Fresher Academy

Pandas

What Will You Learn

- Pandas Introduction
- Series and DataFrame
- Import and Export files
- DataFrame Exploratory
- Pandas with Datetime
- DataFrame ETL
- Data Aggregation

Pandas

FPT Software | FRESHER ACADEMY

Introduction

Pandas Introduction

Pandas - Panel Data

2008

Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data

Pandas Introduction

Key Features

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

Pandas Introduction

Environment

- Install the package

```
$ pip install pandas
```

- Load the package

```
>>> import pandas as pd
```

Pandas Introduction

Pandas Data Structures

- Series: 1D labeled
- DataFrame: General 2D labeled
- Panel: General 3D labeled

- Series: 1D labeled
- DataFrame: General 2D labeled
- Panel: General 3D labeled

Pandas

Series and DataFrames

FPT Software | FRESHER ACADEMY

Series and DataFrames

Create Series from a NumPy array

```
>>> A = data = np.array(['a','b','c','d'])
>>> s = pd.Series(data)
>>> s
0    a
1    b
2    c
3    d
```

Series and DataFrames

Create Series from a NumPy array with indexes

```
>>> data = np.array(['a','b','c','d'])
>>> s = pd.Series(data,index=[100,101,102,103])
>>> s
100    a
101    b
102    c
103    d
```

Series and DataFrames

Create a Series from a dictionary

```
>>> data = {'a' : 0., 'b' : 1., 'c' : 2.}
>>> s = pd.Series(data)
>>> s
a    0.0
b    1.0
c    2.0
```

Series and DataFrames

Create a Series from a dictionary with indexes

```
>>> data = {'a' : 0., 'b' : 1., 'c' : 2.}
>>> s = pd.Series(data,index=['b','c','d','a'])
>>> s
b    1.0
c    2.0
d    NaN
a    0.0
```

Series and DataFrames

Create DataFrame

- pandas.DataFrame(data, index, columns, dtype, copy)
- The data can be:
  - List
  - Dictionary
  - Series
  - Numpy ndarray
  - Another DataFrame

Series and DataFrames

Create a one-column DataFrame from a list

```
>>> data = [1,2,3,4,5]
>>> df = pd.DataFrame(data)
>>> df
   0
0  1
1  2
2  3
3  4
4  5
```

Series and DataFrames

Create a multiple-column DataFrame from a list

```
>>> data = [['Alex',10],['Bob',12],['Clarke',13]]
>>> df = pd.DataFrame(data,columns=['Name','Age'])
>>> df
      Name  Age
0     Alex   10
1      Bob   12
2  Clarke   13
```

Series and DataFrames

Create a DataFrame from a Dictionary of ndarrays/Lists

```
>>> data = {'Name':['Tom', 'Jack', 'Steve'],'Age':[28,34,29]}
>>> df = pd.DataFrame(data)
>>> df
    Name  Age
0    Tom   28
1   Jack   34
2  Steve   29
```

Series and DataFrames

Create a DataFrame from a list of dictionaries

```
>>> data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
>>> df = pd.DataFrame(data)
>>> df
   a   b     c
0  1   2   NaN
1  5  10  20.0
```

Series and DataFrames

Create a DataFrame from a list of dictionaries with indexes

```
>>> data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
>>> df = pd.DataFrame(data, index=['first', 'second'])
a   b      c
first   1   2   NaN
second  5  10  20.0
```

Series and DataFrames

Create a DataFrame from List of dictionaries with indexes and columns

```
>>> data = [{'a': 1, 'b': 2},{'a': 5, 'b': 10, 'c': 20}]
>>> df1 = pd.DataFrame(data, index=[1, 2], columns=['a', 'b'])
>>> df1
   a   b
1  1   2
2  5  10
>>> df2 = pd.DataFrame(data, index=[1, 2], columns=['a', 'b1'])
>>> df2
   a  b1
1  1 NaN
2  5 NaN
```

Pandas

Import and Export Files

FPT Software | FRESHER ACADEMY

Import and Export Files

Import by using read_csv()

```
>>> import pandas as pd
>>> scores = pd.read_csv('scores.csv')
>>> scores
   id Category Gender  Score
0   1        A      F    2.3
1   2        A      M    4.5
2   3        B      F    2.4
3   4        B      M    5.6
4   5        C      F    3.4
5   6        C      M    6.4
6   7        D      M    2.3
7   8        D      F    0.5
```

