Basic Pandas workshop

In [39]:

```
!pip install pandas-datareader
```

```
Requirement already satisfied: pandas-datareader in e:\soft\anaconda3\lib\site-packages (0.9.
0)
Requirement already satisfied: lxml in e:\soft\anaconda3\lib\site-packages (from pandas-datar
eader) (4.3.2)
Requirement already satisfied: requests>=2.19.0 in e:\soft\anaconda3\lib\site-packages (from
pandas-datareader) (2.21.0)
Requirement already satisfied: pandas>=0.23 in e:\soft\anaconda3\lib\site-packages (from pand
as-datareader) (0.24.2)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in e:\soft\anaconda3\lib\site-packages
(from requests>=2.19.0->pandas-datareader) (3.0.4)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in e:\soft\anaconda3\lib\site-packages
(from requests>=2.19.0->pandas-datareader) (1.24.1)
Requirement already satisfied: certifi>=2017.4.17 in e:\soft\anaconda3\lib\site-packages (fro
m requests>=2.19.0->pandas-datareader) (2019.3.9)
Requirement already satisfied: idna<2.9,>=2.5 in e:\soft\anaconda3\lib\site-packages (from re
quests>=2.19.0->pandas-datareader) (2.8)
Requirement already satisfied: numpy>=1.12.0 in e:\soft\anaconda3\lib\site-packages (from pan
das>=0.23->pandas-datareader) (1.16.2)
Requirement already satisfied: pytz>=2011k in e:\soft\anaconda3\lib\site-packages (from panda
s>=0.23->pandas-datareader) (2018.9)
Requirement already satisfied: python-dateutil>=2.5.0 in e:\soft\anaconda3\lib\site-packages
(from pandas>=0.23->pandas-datareader) (2.8.0)
Requirement already satisfied: six>=1.5 in e:\soft\anaconda3\lib\site-packages (from python-d
ateutil>=2.5.0->pandas>=0.23->pandas-datareader) (1.12.0)
```

In [2]:

```
# import essential library
import pandas as pd
import numpy as np
import datetime
import os
import pandas_datareader.data as web
import matplotlib.pyplot as plt
from matplotlib import style
style.use('ggplot')
```

In [3]:

```
start = datetime.datetime(2010, 1,1)
end = datetime.datetime(2017, 1, 5)

df = web.DataReader('aapl', 'yahoo', start, end)
print (df.head())
print ("--------------------------------------")
#print (df.describe())
```

```
                High       Low      Open     Close       Volume  Adj Close
Date
2009-12-31  7.619643  7.520000  7.611786  7.526071  352410800.0   6.482683
2010-01-04  7.660714  7.585000  7.622500  7.643214  493729600.0   6.583586
2010-01-05  7.699643  7.616071  7.664286  7.656429  601904800.0   6.594968
2010-01-06  7.686786  7.526786  7.656429  7.534643  552160000.0   6.490066
2010-01-07  7.571429  7.466071  7.562500  7.520714  477131200.0   6.478067
--------------------------------------
```

In [4]:

```
df # dataframe
```

Out[4]:

