Pandas

January 11, 2021

1 Podstawy Analizy danych w Pythonie: pandas

1.1 10 stycznia 2021

Ostatnia cześć kursu Pythona będzie dotyczyć biblioteki **pandas**, która służy do analizy danych. Zacznijmy zatem od importu. Przeważnie bibliotekę skraca się do *pd*:

[65]:
```
%matplotlib inline
import sys
import numpy as np
import matplotlib.pyplot as plt
```

```
Matplotlib is building the font cache; this may take a moment.
```

[168]:
```
import pandas as pd
```

Pandas posiada dwie podstawowe struktury danych: * szereg (Series), * ramka danych (DataFrame).

Pandas pozwala na wczytanie danych z wielu formatów plików: * csv: `pd.read_csv` * json: `pd.read_json` * excel: `pd.read_excel` * SQL: `pd.read_sql`

Zobacz: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html

[171]:
```
df = pd.read_csv('/home/tomaszd/codes/big-data-python/labs01/gapminder.csv')

df.head(5)
```

[171]:
```
              Country  female_BMI  male_BMI      gdp  population  \
0         Afghanistan    21.07402  20.62058   1311.0  26528741.0
1             Albania    25.65726  26.44657   8644.0   2968026.0
2             Algeria    26.36841  24.59620  12314.0  34811059.0
3              Angola    23.48431  22.25083   7103.0  19842251.0
4 Antigua and Barbuda    27.50545  25.76602  25736.0     85350.0

   under5mortality  life_expectancy  fertility
0            110.4             52.8       6.20
1             17.9             76.8       1.76
2             29.5             75.5       2.73
3            192.0             56.7       6.43
```

```
4              10.9             75.5       2.16
```

[11]:
```
df = pd.read_excel('./bikes.xlsx', engine='openpyxl')

df.head()
```

[11]:
```
           start_date  start_station_code            end_date  \
0 2019-04-14 07:55:22                6001 2019-04-14 08:07:16
1 2019-04-14 07:59:31                6411 2019-04-14 08:09:18
2 2019-04-14 07:59:55                6097 2019-04-14 08:12:11
3 2019-04-14 07:59:57                6310 2019-04-14 08:27:58
4 2019-04-14 08:00:37                7029 2019-04-14 08:14:12

   end_station_code  duration_sec  is_member
0              6132           713          1
1              6411           587          1
2              6036           736          1
3              6345          1680          1
4              6250           814          0
```

[31]:
```
import sqlite3
select = "select * from logs"
with sqlite3.connect('astro-timeline.sqlite3') as db:
    astro_timeline = pd.read_sql(select, db, parse_dates=['datetime'])

astro_timeline.head()
```

[31]:
```
   id                  datetime    level  \
0   1 1969-07-14 21:00:00+00:00     INFO
1   2 1969-07-16 13:31:53+00:00  WARNING
2   3 1969-07-16 13:33:23+00:00    DEBUG
3   4 1969-07-16 13:34:44+00:00  WARNING
4   5 1969-07-16 13:35:17+00:00    DEBUG

                                  message
0                 Terminal countdown started
1                  S-IC engine ignition (#5)
2  Maximum dynamic pressure (735.17 lb/ft^2)
3                            S-II ignition
4              Launch escape tower jettisoned
```

[26]:
```
df = pd.read_csv('/home/tomaszd/codes/big-data-python/labs01/gapminder.csv',␣
 ↪index_col='Country')

df.head()
```

[26]:
```
                     female_BMI  male_BMI      gdp  population  \
Country
Afghanistan            21.07402  20.62058   1311.0  26528741.0
Albania                25.65726  26.44657   8644.0   2968026.0
Algeria                26.36841  24.59620  12314.0  34811059.0
Angola                 23.48431  22.25083   7103.0  19842251.0
Antigua and Barbuda    27.50545  25.76602  25736.0     85350.0

                     under5mortality  life_expectancy  fertility
Country
Afghanistan                    110.4             52.8       6.20
Albania                         17.9             76.8       1.76
Algeria                         29.5             75.5       2.73
Angola                         192.0             56.7       6.43
Antigua and Barbuda             10.9             75.5       2.16
```

