Manipulating DataFrames with pandas

What you will learn

- Extracting, filtering, and transforming data from DataFrames

- Advanced indexing with multiple levels

- Tidying, rearranging and restructuring your data

- Pivoting, melting, and stacking DataFrames

- Identifying and spliuing DataFrames by groups

Indexing DataFrames

A simple DataFrame

```
In [1]: import pandas as pd

In [2]: df = pd.read_csv('sales.csv', index_col='month')

In [3]: df
Out[3]:
        eggs   salt   spam
month
Jan       47   12.0     17
Feb      110   50.0     31
Mar      221   89.0     72
Apr       77   87.0     20
May      132    NaN     52
Jun      205   60.0     55
```

Indexing using square brackets

```
In [4]: df
Out[4]:
       eggs   salt   spam
month
Jan      47    12.     17
Feb     110    50.0    31
Mar     221    89.0    72
Apr      77    87.0    20
May     132     NaN    52
Jun     205    60.0    55


In [5]: df['salt']['Jan']
Out[5]: 12.0
```

Using column atribute and row label

```
In [6]: df
Out[6]:
       eggs    salt    spam
month
Jan       47   12.0      17
Feb      110   50.0      31
Mar      221   89.0      72
Apr       77   87.0      20
May      132    NaN      52
Jun      205   60.0      55

In [7]: df.eggs 'Mar ]
Out[7]: 221     [    '
```

Selecting only some columns

```
In [12]: df_new = df[['salt','eggs']]

In [13]: df_new
Out[13]:
        salt   eggs
month
Jan     12.0     47
Feb     50.0    110
Mar     89.0    221
Apr     87.0     77
May      NaN    132
Jun     60.0    205
```

MANIPULATING DATAFRAMES WITH PANDAS

Slicing DataFrames

sales DataFrame

```
In [1]: df
Out[1]:
      eggs   salt   spam
month
Jan     47   12.0     17
Feb    110   50.0     31
Mar    221   89.0     72
Apr     77   87.0     20
May    132    NaN     52
Jun    205   60.0     55
```

Selecting a column (i.e., Series)

```
In [2]: df['eggs']
Out[2]:
month
Jan      47
Feb     110
Mar     221
Apr      77
May     132
Jun     205
Name: eggs, dtype: int64

In [3]: type(df['eggs'])
Out[3]: pandas.core.series.Series
```

Slicing and indexing a Series

```
In [4]: df['eggs'][1:4]  # Part of the eggs column
Out[4]:
month
Feb    110
Mar    221
Apr     77
Name: eggs, dtype: int64

In [5]: df['eggs'][4]    # The value associated with May
Out[5]: 132
```

Using lists rather than slices (1)

```
In [10]: df.loc['Jan':'May', ['eggs', 'spam']]
Out[10]:
       eggs   spam
month
Jan      47     17
Feb     110     31
Mar     221     72
Apr      77     20
May     132     52
```

Using lists rather than slices (2)

```
In [11]: df.iloc[[0,4,5], 0:2]
Out[11]:
        eggs   salt
month
Jan       47   12.0
May      132    NaN
Jun      205   60.0
```

Series versus 1-column DataFrame

```
# A Series by column name
In [13]: df['egg ']
Out[13]:   [   s
month
Jan      47
Feb     110
Mar     221
Apr      77
May     132
Jun     205
Name: eggs, dtype: int64

In [14]: type(df['eggs'])
Out[14]:
pandas.core.series.Series
```

```
# A DataFrame w/ single column  In [15]:
df          [ 'eggs']
Out[15]:     [      ]
               egg
month   s
Jan       47
Feb      110
Mar      221
Apr       77
May      132
Jun      205

In [16]: type(df[['eggs']])
Out[16]:
pandas.core.frame.DataFrame
```

Filtering DataFrames

Creating a Boolean Series

```
In [1]: df.salt > 60
Out[1]:
month
Jan     False
Feb     False
Mar      True
Apr      True
May     False
Jun     False
Name: salt, dtype: bool
```

Filtering with a Boolean Series

```
In [2]: df[df.salt > 60]
Out[2]:
       eggs   salt   spam
month
Mar     221   89.0     72
Apr      77   87.0     20


In [3]: enough_salt_sold = df.salt > 60


In [4]: df[enough_salt_sold]
Out[4]:
       eggs   salt   spam
month
Mar     221   89.0     72
Apr      77   87.0     20
```

Combining filters

```
In [5]: df[ df.salt >= 50) & (df.eggs < 200)] # Both conditions
Out[5]:    (
        eggs   salt   spam
month
Feb      110   50.0    31
Apr       77   87.0    20


In [6]: df[ df.salt >= 50) | (df.eggs < 200)] # Either condition
Out[6]:    (
        eggs   salt   spam
month
Jan       47   12.0    17
Feb      110   50.0    31
Mar      221   89.0    72
Apr       77   87.0    20
May      132    NaN    52
Jun      205   60.0    55
```

DataFrames with zeros and NaNs

```
In [7]: df2 = df.copy()

In [8]: df2['bacon'] = [0, 0, 50, 60, 70, 80]

In [9]: df2
Out[9]:
        eggs   salt   spam   bacon
month
Jan       47   12.0     17       0
Feb      110   50.0     31       0
Mar      221   89.0     72      50
Apr       77   87.0     20      60
May      132    NaN     52      70
Jun      205   60.0     55      80
```