	High	Low	Open	Close	Volume	Adj Close
Date						
2009-12-31	7.619643	7.520000	7.611786	7.526071	3.524108e+08	6.482683
2010-01-04	7.660714	7.585000	7.622500	7.643214	4.937296e+08	6.583586
2010-01-05	7.699643	7.616071	7.664286	7.656429	6.019048e+08	6.594968
2010-01-06	7.686786	7.526786	7.656429	7.534643	5.521600e+08	6.490066
2010-01-07	7.571429	7.466071	7.562500	7.520714	4.771312e+08	6.478067
2010-01-08	7.571429	7.466429	7.510714	7.570714	4.476108e+08	6.521136
2010-01-11	7.607143	7.444643	7.600000	7.503929	4.622296e+08	6.463610
2010-01-12	7.491786	7.372143	7.471071	7.418571	5.944596e+08	6.390087
2010-01-13	7.533214	7.289286	7.423929	7.523214	6.058920e+08	6.480221
2010-01-14	7.516429	7.465000	7.503929	7.479643	4.328940e+08	6.442690
2010-01-15	7.557143	7.352500	7.533214	7.354643	5.940676e+08	6.335020
2010-01-19	7.685357	7.401429	7.440357	7.680000	7.300076e+08	6.615271
2010-01-20	7.698214	7.482143	7.675357	7.561786	6.121528e+08	6.513446
2010-01-21	7.618214	7.400357	7.574286	7.431071	6.081544e+08	6.400853
2010-01-22	7.410714	7.041429	7.385000	7.062500	8.817676e+08	6.083380
2010-01-25	7.310714	7.149643	7.232500	7.252500	1.065700e+09	6.247038
2010-01-26	7.632500	7.235000	7.355357	7.355000	1.867110e+09	6.335329
2010-01-27	7.520714	7.126071	7.387500	7.424286	1.722568e+09	6.395009
2010-01-28	7.339286	7.096429	7.318929	7.117500	1.173502e+09	6.130756
2010-01-29	7.221429	6.794643	7.181429	6.859286	1.245952e+09	5.908337
2010-02-01	7.000000	6.832143	6.870357	6.954643	7.498764e+08	5.990476
2010-02-02	7.011429	6.906429	6.996786	6.995000	6.983424e+08	6.025239
2010-02-03	7.150000	6.943571	6.970357	7.115357	6.153280e+08	6.128909
2010-02-04	7.084643	6.841786	7.026071	6.858929	7.576520e+08	5.908031
2010-02-05	7.000000	6.816071	6.879643	6.980714	8.503068e+08	6.012933
2010-02-08	7.067143	6.928571	6.988929	6.932857	4.782708e+08	5.971709
2010-02-09	7.053571	6.955357	7.015000	7.006786	6.328868e+08	6.035388
2010-02-10	7.021429	6.937857	6.996071	6.968571	3.703616e+08	6.002471
2010-02-11	7.133929	6.930714	6.960000	7.095357	5.503456e+08	6.111681
2010-02-12	7.201429	6.982143	7.075357	7.156429	6.554688e+08	6.164287
...	...	...	...	...	...	...
2016-11-22	28.105000	27.850000	27.987499	27.950001	1.038620e+08	26.431519
2016-11-23	27.877501	27.582500	27.840000	27.807501	1.097056e+08	26.296761
2016-11-25	27.967501	27.737499	27.782499	27.947500	4.590360e+07	26.429152
2016-11-28	28.117500	27.847500	27.857500	27.892500	1.087760e+08	26.377140
2016-11-29	28.007500	27.517500	27.695000	27.865000	1.141152e+08	26.351133
2016-11-30	28.049999	27.567499	27.900000	27.629999	1.446492e+08	26.128902
2016-12-01	27.735001	27.257500	27.592501	27.372499	1.483476e+08	25.885395
2016-12-02	27.522499	27.212500	27.292500	27.475000	1.061120e+08	25.982323
2016-12-05	27.507500	27.062500	27.500000	27.277500	1.372980e+08	25.795553
2016-12-06	27.590000	27.297501	27.375000	27.487499	1.047820e+08	25.994146
2016-12-07	27.797501	27.290001	27.315001	27.757500	1.199948e+08	26.249474
2016-12-08	28.107500	27.650000	27.715000	28.030001	1.082732e+08	26.507175
2016-12-09	28.674999	28.077499	28.077499	28.487499	1.376104e+08	26.939817
2016-12-12	28.750000	28.122499	28.322500	28.325001	1.054976e+08	26.786144
2016-12-13	28.980000	28.437500	28.459999	28.797501	1.749352e+08	27.232977
2016-12-14	29.049999	28.745001	28.760000	28.797501	1.361272e+08	27.232977
2016-12-15	29.182501	28.807501	28.844999	28.955000	1.860980e+08	27.381918
2016-12-16	29.125000	28.912500	29.117500	28.992500	1.774044e+08	27.417383
2016-12-19	29.344999	28.937500	28.950001	29.160000	1.111176e+08	27.575779
2016-12-20	29.375000	29.170000	29.184999	29.237499	8.570000e+07	27.649071
2016-12-21	29.350000	29.195000	29.200001	29.264999	9.513280e+07	27.675074
2016-12-22	29.127501	28.910000	29.087500	29.072500	1.043436e+08	27.493036
2016-12-23	29.129999	28.897499	28.897499	29.129999	5.699800e+07	27.547411
2016-12-27	29.450001	29.122499	29.129999	29.315001	7.318760e+07	27.722363
2016-12-28	29.504999	29.049999	29.379999	29.190001	8.362360e+07	27.604153
2016-12-29	29.277500	29.100000	29.112499	29.182501	6.015800e+07	27.597061
2016-12-30	29.299999	28.857500	29.162500	28.955000	1.223452e+08	27.381918
2017-01-03	29.082500	28.690001	28.950001	29.037500	1.151276e+08	27.459938
2017-01-04	29.127501	28.937500	28.962500	29.004999	8.447240e+07	27.429203
2017-01-05	29.215000	28.952499	28.980000	29.152500	8.877440e+07	27.568691

1766 rows × 6 columns

In [5]:

```
df.index.values
```

Out[5]:

