COMP1007 Introduction to Python and Its Applications

Lab 6 (Cont.) More Practice on Pandas

Objective:

- to create Pandas objects: Series and DataFrame
- to view data in a DataFrame
- to select data from a DataFrame, such as rows/columns and a combination

Import modules:

```
import pandas as pd

import numpy as np
```

Create Pandas Objects:

pd.Series(my_list): create a series from an iterable my_list

pd.DataFrame(np.random.rand(20, 5)): create a DataFrame with 5 columns and 20 rows of random floats

```
s1 = pd.Series([1, 3, 5, np.nan, 6, 8]) # np.nan means “not a number”, mainly used for missing data

s2 = pd.Series(1, index=list(range(2, 100, 4)))

dates = pd.date_range('20190101', periods=10) # create a DateTimeIndex with 10 dates

print(dates)

df1 = pd.DataFrame(np.random.randn(10, 4), index=dates, columns=list('ABCD'))

print(df1)

df2 = pd.DataFrame({'A': 1.,
                    'B': pd.Timestamp('20190310'),
                    'C': pd.Series(1, index=list(range(4))),
                    'D': np.array([3]*4, dtype='int32'),
                    'E': pd.Categorical(["test", "train", "test", "train"]),
                    'F': 'foo'})  # create a DataFrame from a dictionary
print(df2)

print(df2.dtypes) # show the data type of each column
```

Viewing Data of a DataFrame object:

```
dates = pd.date_range('20190101', periods=10)

df = pd.DataFrame(np.random.randn(10, 5), index=dates, columns=list('ABCDE'))

df.info() # show the index, datatype and memory information

df.head() # show the first 5 rows

df.tail(3) # show the last 3 rows

df.shape # show the shape

print(df.shape[0], df.shape[1]) # show the number of rows and number of columns, respectively

df.index # show the row indices

df.columns # show the column indices

df.describe() # show a quick statistic summary of the numerical columns

df.T # the transpose of the data

df.sort_index(axis = 1, ascending=False) # reorder the columns according the column names

df.sort_values(by='B') # reorder the rows according to the values in column 'B'
```

Selection of Data:

We can select a subset of data from a DataFrame in two different ways: by position or by label.

- Position is the coordinate of the data (i.e., the i-th row, the j-th column). We can use df.iloc[] to select data by position.
- Label means the name of the rows or columns. We can use df.loc[] to select data by label.

Remark: there is a tricky difference between the slices of the two methods. Watch out the samples.

```
# By position through df.iloc[rows, columns]

df.iloc[2] # the 3rd row with all columns, the same as df.iloc[2, :]

df.iloc[0:4] # the first 4 rows with all columns, the same as df.iloc[0:4, :]

df.iloc[0:4, 2:5] # the first 4 rows and columns C/D/E

df.iloc[3, 4] # the data at the 4-th row and the 5-th column

df.iloc[[1, 3, 5, 8], [2, 4]] # 4 rows: [1, 3, 5, 8] and 2 columns: [2, 4]

# By label through df.loc[rows, columns]
```

df['A'] # selecting a single column using column name, which yields a Series

df.A # the same as df['A']. Notice that df[A] will be illegal because A is not a name or position.

df['20190101'] # the row with index '20190101'

df['20190103':'20190108'] # a slice of 6 rows, notice that the row with index '20190108' is included

df.loc['20190102'] # the same as df['20190102'], but this style is recommended

df.loc['20190103':'20190108'] # the same as df['20190103':'20190108']

df.loc[:, ['A', 'C']] # all rows and columns A and C

df.loc['20190102':'20190106', 'C':'E'] # 5 rows and 3 columns, notice that the ending row/col are included

df.loc['20190103', 'D'] # the data at row '20190103' and column D