pandas_DataFrame

October 5, 2019

0.1 Pandas.DataFrame: Hands-on Exercises

0.1.1 DataFrame is a tabular collection of data structure structure with labeled axes (rows and columns)

Creating pandas.DataFrame.

Usually a DataFrame will be created by loading the datasets from a file.

However, Pandas DataFrame can also be created from the lists, dictionary, and from a list of dictionary etc

[2]:
```
import pandas as pd
import numpy as np
```

[ ]:
```
#### Creating dataFrame from a dictionary
```

[3]:
```
data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada', 'Nevada'],
'year': [2000, 2001, 2002, 2001, 2002, 2003],
'pop': [1.5, 1.7, 3.6, 2.4, 2.9, 3.2]}

df = pd.DataFrame(data);
print(df);
```

```
    state  year  pop
0    Ohio  2000  1.5
1    Ohio  2001  1.7
2    Ohio  2002  3.6
3  Nevada  2001  2.4
4  Nevada  2002  2.9
5  Nevada  2003  3.2
```

[9]:
```
## reading a csv file
df = pd.read_csv('nba.csv')
```

[10]:
```
## find number of rows and columns in dataframe
print(df.shape);
```

```
(458, 9)
```

[11]:
```
## find no. of dimensions in dataframe
print(df.ndim);
```

```
2
```

[12]:
```
## check first few rows of dataframe
print(df.head(3));
```

```
            Name            Team  Number Position   Age Height  Weight  \
0  Avery Bradley  Boston Celtics     0.0       PG  25.0    6-2   180.0
1    Jae Crowder  Boston Celtics    99.0       SF  25.0    6-6   235.0
2   John Holland  Boston Celtics    30.0       SG  27.0    6-5   205.0

             College     Salary
0              Texas  7730337.0
1          Marquette  6796117.0
2  Boston University        NaN
```

[17]:
```
## Since default index is assigned as numbers, we can also specify index
df = pd.read_csv('nba.csv', index_col = 'Name')
```

[18]:
```
## Check first few rows
print(df.head(3));
```

```
                         Team  Number Position   Age Height  Weight  \
Name
Avery Bradley  Boston Celtics     0.0       PG  25.0    6-2   180.0
Jae Crowder    Boston Celtics    99.0       SF  25.0    6-6   235.0
John Holland   Boston Celtics    30.0       SG  27.0    6-5   205.0

                         College     Salary
Name
Avery Bradley              Texas  7730337.0
Jae Crowder            Marquette  6796117.0
John Holland   Boston University        NaN
```

[19]:
```
## check column names
print(df.columns);
```

```
Index(['Team', 'Number', 'Position', 'Age', 'Height', 'Weight', 'College',
       'Salary'],
      dtype='object')
```

[21]:
```
## Check structure of dataframe
print(df.info());
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 458 entries, Avery Bradley to nan
Data columns (total 8 columns):
Team        457 non-null object
Number      457 non-null float64
Position    457 non-null object
Age         457 non-null float64
Height      457 non-null object
Weight      457 non-null float64
College     373 non-null object
Salary      446 non-null float64
dtypes: float64(4), object(4)
memory usage: 32.2+ KB
None
```

0.1.2 Indexing and subsetting

Indexing also known as Subset Selection, Indexing in pandas means selecting particular rows and columns of data from a DataFrame.

[50]:
```
## Selecting a single column Age

print(df['Age'].head())
```

```
Name
Avery Bradley    25.0
Jae Crowder      25.0
John Holland     27.0
R.J. Hunter      22.0
Jonas Jerebko    29.0
Name: Age, dtype: float64
```

[52]:
```
## Selecting multiple columns, Age, Height, Weight
print(df[['Age', 'Height', 'Weight']].head())
```

```
                Age Height  Weight
Name
Avery Bradley  25.0    6-2   180.0
Jae Crowder    25.0    6-6   235.0
John Holland   27.0    6-5   205.0
R.J. Hunter    22.0    6-5   185.0
Jonas Jerebko  29.0   6-10   231.0
```