Select columns with all nonzeros

```
In [10]: df2.loc[:, df2.all()]
Out[10]:
       eggs    salt    spam
month
Jan      47    12.0      17
Feb     110    50.0      31
Mar     221    89.0      72
Apr      77    87.0      20
May     132     NaN      52
Jun     205    60.0      55
```

Select columns with any nonzeros

```
In [11]: df2.loc[:, df2.any()]
Out[11]:
        eggs   salt   spam   bacon

month
Jan       47   12.0     17       0
Feb      110   50.0     31       0
Mar      221   89.0     72      50
Apr       77   87.0     20      60
May      132    NaN     52      70
Jun      205   60.0     55      80
```

Select columns with any NaNs

```
In [12]: df.loc[:, df.isnull().any()]
Out[12]:
        salt
month
Jan     12.0
Feb     50.0
Mar     89.0
Apr     87.0
May      NaN
Jun     60.0
```

Select columns without NaNs

```
In [13]: df.loc[:, df.notnull().all()]
Out[13]:
       eggs   spam
month
Jan      47     17
Feb     110     31
Mar     221     72
Apr      77     20
May     132     52
Jun     205     55
```

Drop rows with any NaNs

```
In [14]: df.dropn (how='any')
Out[14]:    a

         eggs    salt    spam
month
Jan        47   12.0      17
Feb       110   50.0      31
Mar       221   89.0      72
Apr        77   87.0      20
Jun       205   60.0      55
```

Filtering a column based on another

```
In [15]: df.eggs[df.salt > 55]
Out[15]:
month
Mar    221
Apr     77
Jun    205
Name: eggs, dtype: int64
```

Modifying a column based on another

```
In [16]: df.eggs[df.salt > 55] += 5

In [17]: df
Out[17]:
        eggs    salt    spam
month
Jan       47    12.0      17
Feb      110    50.0      31
Mar      226    89.0      72
Apr       82    87.0      20
May      132     NaN      52
Jun      210    60.0      55
```

Transforming DataFrames

DataFrame vectorized methods

```
In [1]: df.floordiv(12)    # Convert to dozens unit
Out[1]:
        eggs   salt   spam
month
Jan        3    1.0      1
Feb        9    4.0      2
Mar       18    7.0      6
Apr        6    7.0      1
May       11    NaN      4
Jun       17    5.0      4
```

NumPy vectorized functions

```
In [2]: import numpy as np

In [3]: np.floor_divide(df, 12)    # Convert to dozens unit
Out[3]:
        eggs   salt   spam
month
Jan      3.0    1.0    1.0
Feb      9.0    4.0    2.0
Mar     18.0    7.0    6.0
Apr      6.0    7.0    1.0
May     11.0    NaN    4.0
Jun     17.0    5.0    4.0
```

Plain Python functions (1)

```
In [4]: def dozens(n):
   ....:        return n//12

In [5]: df.apply(dozens)   # Convert to dozens unit
Out[5]:
        eggs   salt   spam
month
Jan        3    1.0      1
Feb        9    4.0      2
Mar       18    7.0      6
Apr        6    7.0      1
May       11    NaN      4
Jun       17    5.0      4
```

Plain Python functions (2)

```
In [6]: df.apply(lambda n: n//12)
Out[6]:
       eggs   salt   spam
month
Jan       3    1.0      1
Feb       9    4.0      2
Mar      18    7.0      6
Apr       6    7.0      1
May      11    NaN      4
Jun      17    5.0      4
```

Storing a transformation

```
In [7]: df['dozens_of_eggs'] = df.eggs.floordiv(12)

In [8]: df
Out[8]:
       eggs  salt  spam  dozens_of_eggs
month
Jan      47  12.0    17               3
Feb     110  50.0    31               9
Mar     221  89.0    72              18
Apr      77  87.0    20               6
May     132   NaN    52              11
Jun     205  60.0    55              17
```

The DataFrame index

```
In [9]: df
Out[9]:
       eggs    salt   spam   dozens_of_eggs
month
Jan      47   12.0    17                  3
Feb     110   50.0    31                  9
Mar     221   89.0    72                 18
Apr      77   87.0    20                  6
May     132    NaN    52                 11
Jun     205   60.0    55                 17

In [10]: df.index
Out[10]: Index(['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun'],
dtype='object', name='month')
```

Working with string values (1)

```
In [11]: df.index = df.index.str.upper()

In [12]: df
Out[12]:
        eggs    salt    spam    dozens_of_eggs
month
JAN       47    12.0      17                 3
FEB      110    50.0      31                 9
MAR      221    89.0      72                18
APR       77    87.0      20                 6
MAY      132     NaN      52                11
JUN      205    60.0      55                17
```

Working with string values (2)

```
In [13]: df.index = df.index.ma (str.lower)
                                p

In [14]: df
Out[14]:
        eggs    salt    spam    dozens_of_eggs
jan       47    12.0     17                  3
feb      110    50.0     31                  9
mar      221    89.0     72                 18
apr       77    87.0     20                  6
may      132     NaN     52                 11
jun      205    60.0     55                 17
```

Defining columns using other columns

```
In [15]: df['salty_eggs'] = df.salt + df.dozens_of_eggs

In [16]: df
Out[16]:
      eggs   salt   spam   dozens_of_eggs   salty_eggs
jan     47   12.0     17                3         15.0
feb    110   50.0     31                9         59.0
mar    221   89.0     72               18        107.
apr     77   87.0     20                6        0
may    132    NaN     52               11         93.0
jun    205   60.0     55               17          NaN
                                                  77.0
```