[28]:
```python
df = pd.read_csv("./titanic_train.tsv", sep='\t', index_col='PassengerId')

df.head()
```

[28]:
```
             Survived  Pclass  \
PassengerId
1                   0       3
2                   1       1
3                   1       3
4                   1       1
5                   0       3

                                                          Name     Sex   Age  \
PassengerId
1                                     Braund\t Mr. Owen Harris    male  22.0
2            Cumings\t Mrs. John Bradley (Florence Briggs T…  female  38.0
3                                      Heikkinen\t Miss. Laina  female  26.0
4                 Futrelle\t Mrs. Jacques Heath (Lily May Peel)  female  35.0
5                                    Allen\t Mr. William Henry    male  35.0

             SibSp  Parch            Ticket     Fare Cabin Embarked
PassengerId
1                1      0         A/5 21171   7.2500   NaN        S
2                1      0          PC 17599  71.2833   C85        C
3                0      0  STON/O2. 3101282   7.9250   NaN        S
4                1      0            113803  53.1000  C123        S
5                0      0            373450   8.0500   NaN        S
```

[173]:
```python
import sqlite3
select = "select * from logs"
with sqlite3.connect('astro-timeline.sqlite3') as db:
    astro_timeline = pd.read_sql(select, db, parse_dates=['datetime'])

astro_timeline.to_csv('astro-timeline.csv')
# astro_timeline.head()
```

[173]:
```
   id                  datetime    level  \
0   1 1969-07-14 21:00:00+00:00     INFO
1   2 1969-07-16 13:31:53+00:00  WARNING
2   3 1969-07-16 13:33:23+00:00    DEBUG
3   4 1969-07-16 13:34:44+00:00  WARNING
4   5 1969-07-16 13:35:17+00:00    DEBUG

                                    message
0                 Terminal countdown started
1                   S-IC engine ignition (#5)
2  Maximum dynamic pressure (735.17 lb/ft^2)
3                             S-II ignition
4               Launch escape tower jettisoned
```

2 Szeregi (pd.Series)

[175]:
```
dane = pd.Series([5, 6, 7, 8])
print(dane)
```

```
0    5
1    6
2    7
3    8
dtype: int64
```

[36]:
```
members = {'April': 211819,'May': 682758, 'June': 737011, 'July': 779511,
           'August': 673790, 'September': 673790, 'October': 444177, 'November':
 ↪ 136791,
}
dane = pd.Series(list(members.values()))
print(dane)
```

```
0    211819
1    682758
2    737011
3    779511
4    673790
5    673790
6    444177
7    136791
dtype: int64
```

Czym różni się szereg od listy? Szereg danych posiada indeks, czyli klucz, dzięki ktoremu możemy zindetyfikować dane. Domyślnie, indeks jest ciągiem liczb zaczynających się od zera. Nie musi tak być, możemy podczas tworzenia przekazać również indeks:

[176]:
```
dane = pd.Series(list(members.values()), index=members.keys())
print(dane)
```

```
April        211819
May          682758
June         737011
July         779511
August       673790
September    673790
October      444177
dtype: int64
```

[177]:
```
dane = pd.Series(list(members.values()), index=members.keys())
print(dane['April'])
```

```
211819
```

[179]:
```
dane = pd.Series(list(members.values()), index=members.keys())
keys = ['April', 'September']
dane[keys]
```

[179]:
```
April        211819
September    673790
dtype: int64
```

[182]:
```
dane = pd.Series(list(members.values()), index=members.keys())

print(dane['June': 'September'])
```

```
June         737011
July         779511
August       673790
September    673790
dtype: int64
```

[184]:
```
dane = pd.Series(list(members.values()), index=members.keys())

dane['June'] = 333000
dane
```

[184]:
```
April        211819
May          682758
June         333000
July         779511
```

```
August       673790
September    673790
October      444177
dtype: int64
```

[53]:
```
print(len(dane))
print(dane.shape)
```

```
8
(8,)
```

Przeważnie zbiory danych, na których pracujemy są duże. Stąd, próba ich wyświetlenia może okazać się karkołomna lub nawet niemożliwa. Czasem chcemy tylko zobaczyć pogląd. Do tego służą dwie metody: **head** i **tail**, które zwrócą nam kilka pierwszych lub ostatnich wierszy z szeregu:

[185]:
```
dane.head()
```

[185]:
```
April     211819
May       682758
June      333000
July      779511
August    673790
dtype: int64
```

[186]:
```
dane.tail()
```

[186]:
```
June         333000
July         779511
August       673790
September    673790
October      444177
dtype: int64
```

[188]:
```
dane.sample(5)
```

[188]:
```
June         333000
July         779511
September    673790
May          682758
April        211819
dtype: int64
```

Szeregi są dostosowane do analizy danych. Np. udostępniają prosty sposób do uzyskania podstawowych statystyk:

[189]:
```
dane = pd.Series([1, 3, 2, 3, 1, 1, 2, 3, 2, 3])
print("Średnia:", dane.mean())
print("Mediana:", dane.median())
```

```
Średnia: 2.1
Mediana: 2.0
```

Jak i inne przydatne funkcje:

```
[191]: dane = pd.Series([1, 3, 2, 3, 1, 1, 2, 3, 2, 3])

       # print("Zbiór wartości:", dane.unique())
       print(dane.value_counts())
```

```
3    4
2    3
1    3
dtype: int64
```

Metoda `value_counts` zwraca nam szereg danych, który możemy wykorzystać do dalszych badań. Na przyklad, żeby wyświetlić 5 najczęściej występujących wartości, możemy napisać:

```
[192]: print(dane.value_counts().head(1))
```

```
3    4
dtype: int64
```

Żeby uzyskać wszystkie podstawowe statystyki, możmey wywołać metodę `describe`:

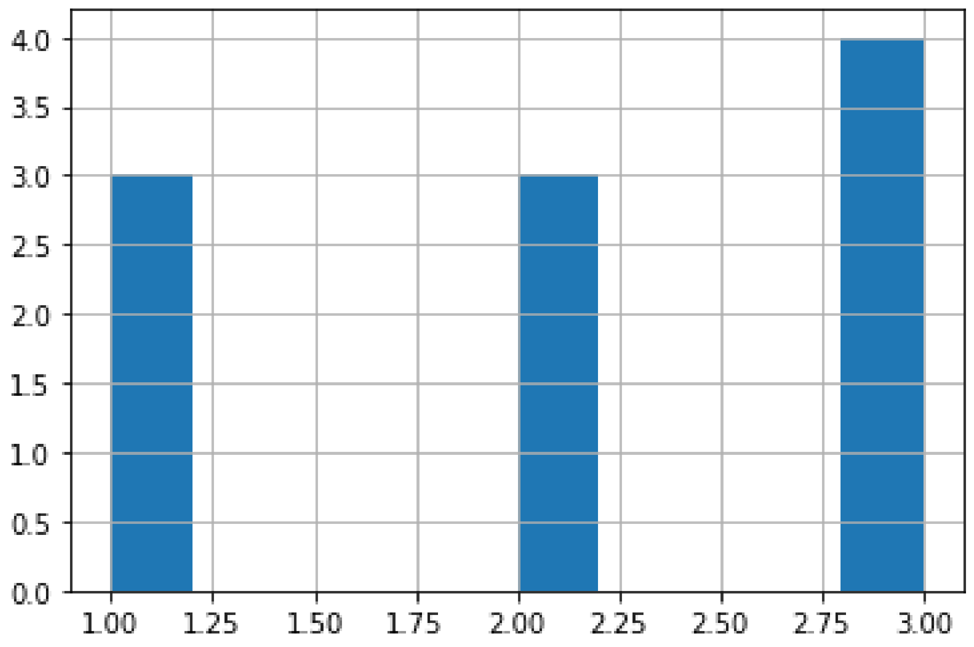
```
[62]: print(dane.describe())
```

```
count    10.000000
mean      2.100000
std       0.875595
min       1.000000
25%       1.250000
50%       2.000000
75%       3.000000
max       3.000000
dtype: float64
```

A żeby wyświetlić je w postaci wykresu:

```
[66]: dane.hist()
```

```
[66]: <AxesSubplot:>
```



[194]:
```
dane = pd.Series(list(members.values()), index=members.keys())

dane.index
```