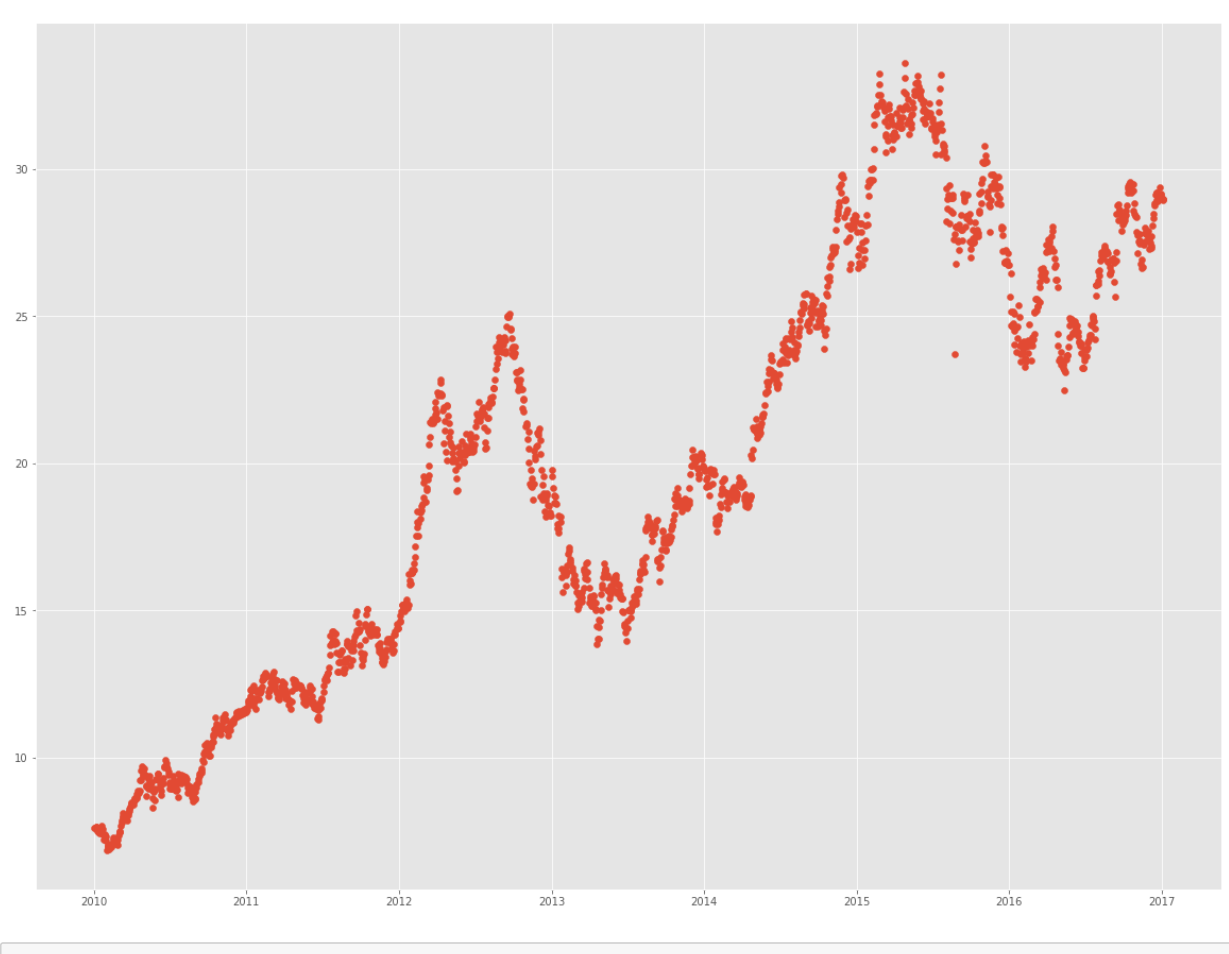
```
array(['2009-12-31T00:00:00.000000000', '2010-01-04T00:00:00.000000000',
       '2010-01-05T00:00:00.000000000', ...,
       '2017-01-03T00:00:00.000000000', '2017-01-04T00:00:00.000000000',
       '2017-01-05T00:00:00.000000000'], dtype='datetime64[ns]')
```

In [10]:

```
#plt.figure(figsize=(20,15))
plt.scatter(df.index.values, df["Open"])
```

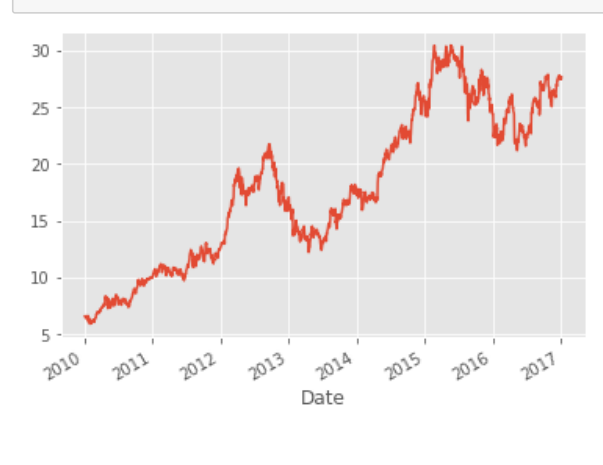
Out[10]:

```
<matplotlib.collections.PathCollection at 0x2c16b530f98>
```



In [4]:

```
df["Adj Close"].plot()
plt.show()
```



1 Create Data Frame

- to create pandas data frame from dictionary in python
- to create pandas data frame from csv

In [11]:

```
# pandas data frame from dictionary in python
data = { "name" : ['a','b','c','d','e','f'],
         "id" : [1,3,8,10,12,60],
         "value" : [23,21,18,29,43,21]}

df = pd.DataFrame(data)

print (type(df))
print (df)

df = df.set_index("id")
print (df)
```

```
<class 'pandas.core.frame.DataFrame'>
  name  id  value
0    a   1     23
1    b   3     21
2    c   8     18
3    d  10     29
4    e  12     43
5    f  60     21
   name  value
id
1     a     23
3     b     21
8     c     18
10    d     29
12    e     43
60    f     21
```

In [6]:

```
print (df['value'])
print (df.name)
```

```
id
1     23
3     21
8     18
10    29
12    43
60    21
Name: value, dtype: int64
id
1     a
3     b
8     c
10    d
12    e
60    f
Name: name, dtype: object
```

In [12]:

```
# your codes here

df
```

Out[12]:

	name	value
id		
1	a	23
3	b	21
8	c	18
10	d	29
12	e	43
60	f	21

In [13]:

```
df["value"].values
```

Out[13]:

```
array([23, 21, 18, 29, 43, 21], dtype=int64)
```

In [8]:

```
# pandas data frames from list of list
listOfList = [[1,2,3],
              [4,5,6],
              [7,8,9]]

dfFromList = pd.DataFrame(listOfList)
print (dfFromList)


lstA = [1,2,3]
lstB = [4,5,6]
lstC = [7,8,9]
listOfList2 = []
listOfList2.append(lstA)
listOfList2.append(lstB)
listOfList2.append(lstC)

dfFromList = pd.DataFrame(listOfList2)
dfFromList.columns = ["ID", "valA", 'valB']
print (dfFromList)
```

```
   0  1  2
0  1  2  3
1  4  5  6
2  7  8  9
   ID  valA  valB
0   1     2     3
1   4     5     6
2   7     8     9
```

