out[21]:

	player	salary	games	goals	assists	shots_on_target	points_per_game	points	position	team
1	alexis sánchez	15	0	12	7	29	11.19	223.86	forward	arsenal
3	eden hazard	18.9	21	8	4	17	13.05	274.04	midfield	chelsea
7	santiago cazorla	14.8	20	4	0	20	9.97	0.00	midfield	arsenal
9	cesc fàbregas	14.0	20	2	14	10	10.47	209.49	midfield	chelsea

In [22]:

```
# Selecting forwards from Arsenal only

df[ (df['team'] == 'arsenal') & (df['position'] == 'forward') ]
```

Out[22]:

	player	salary	games	goals	assists	shots_on_target	points_per_game	points	position	team
1	alexis sánchez	15	0	12	7	29	11.19	223.86	forward	arsenal

In [23]:

```
types = df.columns.to_series().groupby(df.dtypes).groups
```

```
types
```

Out[23]:

```
{dtype('float64'): ['games',
  'goals',
  'assists',
  'shots_on_target',
  'points_per_game',
  'points'],
 dtype('O'): ['player', 'salary', 'position', 'team']}
```

Selecting by Column Type¶

In [24]:

```
# select string columns
df.loc[:, (df.dtypes == np.dtype('O')).values].head()
```

Out[24]:

	player	salary	position	team
0	sergio agüero	20	forward	manchester city
1	alexis sánchez	15	forward	arsenal
2	saido berahino	13.8	forward	west brom
3	eden hazard	18.9	midfield	chelsea
4	yaya touré	16.6	midfield	manchester city

In [25]:

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

In [26]:

```
types = df.columns.to_series().groupby(df.dtypes).groups
types
```

Out[26]:

```
{dtype('float64'): ['salary',
  'games',
  'goals',
  'assists',
  'shots_on_target',
  'points_per_game',
  'points'],
 dtype('O'): ['player', 'position', 'team']}
```

I was recently asked how to do an if-test in pandas, that is, how to create an array of 1s and 0s depending on a condition, e.g., if `val` less than 0.5 -> 0, else -> 1. Using the boolean mask, that's pretty simple since `True` and `False` are integers after all.

In [2]:

```
import pandas as pd


a = [[2., .3, 4., 5.], [.8, .03, 0.02, 5.]]
df = pd.DataFrame(a)
df
```

Out[2]:

	0	1	2	3
0	2.0	0.30	4.00	5
1	0.8	0.03	0.02	5

Out[3]:

	0	**1**	**2**	**3**
0	False	False	False	False
1	False	True	True	False

Out[4]:

	0	**1**	**2**	**3**
0	0	0	0	0
1	0	1	1	0