[194]:
```
Index(['April', 'May', 'June', 'July', 'August', 'September', 'October'],
dtype='object')
```

[196]:
```
dane = pd.Series(list(members.values()), index=members.keys())

dane.values
```

[196]:
```
array([211819, 682758, 737011, 779511, 673790, 673790, 444177])
```

Jeżeli chcemy zmienić cały szereg przy pomocy funkcji, możemy wykorzystać metodę `map`:

[198]:
```
def cube(x):
    return x / 1000

print(dane.map(cube))
# dane
```

```
April        211.819
May          682.758
June         737.011
July         779.511
August       673.790
```

```
September    673.790
October      444.177
dtype: float64
```

Uwaga: w Pythonie istnieją funkcje lambda, które można tu wykorzystać.

[200]:
```
members = {'April': 211819,'May': 682758, 'June': 737011, 'July': 779511,
           'August': 673790, 'September': 673790, 'October': 444177, 'November':
 ↪ 136791, 'December': None
}

dane = pd.Series(list(members.values()), index=members.keys())
print(dane.tail())
```

```
August       673790.0
September    673790.0
October      444177.0
November     136791.0
December          NaN
dtype: float64
```

[203]:
```
dane.notna()
```

[203]:
```
April         True
May           True
June          True
July          True
August        True
September     True
October       True
November      True
December     False
dtype: bool
```

[205]:
```
members = {'April': 211819,'May': 682758, 'June': 737011, 'July': None,
           'August': 673790, 'September': None, 'October': 444177, 'November':␣
 ↪136791, 'December': None
}

dane = pd.Series(list(members.values()), index=members.keys())
print(dane.head())
```

```
April     211819.0
May       682758.0
June      737011.0
July           NaN
August    673790.0
dtype: float64
```

[206]:
```python
dane.dropna()
```

[206]:
```
April       211819.0
May         682758.0
June        737011.0
August      673790.0
October     444177.0
November    136791.0
dtype: float64
```

[208]:
```python
dane.fillna(dane.mean())
```

[208]:
```
April        211819.000000
May          682758.000000
June         737011.000000
July         481057.666667
August       673790.000000
September    481057.666667
October      444177.000000
November     136791.000000
December     481057.666667
dtype: float64
```

[88]:
```python
dane.interpolate()
```

[88]:
```
April        211819.0
May          682758.0
June         737011.0
July         705400.5
August       673790.0
September    558983.5
October      444177.0
November     136791.0
December     136791.0
dtype: float64
```

[209]:
```python
members = {'April': 211819, 'May': 682758, 'June': 737011, 'July': 779511,␣
 ↪'August': 673790,
           'September': 673790, 'October': 444177, 'November': 136791}


members_series = pd.Series(list(members.values()), index=members.keys())

print(members_series + 1000)
```

```
April        212819
May          683758
```

```
June         738011
July         780511
August       674790
September    674790
October      445177
November     137791
dtype: int64
```

[92]:
```
members = {'April': 211819, 'May': 682758, 'June': 737011, 'July': 779511,␣
 ↪'August': 673790,
           'September': 673790, 'October': 444177, 'November': 136791}

occasionals = {'April': 32058, 'May': 147898, 'June': 171494, 'July': 194316,␣
 ↪'August': 206809,
               'September': 140492, 'October': 53596,'November': 10516}

members_series = pd.Series(list(members.values()), index=members.keys())
occasionals_series = pd.Series(list(occasionals.values()), index=occasionals.
 ↪keys())

all_series = members_series + occasionals_series
all_series
```

[92]:
```
April        243877
May          830656
June         908505
July         973827
August       880599
September    814282
October      497773
November     147307
dtype: int64
```

[94]:
```
members = {'April': 211819, 'May': 682758, 'June': 737011, 'July': 779511,␣
 ↪'August': 673790,
           'September': 673790, 'October': 444177}

occasionals = {'May': 147898, 'June': 171494, 'July': 194316, 'August': 206809,
               'September': 140492, 'October': 53596,'November': 10516}

members_series = pd.Series(list(members.values()), index=members.keys())
occasionals_series = pd.Series(list(occasionals.values()), index=occasionals.
 ↪keys())

all_series = members_series + occasionals_series
all_series
```