In [14]:

```
# your codes here

myDF = pd.read_csv("BostonWeatherFromWeb.csv")

myDF
```

Out[14]:

	Unnamed: 0	Month	Average high in :	Average low in :	Av. precipitation in :	Days with precipitation:	Hours of sunshine:	Average snowfall in :
0	0	Jan	36	22	3.35	13	148	13
1	1	Feb	39	25	3.27	11	168	11
2	2	Mar	45	31	4.33	13	212	8
3	3	Apr	56	41	3.74	12	222	2
4	4	May	66	50	3.50	13	263	0
5	5	Jun	76	60	3.66	12	283	0
6	6	Jul	81	65	3.43	11	300	0
7	7	Aug	80	65	3.35	10	280	0
8	8	Sep	72	57	3.43	10	232	0
9	9	Oct	61	47	3.94	10	207	0
10	10	Nov	51	38	3.98	11	152	1
11	11	Dec	41	28	3.78	11	148	9

In [23]:

```
# pandas data frame from csv
def getPandasFromFile(path, fileName, theSkipRow):
    path = os.path.join(path , fileName)
    return pd.read_csv(path, skiprows= theSkipRow , header=None, index_col = False)
def SavePandasToCSV(d, path):
    d.to_csv(path)
    return "done!!"

df = pd.read_csv("BostonWeatherFromWeb.csv")
df
```

Out[23]:

	Unnamed: 0	Month	Average high in :	Average low in :	Av. precipitation in :	Days with precipitation:	Hours of sunshine:	Average snowfall in :
0	0	Jan	36	22	3.35	13	148	13
1	1	Feb	39	25	3.27	11	168	11
2	2	Mar	45	31	4.33	13	212	8
3	3	Apr	56	41	3.74	12	222	2
4	4	May	66	50	3.50	13	263	0
5	5	Jun	76	60	3.66	12	283	0
6	6	Jul	81	65	3.43	11	300	0
7	7	Aug	80	65	3.35	10	280	0
8	8	Sep	72	57	3.43	10	232	0
9	9	Oct	61	47	3.94	10	207	0
10	10	Nov	51	38	3.98	11	152	1
11	11	Dec	41	28	3.78	11	148	9

2. index and columns, and converting data

- convert data frame to list
- convert data frame to numpy
- convert numpy or list to data frame
- manipulation columns of data frame
- manipulation row of data frame

In [16]:

```
mylist = myDF["Av. precipitation in :"].tolist()
mylist
```

Out[16]:

```
[3.35, 3.27, 4.33, 3.74, 3.5, 3.66, 3.43, 3.35, 3.43, 3.94, 3.98, 3.78]
```

In [19]:

```
for i in range(0,12):
    if mylist[i] > 3.5:
        print (mylist[i])
```

```
4.33
3.74
3.66
3.94
3.98
3.78
```

In [21]:

```
print ("the index :", df.index)
print ("---------------------------------------------------")
print ("the index :", df.index.values)
print ("---------------------------------------------------")
print ('the columns', df.columns.values)
print ("---------------------------------------------------")
#print (df['Average high in :'])
#print (df[['Month','Days with precipitation:']])
```

```
the index : Int64Index([1, 3, 8, 10, 12, 60], dtype='int64', name='id')
---------------------------------------------------
the index : [ 1  3  8 10 12 60]
---------------------------------------------------
the columns ['name' 'value']
---------------------------------------------------
```

In [12]:

```
# your codes here
```

In [13]:

```
# to List
airList = df['Hours of sunshine:'].tolist()
print (type(airList))
print (len(airList))
print ("------------------")
print (airList)
```

```
<class 'list'>
12
------------------
[148, 168, 212, 222, 263, 283, 300, 280, 232, 207, 152, 148]
```

In [14]:

```
# to Numpy array
npArray = np.array(df['Hours of sunshine:'])
print (type(npArray))
print (npArray)
```

```
<class 'numpy.ndarray'>
[148 168 212 222 263 283 300 280 232 207 152 148]
```

In [43]:

```
# to Pandas Data Frame
temp = df[['Month', 'Hours of sunshine:', 'Average snowfall in :']]

newNp = np.array(temp)
print (newNp)

newDF = pd.DataFrame(temp)
print (newDF)
```

```
[['Jan' 148 13]
 ['Feb' 168 11]
 ['Mar' 212 8]
 ['Apr' 222 2]
 ['May' 263 0]
 ['Jun' 283 0]
 ['Jul' 300 0]
 ['Aug' 280 0]
 ['Sep' 232 0]
 ['Oct' 207 0]
 ['Nov' 152 1]
 ['Dec' 148 9]]
   Month  Hours of sunshine:  Average snowfall in :
0    Jan                 148                     13
1    Feb                 168                     11
2    Mar                 212                      8
3    Apr                 222                      2
4    May                 263                      0
5    Jun                 283                      0
6    Jul                 300                      0
7    Aug                 280                      0
8    Sep                 232                      0
9    Oct                 207                      0
10   Nov                 152                      1
11   Dec                 148                      9
```

