SKILL<S/ASH>

PANDAS Cheatsheet

Part-1

Pandas Basic

Introduction

Data processing is important part of analyzing the data, because data is not always available in desired format. Various processing are required before analyzing the data such as cleaning, restructuring or merging etc. Numpy, Scipy, Cython and Panda are the tools available in python which can be used fast processing of the data. Further, Pandas are built on the top of Numpy.

Pandas provides rich set of functions to process various types of data. Further, working with Panda is fast, easy and more expressive than other tools. Pandas provides fast data processing as Numpy along with flexible data manipulation techniques as spreadsheets and relational databases. Lastly, pandas integrates well with matplotlib library, which makes it very handy tool for analyzing the data.

Data structures

Pandas provides two very useful data structures to process the data i.e. Series and DataFrame, which are discussed in this section.

Series

The Series is a one-dimensional array that can store various data types, including mix data types. The row labels in a Series are called the index. Any list, tuple and dictionary can be converted in to Series using ‘series’ method as shown below -

```
>>> import pandas as pd

>>> # converting tuple to Series
>>> h = ('AA', '2012-02-01', 100, 10.2)
>>> s = pd.Series(h)

>>> type(s)
<class 'pandas.core.series.Series'>

>>> print(s)
0              AA
1      2012-02-01
2             100
3            10.2
dtype: object

>>> # converting dict to Series
>>> d = {'name' : 'IBM', 'date' : '2010-09-08', 'shares' : 100, 'price' : 10.2}
>>> ds = pd.Series(d)

>>> type(ds)
<class 'pandas.core.series.Series'>

>>> print(ds)
date      2010-09-08
name             IBM
price           10.2
shares           100
dtype: object
```

Note that in the tuple-conversion, the index are set to '0, 1, 2 and 3'. We can provide custom index names as follows.

```
>>> f = ['FB', '2001-08-02', 90, 3.2]
>>> f = pd.Series(f, index = ['name', 'date', 'shares', 'price'])
>>> print(f)
name                FB
date        2001-08-02
shares              90
price              3.2
dtype: object

>>> f['shares']
90
>>> f[0]
'FB'
>>>
```

Elements of the Series can be accessed using index name e.g. f['shares'] or f[0] in below code. Further, specific elements can be selected by providing the index in the list,

```
>>> f[['shares', 'price']]
shares      90
price      3.2
dtype: object
```

DataFrame

DataFrame is the widely used data structure of pandas. Note that, Series are used to work with one dimensional array, whereas DataFrame can be used with two dimensional arrays. DataFrame has two different index i.e. column-index and row-index.

The most common way to create a DataFrame is by using the dictionary of equal-length list as shown below. Further, all the spreadsheets and text files are read as DataFrame, therefore it is very important data structure of pandas.

```
>>> data = { 'name' : ['AA', 'IBM', 'GOOG'],
...  'date' : ['2001-12-01', '2012-02-10', '2010-04-09'],
...  'shares' : [100, 30, 90],
...  'price' : [12.3, 10.3, 32.2] ...
}

>>> df = pd.DataFrame(data)
>>> type(df)
<class 'pandas.core.frame.DataFrame'>

>>> df
         date   name   price   shares
0   2001-12-01     AA    12.3      100
1   2012-02-10    IBM    10.3       30
2   2010-04-09   GOOG    32.2       90
```

Additional columns can be added after defining a DataFrame as below,

```
>>> df['owner'] = 'Unknown'
>>> df
        date    name    price    shares      owner
0   2001-12-01      AA     12.3       100    Unknown
1   2012-02-10     IBM     10.3        30    Unknown
2   2010-04-09    GOOG     32.2        90    Unknown
```

Currently, the row index are set to 0, 1 and 2. These can be changed using 'index' attribute as below,

```
>>> df.index = ['one', 'two', 'three']
>>> df
                date     name     price     shares      owner
one       2001-12-01       AA      12.3        100   Unknown
two       2012-02-10      IBM      10.3         30   Unknown
three     2010-04-09     GOOG      32.2         90   Unknown
```

Further, any column of the DataFrame can be set as index using 'set_index()' attribute, as shown below,

```
>>> df = df.set_index(['name'])
>>> df
              date    price    shares       owner
name
AA      2001-12-01     12.3       100     Unknown
IBM     2012-02-10     10.3        30     Unknown
GOOG    2010-04-09     32.2        90     Unknown
```

Data can be accessed in two ways i.e. using row and column index,

```
>>> # access data using column-index
>>> df['shares']
name
AA     100
IBM     30
GOOG    90
Name: shares, dtype: int64

>>> # access data by row-index
>>> df.ix['AA']
date    2001-12-01
price         12.3
shares         100
owner      Unknown
Name: AA, dtype: object
```

```
>>> # access all rows for a column
>>> df.ix[:, 'name']
0    AA
1   IBM
2  GOOG Name: name, dtype: object

>>> # access specific element from the DataFrame,
>>> df.ix[0, 'shares']
100
```

Any column can be deleted using 'del' or 'drop' commands,

```
>>> del df['owner']
>>> df
           date price shares name
AA 2001-12-01 12.3 100
IBM 2012-02-10 10.3 30
GOOG 2010-04-09 32.2 90

>>> df.drop('shares', axis = 1)
            date      price
name
AA      2001-12-01       12.3
IBM     2012-02-10       10.3
GOOG    2010-04-09       32.2
```

SKILL<S/ASH>

Drop a follow
for more such content :)

Upskill with **Skillslash**