Import and Export Files

Import by using read_csv() with the index column

```
>>> scores = pd.read_csv('scores.csv', index_col='id')
>>> scores
   Category Gender  Score
id
1         A      F    2.3
2         A      M    4.5
3         B      F    2.4
4         B      M    5.6
5         C      F    3.4
6         C      M    6.4
7         D      M    2.3
8         D      F    0.5
```

Import and Export Files

Import by using read_csv() with data types

```
>>> scores = pd.read_csv('scores.csv', index_col = 'id', dtype =
{'Score':np.float32})
>>> scores.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8 entries, 1 to 8
Data columns (total 3 columns):
Category    8 non-null object
Gender      8 non-null object
Score       8 non-null float32
dtypes: float32(1), object(2)
memory usage: 224.0+ bytes
```

Import and Export Files

Import by using read_csv() and skip rows

```
>>> scores = pd.read_csv('scores.csv', skiprows=5, header=None,
names=['id', 'Category','Gender', 'Score'], index_col = 'id')
>>> scores
   Category Gender  Score
id
5         C      F    3.4
6         C      M    6.4
7         D      M    2.3
8         D      F    0.5
```

Import and Export Files

Export a DataFrame to files

```
>>> scores.to_csv('scores2.csv')
```

Pandas

DataFrame Exploratory

DataFrame Exploratory

DataFrame Attributes

```
>>> samples = pd.read_csv('samples.csv', index_col='month')
>>> samples.axes
[Index(['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug'],
dtype='object', name='month'), Index(['length_cm', 'time', 'class'],
dtype='object')]
>>> samples.empty
False
>>> samples.ndim
2
>>> samples.size
24
```

DataFrame Exploratory

Basic information with info()

```
>>> samples.info()
<class 'pandas.core.frame.DataFrame'>
Index: 8 entries, Jan to Aug
Data columns (total 3 columns):
length_cm    8 non-null int64
time         8 non-null int64
class        8 non-null object
dtypes: int64(2), object(1)
memory usage: 256.0+ bytes
```

DataFrame Exploratory

head()

```
>>> samples.head()
        length_cm  time class
month
Jan           250  3500     A
Feb           340  4232     A
Mar           198  2956     B
Apr           234  3348     A
May           259  3598     B
```

DataFrame Exploratory

tail()

```
>>> samples.tail()
       length_cm  time class
month
Apr          234  3348     A
May          259  3598     B
Jun          301  4014     B
Jul          268  3601     B
Aug          156  1905     A
```

```
>>> samples.tail(2)
       length_cm  time class
month
Jul          268  3601     B
Aug          156  1905     A
```

DataFrame Exploratory

Extract data with indexes and columns

```
>>> samples['time']['Jan'] # samples.time['Jan']
3500
>>> samples['Jan':'Mar'][:]
       length_cm  time class
month
Jan          250  3500     A
Feb          340  4232     A
>>> samples[['length_cm','class']]
       length_cm class
month
Jan          250     A
Feb          340     A
Mar          198     B
Apr          234     A
```

DataFrame Exploratory

Extract data using loc

```
>>> samples.loc['May','time']
3598
samples.loc['May':'Jun','length_cm':'time']
       length_cm  time
month
May          259  3598
Jun          301  4014
>>> samples.loc[['Mar','May'],'class']
month
Mar    B
May    B
```

DataFrame Exploratory

Extract data using loc

```
>>> samples.loc[['Mar','May','Jun'],'class']
month
Mar    B
May    B
Jun    B
```

DataFrame Exploratory

Extract data using loc

```
>>> samples.loc[:,'time']
month
Jan    3500
Feb    4232
Mar    2956
Apr    3348
May    3598
Jun    4014
Jul    3601
Aug    1905
```

DataFrame Exploratory

Extract data using iloc

```
>>> samples.iloc[4,2]
'B'
>>> samples.iloc[2:4,:]
       length_cm  time class
month
Mar          198  2956     B
Apr          234  3348     A
>>> samples.iloc[2:5,[1,2]]
       time class
month
Mar    2956     B
Apr    3348     A
May    3598     B
```

DataFrame Exploratory

Series versus 1-column DataFrame

- [[]]: DataFrame
- []: Series

```
>>> data = samples[['time']]
>>> type(data)
<class 'pandas.core.frame.DataFrame'>
>>> data = samples['time']
>>> type(data)
<class 'pandas.core.series.Series'>
```

Pandas

FPT Software | FRESHER ACADEMY

Pandas with Datetime

Pandas with Datetime

Data File

```
            Date  Company    Product  Units
  6/2/17 4:30       FPT   Software     11
 6/2/17 23:00   VIETTEL   Hardware     17
 6/3/17 15:00       VIN   Software     21
 6/4/17 14:30        LG   Software     21
 6/4/17 21:00        HP   Hardware     22
```