0.1.3 Indexing a DataFrame using .loc[]

.loc method selects subset of data by label of rows and columns

[53]:
```
## Extracting by loc method
## Extract record for Player Avery Bradley
```

```
print(df.loc['Avery Bradley']);
```

```
Team        Boston Celtics
Number                   0
Position                PG
Age                     25
Height                 6-2
Weight                 180
College              Texas
Salary         7.73034e+06
Name: Avery Bradley, dtype: object
```

[54]:
```
## Extract rows of specific columns
df.loc[['Avery Bradley','John Holland'], ['Team', 'Number', 'Position', 'Age']]
```

[54]:
```
                         Team  Number Position   Age
Name
Avery Bradley  Boston Celtics     0.0       PG  25.0
John Holland   Boston Celtics    30.0       SG  27.0
```

0.1.4 Indexing a DataFrame using .iloc[ ] :

.loc allows us to retrieve rows and columns by position

[57]:
```
# retrieve first 3 rows and first 5 columns of dataset
print(df.iloc[0:3, 0:5])
```

```
                         Team  Number Position   Age Height
Name
Avery Bradley  Boston Celtics     0.0       PG  25.0    6-2
Jae Crowder    Boston Celtics    99.0       SF  25.0    6-6
John Holland   Boston Celtics    30.0       SG  27.0    6-5
```

0.1.5 Selecting rows in pandas DataFrame based on conditions

[59]:
```
## Find all players whose age is max in dataset

df.loc[df['Age'] == df['Age'].max()]
```

[59]:
```
                                   Team  Number Position   Age Height  Weight  \
Name
Tim Duncan            San Antonio Spurs    21.0        C  40.0   6-11   250.0
Andre Miller          San Antonio Spurs    24.0       PG  40.0    6-3   200.0
Kevin Garnett    Minnesota Timberwolves    21.0       PF  40.0   6-11   240.0

                   College     Salary
Name
```

```
Tim Duncan      Wake Forest  5250000.0
Andre Miller           Utah   250750.0
Kevin Garnett           NaN  8500000.0
```

[61]:
```
## Find all players who play at Position C and older than 35 years of age

df.loc[(df['Position'] == 'C') & (df['Age'] > 35)]
```

[61]:
```
                                Team  Number Position   Age Height  Weight  \
Name
Matt Bonner        San Antonio Spurs    15.0        C  36.0   6-10   235.0
Tim Duncan         San Antonio Spurs    21.0        C  40.0   6-11   250.0
Nazr Mohammed  Oklahoma City Thunder    13.0        C  38.0   6-10   250.0

                   College     Salary
Name
Matt Bonner        Florida   947276.0
Tim Duncan     Wake Forest  5250000.0
Nazr Mohammed     Kentucky   222888.0
```

0.1.6 isin() method

[64]:
```
## Selecting those players who are from College 'Florida', 'Kentucky'

college_list = ['Florida', 'Kentucky']

df.loc[df['College'].isin(college_list)].head()
```

[64]:
```
                                    Team  Number Position   Age Height  \
Name
James Young               Boston Celtics    13.0       SG  20.0    6-6
Nerlens Noel          Philadelphia 76ers     4.0       PF  22.0   6-11
Patrick Patterson        Toronto Raptors    54.0       PF  27.0    6-9
Marreese Speights  Golden State Warriors     5.0        C  28.0   6-10
Julius Randle         Los Angeles Lakers    30.0       PF  21.0    6-9

                   Weight   College     Salary
Name
James Young         215.0  Kentucky  1749840.0
Nerlens Noel        228.0  Kentucky  3457800.0
Patrick Patterson   235.0  Kentucky  6268675.0
Marreese Speights   255.0   Florida  3815000.0
Julius Randle       250.0  Kentucky  3132240.0
```

[66]:
```
## Selecting those players who are Not from College 'Florida', 'Kentucky'

college_list = ['Florida', 'Kentucky']

df.loc[~ df['College'].isin(college_list)].head()
```