In [44]:

```
# your codes here
```

In [45]:

```
# rename columns
print (newDF.head(3))

newDF.columns = ["Time", "Sun", 'Snow']

print (newDF.head(3))
```

```
  Month  Hours of sunshine:  Average snowfall in :
0   Jan                 148                     13
1   Feb                 168                     11
2   Mar                 212                      8
  Time  Sun  Snow
0  Jan  148    13
1  Feb  168    11
2  Mar  212     8
```

In [46]:

```
# get data from selected columns
dataCol = newDF['Sun'].values
print (dataCol)

dataCol = newDF['Sun']
print (dataCol)
```

```
[148 168 212 222 263 283 300 280 232 207 152 148]
0     148
1     168
2     212
3     222
4     263
5     283
6     300
7     280
8     232
9     207
10    152
11    148
Name: Sun, dtype: int64
```

In [47]:

```
# your codes here
```

In [48]:

```
# slice data between start and end index
dataCol = newDF['Sun']

print (dataCol[2:10])
```

```
2    212
3    222
4    263
5    283
6    300
7    280
8    232
9    207
Name: Sun, dtype: int64
```

In [55]:

```
# your codes here
```

In [56]:

```
print (df.describe())
print ("--------------------------------")
```

```
       Unnamed: 0  Average high in :  Average low in :  \
count   12.000000          12.000000         12.000000
mean     5.500000          58.666667         44.083333
std      3.605551          16.394752         15.570709
min      0.000000          36.000000         22.000000
25%      2.750000          44.000000         30.250000
50%      5.500000          58.500000         44.000000
75%      8.250000          73.000000         57.750000
max     11.000000          81.000000         65.000000

       Av. precipitation in :  Days with precipitation:  Hours of sunshine:  \
count               12.000000                 12.000000           12.000000
mean                 3.646667                 11.416667          217.916667
std                  0.319668                  1.164500           55.467367
min                  3.270000                 10.000000          148.000000
25%                  3.410000                 10.750000          164.000000
50%                  3.580000                 11.000000          217.000000
75%                  3.820000                 12.250000          267.250000
max                  4.330000                 13.000000          300.000000

       Average snowfall in :
count              12.000000
mean                3.666667
std                 5.033223
min                 0.000000
25%                 0.000000
50%                 0.500000
75%                 8.250000
max                13.000000
--------------------------------
```

In [57]:

```
print ("the sum of value column :", df['Average high in :'].sum())
print ("the standard deviation of value column :", df['Average high in :'].std())
print ("the mean of value column :", df['Average high in :'].mean())
print ("the min of value column :", df['Average high in :'].min())
print ("the max of value column :", df['Average high in :'].max())
```

```
the sum of value column : 704
the standard deviation of value column : 16.39475156224939
the mean of value column : 58.666666666666664
the min of value column : 36
the max of value column : 81
```

In [58]:

```
# sort
sortedData = df.sort_values(['Hours of sunshine:'], ascending=[True])
sortedData
```

Out[58]:

	Unnamed: 0	Month	Average high in :	Average low in :	Av. precipitation in :	Days with precipitation:	Hours of sunshine:	Average snowfall in :
0	0	Jan	36	22	3.35	13	148	13
11	11	Dec	41	28	3.78	11	148	9
10	10	Nov	51	38	3.98	11	152	1
1	1	Feb	39	25	3.27	11	168	11
9	9	Oct	61	47	3.94	10	207	0
2	2	Mar	45	31	4.33	13	212	8
3	3	Apr	56	41	3.74	12	222	2
8	8	Sep	72	57	3.43	10	232	0
4	4	May	66	50	3.50	13	263	0
7	7	Aug	80	65	3.35	10	280	0
5	5	Jun	76	60	3.66	12	283	0
6	6	Jul	81	65	3.43	11	300	0

In [59]:

```
# conditional statement
newDF
```

Out[59]:

	Time	Sun	Snow
0	Jan	148	13
1	Feb	168	11
2	Mar	212	8
3	Apr	222	2
4	May	263	0
5	Jun	283	0
6	Jul	300	0
7	Aug	280	0
8	Sep	232	0
9	Oct	0	0
10	Nov	152	1
11	Dec	148	9

In [60]:

```
print ("---------------------")
print (newDF['Sun'] == 0)
print ("---------------------")
print (newDF[newDF['Sun'] > 200])
print ("---------------------")
print (newDF[(newDF["Sun"] > 100) & (newDF['Sun'] < 200) ])
print ("---------------------")
print (newDF[ (newDF["Sun"] > 100) & (newDF['Snow'] > 1) ])
```

```
---------------------
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9      True
10    False
11    False
Name: Sun, dtype: bool
---------------------
  Time  Sun  Snow
2  Mar  212     8
3  Apr  222     2
4  May  263     0
5  Jun  283     0
6  Jul  300     0
7  Aug  280     0
8  Sep  232     0
---------------------
   Time  Sun  Snow
0   Jan  148    13
1   Feb  168    11
10  Nov  152     1
11  Dec  148     9
---------------------
   Time  Sun  Snow
0   Jan  148    13
1   Feb  168    11
2   Mar  212     8
3   Apr  222     2
11  Dec  148     9
```

3. save data

- save data as csv
- save data as HTML

In [62]:

```
# save data frame as csv
newDF.to_csv("new_data.csv", header=True)
newDF.to_html("new_data.html", header=True)
```

Read the weather file

In [24]:

```
d = getPandasFromFile("",'USA_MA_Boston-Logan.Intl.AP.725090_TMY3.epw', 8)
d.head(3)
```

Out[24]:

	0	1	2	3	4	5	6	7	8	9	...	25	26	27	28	29	30	31	32	33	34
0	1976	1	1	1	0	?9?9?9?9E0?9? 9?9?9?9?9?9? 9?9?9?9?9?9? 9?9*9*9*9...	1.7	-3.9	67	100900	...	3960	9	999999999	89	0.089	0	88	999.0	999.0	99.0
1	1976	1	1	2	0	?9?9?9?9E0?9? 9?9?9?9?9?9? 9?9?9?9?9?9? 9?9*9*9*9...	1.7	-3.9	67	100800	...	3960	9	999999999	89	0.089	0	88	999.0	999.0	99.0
2	1976	1	1	3	0	?9?9?9?9E0?9? 9?9?9?9?9?9? 9?9?9?9?9?9? 9?9*9*9*9...	1.1	-3.9	70	100800	...	3960	9	999999999	89	0.089	0	88	999.0	999.0	99.0

3 rows × 35 columns

In [25]:

```
# Rename the column

theDf =d.rename(index=str, columns={0:"year", 1: "Month", 2: "Day", 3:"Hour",4:"Minute",6:"D
B_temp",
                                    7:"Dew_Point",
                                    8:"RH",9:"P", 10:"Horiz_Rad",11:"Normal_Rad",12:"Sky_Ra
d",
                                    13:"G_Horiz_Rad",14:"Dir_Normal_Rad",15:"Diff_Horiz_Ra
d",
                                    16:"G_Horiz_Illu",17:"Dir_Normal_Illu",18:"Diff_Horiz_I
llu",
                                    19:"Zenith_Illu",20:"Wind_Direction",21:"Wind_Speed",
                                    22:"Total_Sky_Cov",23:"Opaque_Sky_Cov",24:"Visibility",
                                    25:"field_Ceiling_H",26:"Whtr_Observ",27:"Whtr_Codes",
                                    28:"Pred_Water",29:"AeroesOptical_D",30:"Snow_Depth",
                                    31:"Days_Since_Snow"
                                   })
```

In [67]:

```
theDf.head(3)
```

Out[67]:

	year	Month	Day	Hour	Minute	5	DB_temp	Dew_Point	RH	P	...	field_Ceiling_H	Whtr_Observ	Whtr_Cod
0	1976	1	1	1	0	?9?9?9? 9E0?9? 9?9?9? 9?9?9? 9?9?9? 9?9?9? 9? 9*9*9*9...	1.7	-3.9	67	100900	...	3960	9	9999999
1	1976	1	1	2	0	?9?9?9? 9E0?9? 9?9?9? 9?9?9? 9?9?9? 9?9?9? 9? 9*9*9*9...	1.7	-3.9	67	100800	...	3960	9	9999999
2	1976	1	1	3	0	?9?9?9? 9E0?9? 9?9?9? 9?9?9? 9?9?9? 9?9?9? 9? 9*9*9*9...	1.1	-3.9	70	100800	...	3960	9	9999999

3 rows × 35 columns

In [68]:

```
theDf.columns.values
```

Out[68]:

```
array(['year', 'Month', 'Day', 'Hour', 'Minute', 5, 'DB_temp',
       'Dew_Point', 'RH', 'P', 'Horiz_Rad', 'Normal_Rad', 'Sky_Rad',
       'G_Horiz_Rad', 'Dir_Normal_Rad', 'Diff_Horiz_Rad', 'G_Horiz_Illu',
       'Dir_Normal_Illu', 'Diff_Horiz_Illu', 'Zenith_Illu',
       'Wind_Direction', 'Wind_Speed', 'Total_Sky_Cov', 'Opaque_Sky_Cov',
       'Visibility', 'field_Ceiling_H', 'Whtr_Observ', 'Whtr_Codes',
       'Pred_Water', 'AeroesOptical_D', 'Snow_Depth', 'Days_Since_Snow',
       32, 33, 34], dtype=object)
```

In [27]:

```
# select the data that needed

NewDf = theDf[['DB_temp',"Dew_Point",
               'RH','P','Normal_Rad', 'Sky_Rad','Wind_Direction', 'Wind_Speed'
             ]]
NewDf.head()
```

Out[27]:

	DB_temp	Dew_Point	RH	P	Normal_Rad	Sky_Rad	Wind_Direction	Wind_Speed
0	1.7	-3.9	67	100900	0	290	360	6.2
1	1.7	-3.9	67	100800	0	290	350	5.7
2	1.1	-3.9	70	100800	0	287	360	6.2
3	1.1	-3.9	70	100700	0	287	10	5.7
4	1.1	-4.4	67	100700	0	287	10	5.7

In [28]:

```
NewDf.describe()
```

Out[28]:

	DB_temp	Dew_Point	RH	P	Normal_Rad	Sky_Rad	Wind_Direction	Wind_Speed
count	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000	8760.000000
mean	10.599578	3.830559	65.676142	101468.824201	689.765982	318.564498	219.007991	5.484703
std	9.918710	10.730105	19.749037	929.330511	661.566938	57.612886	96.724574	2.267986
min	-20.000000	-32.200000	19.000000	98200.000000	0.000000	161.000000	0.000000	0.000000
25%	2.800000	-3.900000	50.000000	100900.000000	0.000000	277.000000	150.000000	4.100000
50%	10.600000	4.400000	65.000000	101500.000000	799.500000	317.000000	240.000000	5.200000
75%	18.300000	12.800000	83.000000	102100.000000	1351.000000	362.000000	300.000000	6.700000
max	37.200000	23.300000	100.000000	104100.000000	1415.000000	470.000000	360.000000	21.600000