Pandas with Datetime

Read without parsing date objects

```
>>> sales = pd.read_csv('sales.csv', index_col='Date')
>>> sales.info()
<class 'pandas.core.frame.DataFrame'>
Index: 19 entries, 6/2/17 4:30 to 6/26/17 8:00
Data columns (total 3 columns):
Company    19 non-null object
Product    19 non-null object
Units      19 non-null int64
dtypes: int64(1), object(2)
memory usage: 608.0+ bytes
```

Pandas with Datetime

Read with parsing date objects

```
>>> sales = pd.read_csv('sales.csv',parse_dates=True,index_col='Date')
>>> sales.info()
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 19 entries, 2017-06-02 04:30:00 to 2017-06-26 08:00:00
Data columns (total 3 columns):
Company    19 non-null object
Product    19 non-null object
Units      19 non-null int64
dtypes: int64(1), object(2)
memory usage: 608.0+ bytes
```

Pandas with Datetime

Selecting single datetime

```
>>> sales.loc['2017-06-04 21:00:00', 'Company']
Date
2017-06-04 21:00:00    HP
Name: Company, dtype: object
```

Pandas with Datetime

Selecting whole day

```
>>> sales.loc['2017-06-21']
                     Company    Product  Units
Date
2017-06-21 04:00:00  VIETTEL   Software     11
2017-06-21 19:30:00      FPT   Hardware     11
```

Pandas with Datetime

Partial Datetime

```
>>> sales.loc['2017-06']
>>> sales.loc['2017-Jun-09']
>>> sales.loc['June 09, 2017']
>>> sales.loc['2017']
>>> sales.loc['2017-06-25':'2017-06-26']
                     Company  Product  Units
Date
2017-06-25 01:30:00     VIN  Service     18
2017-06-26 08:00:00      LG  Service     12
```

Pandas with Datetime

Convert strings to datetime

```
>>> pd.to_datetime('6/11/18 23:00')
Timestamp('2018-06-11 23:00:00')
```

```
>>> evening_06_17 = pd.to_datetime(['6/11/17 23:00', '6/11/17 21:00'])
>>> sales.reindex(evening_06_17)
                     Company   Product  Units
2017-06-11 23:00:00     VIN  Software     15
2017-06-11 21:00:00     FPT  Software     12
```

Pandas with Datetime

Resampling

```
>>> daily_mean = sales.resample('D').mean()
>>> daily_mean
            Units
Date
2017-06-02   14.0
2017-06-03   21.0
2017-06-04   21.5
2017-06-05   22.5
```

```
>>> daily_sum_max = sales.resample('D').sum().max()
>>> daily_sum_max
Units    45
```

Pandas with Datetime

Datetime options

Input	Description
‘min’, ‘ T’	minute
‘H’	hour
‘D’	day
‘B’	business day
‘W’	week
‘M’	month
‘Q’	quarter
‘A’	year

Pandas with Datetime

Resampling

```
>>> sales.loc[:,'Units'].resample('2W').sum()
Date
2017-06-04     92
2017-06-18    141
2017-07-02     94
Name: Units, dtype: int64
```

Pandas

DataFrame ETL

DataFrame ETL

Broadcasting

```
>>> samples['length_m'] = samples.length_cm / 100
>>> samples.head(2)
       length_cm  time class  length_m
month
Jan          250  3500     A      2.50
Feb          340  4232     A      3.40
```

DataFrame ETL

Filtering

```
>>> samples = pd.read_csv('samples.csv',index_col='month')
>>> samples[samples.time > 3500]
       length_cm  time group
month
Feb          340  4232     A
May          259  3598     B
Jun          301  4014     B
Jul          268  3601     B
```

DataFrame ETL

Filtering

```
>>> samples[(samples.time > 3500) & (samples.length_cm < 300) ]
       length_cm  time group
month
May          259  3598     B
Jul          268  3601     B
```

DataFrame ETL

Filtering

```
>>> samples[(samples.time < 3000) & (samples.group != 'B') ]
        length_cm  time group
month
Aug           156  1905     A
```

DataFrame ETL

Filtering zeroes and null values

```
>>> samples2 = samples.copy()
>>> samples2.loc['Feb','time'] = 0
>>> samples2.loc['Jun','length_cm'] = None
```

```
>>> samples2.loc[:, samples2.all()]
       length_cm group
month
Jan        250.0     A
. . .
Jun          NaN     B
Jul        268.0     B
Aug        156.0     A
```