[94]:
```
April              NaN
August        880599.0
July          973827.0
June          908505.0
May           830656.0
November           NaN
October       497773.0
September     814282.0
dtype: float64
```

[210]:
```
members = {'April': 211819, 'June': 737011, 'July': 779511, 'August': 673790,
           'September': 673790, 'October': 444177}

occasionals = {'May': 147898, 'June': 171494, 'July': 194316, 'August': 206809,
               'September': 140492, 'October': 53596,'November': 10516}

members_series = pd.Series(list(members.values()), index=members.keys())
occasionals_series = pd.Series(list(occasionals.values()), index=occasionals.
 ↪keys())

all_series = members_series.add(occasionals_series, fill_value=0)
all_series
```

[210]:
```
April         211819.0
August        880599.0
July          973827.0
June          908505.0
May           147898.0
November       10516.0
October       497773.0
September     814282.0
dtype: float64
```

2.1 Ramki danych (pd.DataFrame)

[211]:
```
members = {'April': 211819, 'May': 682758, 'June': 737011, 'July': 779511,␣
 ↪'August': 673790,
           'September': 673790, 'October': 444177}

occasionals = {'May': 147898, 'June': 171494, 'July': 194316, 'August': 206809,
               'September': 140492, 'October': 53596,'November': 10516}

df = pd.DataFrame({'members': members, 'occasionals': occasionals})

df.head()
```

[211]:
```
         members  occasionals
April   211819.0          NaN
May     682758.0     147898.0
June    737011.0     171494.0
July    779511.0     194316.0
August  673790.0     206809.0
```

[212]:
```
data = [{'members': 682758, 'occasionals': 147898},
        {'members': 737011, 'occasionals': 171494},
        {'members': 779511, 'occasionals': 194316}]

df = pd.DataFrame(data)

df.head()
```

[212]:
```
   members  occasionals
0   682758       147898
1   737011       171494
2   779511       194316
```

[213]:
```
data = [(682758, 147898), (737011, 171494), (779511, 194316)]

df = pd.DataFrame(data)

df.head()
```

[213]:
```
        0       1
0  682758  147898
1  737011  171494
2  779511  194316
```

[215]:
```
data = [(682758, 147898), (737011, 171494), (779511, 194316)]

df = pd.DataFrame(data)
df.columns = ['members', 'occasionals']

df.head()
```

[215]:
```
   members  occasionals
0   682758       147898
1   737011       171494
2   779511       194316
```

[216]:
```
data = [(682758, 147898), (737011, 171494), (779511, 194316)]

df = pd.DataFrame(data, columns=['members', 'occasionals'])
```

```
df.head()
```

```
[216]:    members  occasionals
       0   682758       147898
       1   737011       171494
       2   779511       194316
```

```
[114]: members = {'April': 211819, 'May': 682758, 'June': 737011, 'July': 779511,␣
        ↪'August': 673790,
                   'September': 673790, 'October': 444177}

       occasionals = {'May': 147898, 'June': 171494, 'July': 194316, 'August': 206809,
                       'September': 140492, 'October': 53596,'November': 10516}

       df = pd.DataFrame({'members': members, 'occasionals': occasionals})

       df.index
```

```
[114]: Index(['April', 'May', 'June', 'July', 'August', 'September', 'October',
               'November'],
              dtype='object')
```

```
[219]: members = {'April': 211819, 'May': 682758, 'June': 737011, 'July': 779511,␣
        ↪'August': 673790,
                   'September': 673790, 'October': 444177}

       occasionals = {'May': 147898, 'June': 171494, 'July': 194316, 'August': 206809,
                       'September': 140492, 'October': 53596,'November': 10516}

       df = pd.DataFrame({'members': members, 'occasionals': occasionals})

       df.values
```

```
[219]: array([[211819.,     nan],
              [682758., 147898.],
              [737011., 171494.],
              [779511., 194316.],
              [673790., 206809.],
              [673790., 140492.],
              [444177.,  53596.],
              [    nan,  10516.]])
```

```
[220]: df.head()
```

```
[220]:          members  occasionals
       April   211819.0          NaN
       May     682758.0     147898.0
```

```
June     737011.0     171494.0
July     779511.0     194316.0
August   673790.0     206809.0
```