In [29]:

```
SavePandasToCSV(NewDf, "myBoston.csv")
```

Out[29]:

```
'done!!'
```

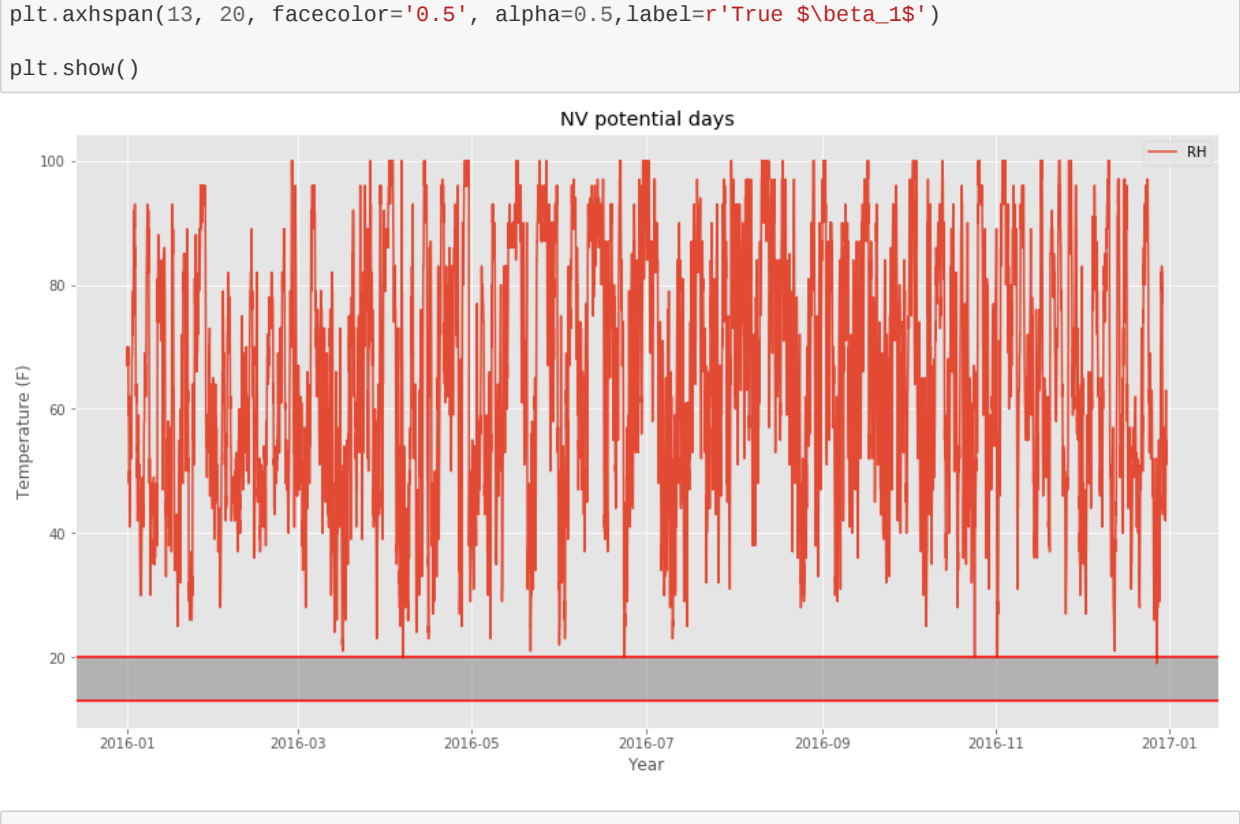
Visualize weather file

In [30]:

```
import matplotlib
import matplotlib.pyplot as plt
```

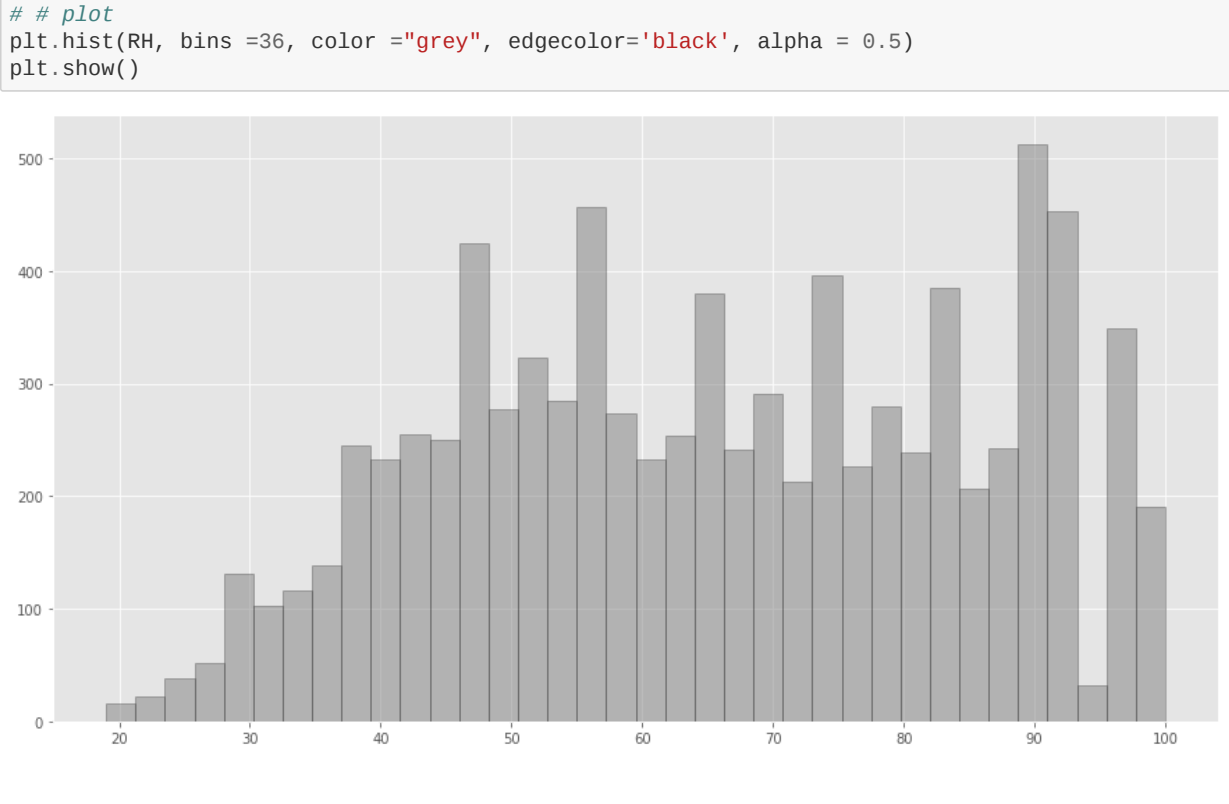
In [33]:

```
x = NewDf.index
temp = NewDf["DB_temp"]
#rh = NewDf["RH"]..................................relative humidity


plt.figure(figsize=(15,8))
myTime = pd.date_range('1/1/2016', periods=8760, freq='H')

# plt.ylim((20,100) )
plt.plot(myTime,rh) # you could change temp to rh.........................................
plt.title('NV potential days')
plt.xlabel('Year')
plt.ylabel('Temperature (F)')
plt.legend(loc='best')

plt.axhline(13, color='r', label=r'True $\beta_1$')
plt.axhline(20, color='r', label=r'True $\beta_1$')
plt.axhspan(13, 20, facecolor='0.5', alpha=0.5,label=r'True $\beta_1$')

plt.show()
```



In [34]:

```
x = NewDf.index
RH = NewDf["RH"]

plt.figure(figsize=(15,8))
# # plot
plt.hist(RH, bins =36, color ="grey", edgecolor='black', alpha = 0.5)
plt.show()
```



epw Manipulation

In [35]:

```
# some modification before you run the simulation, i.e. for future scenarios, or to specify
 the condition



d = getPandasFromFile("",'USA_MA_Boston-Logan.Intl.AP.725090_TMY3.epw', 8)
d.head(3)
```

Out[35]:

	0	1	2	3	4	5	6	7	8	9	...	25	26	27	28	29	30	31	32	33	34
0	1976	1	1	1	0	?9?9?9?9E0?9? 9?9?9?9?9?9? 9?9?9?9?9?9? 9?9*9*9*9...	1.7	-3.9	67	100900	...	3960	9	999999999	89	0.089	0	88	999.0	999.0	99.0