DataFrame ETL

Filtering zeroes and null values

```
>>> samples2.loc[:, samples2.any()]
       length_cm  time group
month
Jan        250.0  3500     A
Feb        340.0     0     A
Mar        198.0  2956     B
Apr        234.0  3348     A
May        259.0  3598     B
Jun          NaN  4014     B
Jul        268.0  3601     B
Aug        156.0  1905     A
```

DataFrame ETL

Filtering zeroes and null values

```
>>> samples2.loc[:, samples2.isnull().any()]
       length_cm
month
Jan        250.0
Feb        340.0
Mar        198.0
Apr        234.0
May        259.0
Jun          NaN
Jul        268.0
Aug        156.0
```

DataFrame ETL

Filtering zeroes and null values

```
>>> samples2.loc[:, samples2.notnull().all()]
       time group
month
Jan    3500     A
Feb       0     A
Mar    2956     B
Apr    3348     A
May    3598     B
Jun    4014     B
Jul    3601     B
Aug    1905     A
```

DataFrame ETL

Filtering zeroes and null values

```
>>> samples2.dropna(how='any')
       length_cm  time group
month
Jan        250.0  3500     A
Feb        340.0     0     A
Mar        198.0  2956     B
Apr        234.0  3348     A
May        259.0  3598     B
Jul        268.0  3601     B
Aug        156.0  1905     A
```

DataFrame ETL

Indexes

```
>>> samples.index
Index(['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug'],
dtype='object', name='month')
```

```
>>> samples.index = samples.index.str.upper()
>>> samples.head(2)
       length_cm  time group
month
JAN          250  3500     A
FEB          340  4232     A
```

DataFrame ETL

Indexes

```
>>> samples.index = samples.index.map(str.lower)
>>> samples.head(2)
       length_cm  time group
month
jan          250  3500     A
feb          340  4232     A
```

DataFrame ETL

Multiple-Column Indexes

```
>>> samples.index = samples.index.map(str.lower)
>>> scores.set_index(['Category','Gender'])
```

DataFrame ETL

Pivoting

```
>>> scores = pd.read_csv('scores.csv')
>>> scores.pivot(index='Category', columns='Gender', values='Score')
Gender      F    M
Category
A         2.3  4.5
B         2.4  5.6
C         3.4  6.4
D         0.5  2.3
```

DataFrame ETL

Pivoting Multiple Columns

```
>>> scores.pivot(index='Category', columns='Gender')
          id    Score
Gender     F  M     F    M
Category
A          1  2   2.3  4.5
B          3  4   2.4  5.6
C          5  6   3.4  6.4
D          8  7   0.5  2.3
```

Pandas

FPT Software | FRESHER ACADEMY

Data Aggregation

Data Aggregation

Group By

```
>>> scores = pd.read_csv('scores.csv', index_col='id')
>>> scores.groupby('Category').count()
          Gender  Score
Category
A              2      2
B              2      2
C              2      2
D              2      2
```

Data Aggregation

Group By

```
>>> scores.groupby('Category')['Score'].sum()
Category
A    6.8
B    8.0
C    9.8
D    2.8
```

Data Aggregation

Categorical Data

- Uses less memory
- Speeds up operations like groupby()

```
>>> scores['Category'].unique()
array(['A', 'B', 'C', 'D'], dtype=object)
```

Data Aggregation

Categorical Data

```
>>> scores['Gender'].astype('category')
id
1    F
2    M
3    F
4    M
5    F
6    M
7    M
8    F
Name: Gender, dtype: category
Categories (2, object): [F, M]
```

Data Aggregation

Aggregation

```
>>> points = pd.read_csv('points.csv', index_col='id')
>>> points.groupby('Gender')['Score','Duration'].max()
        Score  Duration
Gender
F         3.4         6
M         6.4         7
```

Data Aggregation

Aggregation

```
>>> points.groupby('Gender')['Score','Duration'].agg(['max','sum'])
        Score        Duration
          max    sum       max sum
Gender
F         3.4    8.6         6  16
M         6.4   18.8         7  21
```

Data Aggregation

Aggregation

```
>>> def data_range(series):
...     return series.max() - series.min()
...
>>> points.groupby('Gender')['Score','Duration'].agg(data_range)
        Score  Duration
Gender
F         2.9         4
M         4.1         4
```

Fresher Academy

FPT Software | FRESHER ACADEMY

Happy Analyzing!