[221]:
```
df.tail()
```

[221]:
```
              members  occasionals
July         779511.0     194316.0
August       673790.0     206809.0
September    673790.0     140492.0
October      444177.0      53596.0
November          NaN      10516.0
```

[222]:
```
print(df.max())
```

```
members         779511.0
occasionals     206809.0
dtype: float64
```

[119]:
```
print(df.describe())
```

```
             members    occasionals
count       7.000000       7.000000
mean   600408.000000  132160.142857
std    201535.422382   73336.753852
min    211819.000000   10516.000000
25%    558983.500000   97044.000000
50%    673790.000000  147898.000000
75%    709884.500000  182905.000000
max    779511.000000  206809.000000
```

[120]:
```
print(df.sample(3))
```

```
            members  occasionals
June       737011.0     171494.0
November        NaN      10516.0
July       779511.0     194316.0
```

[224]:
```
# df.members

df['members']
```

[224]:
```
April        211819.0
May          682758.0
June         737011.0
July         779511.0
August       673790.0
September    673790.0
```

```
October      444177.0
November          NaN
Name: members, dtype: float64
```

```
[125]: df[['members', 'occasionals']]
```

```
[125]:             members  occasionals
       April      211819.0          NaN
       May        682758.0     147898.0
       June       737011.0     171494.0
       July       779511.0     194316.0
       August     673790.0     206809.0
       September  673790.0     140492.0
       October    444177.0      53596.0
       November        NaN      10516.0
```

```
[225]: df.loc['August']
```

```
[225]: members        673790.0
       occasionals    206809.0
       Name: August, dtype: float64
```

```
[128]: df.loc[['August', 'September']]
```

```
[128]:             members  occasionals
       August     673790.0     206809.0
       September  673790.0     140492.0
```

```
[129]: df.loc['June': 'September']
```

```
[129]:             members  occasionals
       June       737011.0     171494.0
       July       779511.0     194316.0
       August     673790.0     206809.0
       September  673790.0     140492.0
```

```
[130]: df.loc['June': 'September', 'members']
```

```
[130]: June         737011.0
       July         779511.0
       August       673790.0
       September    673790.0
       Name: members, dtype: float64
```

```
[226]: df.at['June', 'members']
```

```
[226]: 737011.0
```

```
[133]: df.at['June', 'members'] = 123456
       df.at['June', 'members']
```

```
[133]: 123456.0
```

```
[228]: df['all_rides'] = df['members'] + df['occasionals']

       print(df.head())
```

```
          members  occasionals  all_rides
April    211819.0          NaN        NaN
May      682758.0     147898.0   830656.0
June     737011.0     171494.0   908505.0
July     779511.0     194316.0   973827.0
August   673790.0     206809.0   880599.0
```

```
[229]: df.loc['December'] = [0, 0, 0]

       print(df.tail())
```

```
            members  occasionals  all_rides
August     673790.0     206809.0   880599.0
September  673790.0     140492.0   814282.0
October    444177.0      53596.0   497773.0
November        NaN      10516.0        NaN
December        0.0          0.0        0.0
```

```
[151]: df.drop('April')
```

```
[151]:             members  occasionals  all_rides
        May        682758.0     147898.0   830656.0
        June       737011.0     171494.0   908505.0
        July       779511.0     194316.0   973827.0
        August     673790.0     206809.0   880599.0
        September  673790.0     140492.0   814282.0
        October    444177.0      53596.0   497773.0
        November        NaN      10516.0        NaN
        December        0.0          0.0        0.0
```

```
[154]: df.drop('members', axis='columns') # df.drop(columns='members')
```

```
[154]:            occasionals  all_rides
        April              NaN        NaN
        May           147898.0   830656.0
        June          171494.0   908505.0
        July          194316.0   973827.0
        August        206809.0   880599.0
```

```
September     140492.0   814282.0
October        53596.0   497773.0
November       10516.0        NaN
December           0.0        0.0
```

[155]:
```
df.transpose()
```

[155]:
```
                April       May      June      July    August  September  \
members      211819.0  682758.0  737011.0  779511.0  673790.0   673790.0
occasionals       NaN  147898.0  171494.0  194316.0  206809.0   140492.0
all_rides         NaN  830656.0  908505.0  973827.0  880599.0   814282.0

              October  November  December
members      444177.0       NaN       0.0
occasionals   53596.0   10516.0       0.0
all_rides    497773.0       NaN       0.0
```