1	1976	1	1	2	0	?9?9?9?9E0?9? 9?9?9?9?9?9? 9?9?9?9?9?9? 9?9*9*9*9...	1.7	-3.9	67	100800	...	3960	9	999999999	89	0.089	0	88	999.0	999.0	99.0
2	1976	1	1	3	0	?9?9?9?9E0?9? 9?9?9?9?9?9? 9?9?9?9?9?9? 9?9*9*9*9...	1.1	-3.9	70	100800	...	3960	9	999999999	89	0.089	0	88	999.0	999.0	99.0

3 rows × 35 columns

In [36]:

```
# extract wind speed

df_wind = d.iloc[:,21]
print (df_wind)
```

```
0       6.2
1       5.7
2       6.2
3       5.7
4       5.7
5       6.7
6       7.7
7       8.2
8       6.2
9       7.7
10      6.2
11      5.2
12      5.7
13      8.8
14      7.7
15      6.7
16      6.7
17      7.7
18      8.2
19      7.7
20      8.2
21      8.8
22      8.2
23      6.7
24      6.2
25      6.7
26      7.2
27      3.1
28      5.7
29      6.2
       ...
8730    9.3
8731    8.8
8732    8.2
8733    6.7
8734    8.8
8735    7.2
8736    7.2
8737    8.8
8738    9.8
8739    8.8
8740    6.7
8741    7.2
8742    6.7
8743    6.7
8744    7.2
8745    7.7
8746    6.2
8747    4.6
8748    6.7
8749    4.1
8750    5.2
8751    4.6
8752    4.1
8753    4.6
8754    3.6
8755    8.8
8756    7.7
8757    7.7
8758    7.2
8759    7.2
Name: 21, Length: 8760, dtype: float64
```

In [37]:

```
array = []
for i in range (0,len(df_wind)):
    array.append(float(0.1))

d.loc[:,21] = array

# print (array)
```

In [78]:

```
df_wind = d.iloc[:,21]
print (df_wind)
```

```
0       0.1
1       0.1
2       0.1
3       0.1
4       0.1
       ...
8755    0.1
8756    0.1
8757    0.1
8758    0.1
8759    0.1
Name: 21, Length: 8760, dtype: float64
```

In [38]:

```
d.to_csv("my.epw", index=False)
```

In [ ]:

```
```