[66]:
```
                         Team  Number Position   Age Height  Weight  \
Name
Avery Bradley  Boston Celtics     0.0       PG  25.0    6-2   180.0
Jae Crowder    Boston Celtics    99.0       SF  25.0    6-6   235.0
John Holland   Boston Celtics    30.0       SG  27.0    6-5   205.0
R.J. Hunter    Boston Celtics    28.0       SG  22.0    6-5   185.0
Jonas Jerebko  Boston Celtics     8.0       PF  29.0   6-10   231.0

                         College     Salary
Name
Avery Bradley              Texas  7730337.0
Jae Crowder            Marquette  6796117.0
John Holland   Boston University        NaN
R.J. Hunter        Georgia State  1148640.0
Jonas Jerebko                NaN  5000000.0
```

[68]:
```
## Find mean salary of players whose age is > 35
df.loc[df['Age'] > 35, 'Salary'].mean()
```

[68]:
```
3959599.5
```

0.1.7 Drop rows from the dataframe based on certain condition applied on a column

[69]:
```
## filter out those rows which does not contain any data
df = df.dropna(how='all')
```

[78]:
```
## drop specific columns
df.drop(['College','Number'], axis=1).head()
```

[78]:
```
                         Team Position   Age Height  Weight     Salary
Name
Avery Bradley  Boston Celtics       PG  25.0    6-2   180.0  7730337.0
Jae Crowder    Boston Celtics       SF  25.0    6-6   235.0  6796117.0
John Holland   Boston Celtics       SG  27.0    6-5   205.0        NaN
R.J. Hunter    Boston Celtics       SG  22.0    6-5   185.0  1148640.0
Jonas Jerebko  Boston Celtics       PF  29.0   6-10   231.0  5000000.0
```

[77]:
```
## drop specific row by index
df.drop(['Avery Bradley'], axis=0).head()
```

[77]:
```
                         Team  Number Position   Age Height  Weight  \
Name
Jae Crowder    Boston Celtics    99.0       SF  25.0    6-6   235.0
John Holland   Boston Celtics    30.0       SG  27.0    6-5   205.0
R.J. Hunter    Boston Celtics    28.0       SG  22.0    6-5   185.0
Jonas Jerebko  Boston Celtics     8.0       PF  29.0   6-10   231.0
Amir Johnson   Boston Celtics    90.0       PF  29.0    6-9   240.0

                         College      Salary
Name
```

```
Jae Crowder              Marquette   6796117.0
John Holland   Boston University         NaN
R.J. Hunter          Georgia State   1148640.0
Jonas Jerebko                  NaN   5000000.0
Amir Johnson                   NaN  12000000.0
```

[83]:
```
## filter out those rows which do not satisfy condition
## Create a a df young_players where age is less than 25

young_players = df.drop(df[df['Age'] >= 25].index)

young_players.head()
```

[83]:
```
                           Team  Number Position   Age Height  Weight  \
Name
R.J. Hunter      Boston Celtics    28.0       SG  22.0    6-5   185.0
Jordan Mickey    Boston Celtics    55.0       PF  21.0    6-8   235.0
Terry Rozier     Boston Celtics    12.0       PG  22.0    6-2   190.0
Marcus Smart     Boston Celtics    36.0       PG  22.0    6-4   220.0
Jared Sullinger  Boston Celtics     7.0        C  24.0    6-9   260.0

                        College     Salary
Name
R.J. Hunter       Georgia State  1148640.0
Jordan Mickey               LSU  1170960.0
Terry Rozier         Louisville  1824360.0
Marcus Smart     Oklahoma State  3431040.0
Jared Sullinger      Ohio State  2569260.0
```

0.1.8 Filter records using query method : Dataframe.query()

[89]:
```
## query(), filter query should be given in string
df.query('Age >=35 and Position =="SG" and College=="Duke"')
```

[89]:
```
                              Team  Number Position   Age Height  Weight  \
Name
Mike Dunleavy        Chicago Bulls    34.0       SG  35.0    6-9   230.0
Dahntay Jones  Cleveland Cavaliers    30.0       SG  35.0    6-6   225.0

              College     Salary
Name
Mike Dunleavy    Duke  4500000.0
Dahntay Jones    Duke        NaN
```

0.1.9 groupby() : split-apply-combine

Splitting the data into groups based on some criteria

Applying a function to each group independently

Combining the results into a data structure

Split step is the most straightforward. In the apply step, we might wish to do one of the following

Aggregation: compute a summary statistic (or statistics) for each group

Transformation: perform some group-specific computations and return a like-indexed object

Filtration: discard some groups, according to a group-wise computation that evaluates True or False

[100]:
```
## Find average salary for each team
df.groupby('Team')['Salary'].mean()
```

[100]:
```
Team
Atlanta Hawks             4.860197e+06
Boston Celtics            4.181505e+06
Brooklyn Nets             3.501898e+06
Charlotte Hornets         5.222728e+06
Chicago Bulls             5.785559e+06
Cleveland Cavaliers       7.642049e+06
Dallas Mavericks          4.746582e+06
Denver Nuggets            4.294424e+06
Detroit Pistons           4.477884e+06
Golden State Warriors     5.924600e+06
Houston Rockets           5.018868e+06
Indiana Pacers            4.450122e+06
Los Angeles Clippers      6.323643e+06
Los Angeles Lakers        4.784695e+06
Memphis Grizzlies         5.467920e+06
Miami Heat                6.347359e+06
Milwaukee Bucks           4.350220e+06
Minnesota Timberwolves    4.593054e+06
New Orleans Pelicans      4.355304e+06
New York Knicks           4.581494e+06
Oklahoma City Thunder     6.251020e+06
Orlando Magic             4.297248e+06
Philadelphia 76ers        2.213778e+06
Phoenix Suns              4.229676e+06
Portland Trail Blazers    3.220121e+06
Sacramento Kings          4.778911e+06
San Antonio Spurs         5.629516e+06
Toronto Raptors           4.741174e+06
Utah Jazz                 4.204006e+06
Washington Wizards        5.088576e+06
```

```
Name: Salary, dtype: float64
```

[102]:
```
## Find mean salary of players based on position they play

df.groupby('Position')['Salary'].mean()
```

[102]:
```
Position
C     5.967052e+06
PF    4.562483e+06
PG    5.077829e+06
SF    4.857393e+06
SG    4.009861e+06
Name: Salary, dtype: float64
```

0.1.10 merge()

[136]:
```
t1 = pd.DataFrame({'A' : [1,5,7,9],
                   'B' : [2,4,6,8]}, index=['K0', 'K1', 'K2', 'K3'])


t2 = pd.DataFrame({'C': [5,9,11,13],
                   'D': [2,6,10,12]}, index = ['K0', 'K1', 'K2', 'K3'])
```

[137]:
```
print(t1);
```

```
    A  B
K0  1  2
K1  5  4
K2  7  6
K3  9  8
```

[138]:
```
print(t2);
```

```
     C   D
K0   5   2
K1   9   6
K2  11  10
K3  13  12
```

[139]:
```
## inner join between t1 and t2 where values of column A and C match

pd.merge(left=t1, right=t2, how='inner', left_on=['A'], right_on=['C'])
```

[139]:
```
   A  B  C  D
0  5  4  5  2
1  9  8  9  6
```

[140]:
```
# t1.A left join t2.C

pd.merge(left=t1, right=t2, how='left', left_on=['A'], right_on=['C'])
```

[140]:
```
   A  B    C    D
0  1  2  NaN  NaN
1  5  4  5.0  2.0
2  7  6  NaN  NaN
3  9  8  9.0  6.0
```

[141]:
```
# t1.A right join t2.C

pd.merge(left=t1, right=t2, how='right', left_on=['A'], right_on=['C'])
```

[141]:
```
     A    B   C   D
0  5.0  4.0   5   2
1  9.0  8.0   9   6
2  NaN  NaN  11  10
3  NaN  NaN  13  12
```

[142]:
```
# t1.A outer join t2.C

pd.merge(left=t1, right=t2, how='outer', left_on=['A'], right_on=['C'])
```

[142]:
```
     A    B     C     D
0  1.0  2.0   NaN   NaN
1  5.0  4.0   5.0   2.0
2  7.0  6.0   NaN   NaN
3  9.0  8.0   9.0   6.0
4  NaN  NaN  11.0  10.0
5  NaN  NaN  13.0  12.0
```

[ ]:
```

```