[231]:
```
df = pd.read_csv("./titanic_train.tsv", sep='\t', index_col='PassengerId')

df.head()
```

[231]:
```
             Survived  Pclass  \
PassengerId
1                   0       3
2                   1       1
3                   1       3
4                   1       1
5                   0       3

                                                          Name     Sex   Age  \
PassengerId
1                                      Braund\t Mr. Owen Harris    male  22.0
2            Cumings\t Mrs. John Bradley (Florence Briggs T…  female  38.0
3                                       Heikkinen\t Miss. Laina  female  26.0
4                Futrelle\t Mrs. Jacques Heath (Lily May Peel)  female  35.0
5                                     Allen\t Mr. William Henry    male  35.0

             SibSp  Parch            Ticket     Fare Cabin Embarked
PassengerId
1                1      0         A/5 21171   7.2500   NaN        S
2                1      0          PC 17599  71.2833   C85        C
3                0      0  STON/O2. 3101282   7.9250   NaN        S
4                1      0            113803  53.1000  C123        S
5                0      0            373450   8.0500   NaN        S
```

[18]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 11 columns):
Survived    891 non-null int64
Pclass      891 non-null int64
Name        891 non-null object
Sex         891 non-null object
Age         714 non-null float64
SibSp       891 non-null int64
Parch       891 non-null int64
Ticket      891 non-null object
Fare        891 non-null float64
Cabin       204 non-null object
Embarked    889 non-null object
dtypes: float64(2), int64(4), object(5)
memory usage: 83.5+ KB
```

[22]:
```
print(df.shape)
```

```
(891, 11)
```

[232]:
```
# df['Survived'] == 1
df.loc[df['Survived'] == 1].head()
```

[232]:
```
PassengerId
1       False
2        True
3        True
4        True
5       False
        …
887     False
888      True
889     False
890      True
891     False
Name: Survived, Length: 891, dtype: bool
```

[140]:
```
df['Survived'] == 1
```

[140]:
```
PassengerId
1       False
2        True
3        True
4        True
5       False
        …
887     False
```

```
888     True
889    False
890     True
891    False
Name: Survived, Length: 891, dtype: bool
```

[144]:
```
survived = df['Survived'] == 1
first_class = df['Pclass'] == 1

# df[survived & first_class]
# df[survived | first_class]
df[~first_class]
```

[144]:
```
             Survived  Pclass                                         Name  \
PassengerId
1                   0       3                     Braund\t Mr. Owen Harris
3                   1       3                      Heikkinen\t Miss. Laina
5                   0       3                    Allen\t Mr. William Henry
6                   0       3                            Moran\t Mr. James
8                   0       3              Palsson\t Master. Gosta Leonard
…                …      …                                           …
885                 0       3                      Sutehall\t Mr. Henry Jr
886                 0       3         Rice\t Mrs. William (Margaret Norton)
887                 0       2                       Montvila\t Rev. Juozas
889                 0       3  Johnston\t Miss. Catherine Helen "Carrie"
891                 0       3                          Dooley\t Mr. Patrick

                Sex   Age  SibSp  Parch            Ticket     Fare Cabin  \
PassengerId
1              male  22.0      1      0         A/5 21171   7.2500   NaN
3            female  26.0      0      0  STON/O2. 3101282   7.9250   NaN
5              male  35.0      0      0            373450   8.0500   NaN
6              male   NaN      0      0            330877   8.4583   NaN
8              male   2.0      3      1            349909  21.0750   NaN
…             …   …    …     …                …      …   …
885            male  25.0      0      0   SOTON/OQ 392076   7.0500   NaN
886          female  39.0      0      5            382652  29.1250   NaN
887            male  27.0      0      0            211536  13.0000   NaN
889          female   NaN      1      2        W./C. 6607  23.4500   NaN
891            male  32.0      0      0            370376   7.7500   NaN

            Embarked
PassengerId
1                  S
3                  S
5                  S
6                  Q
```

```
8                  S
…               …
885                S
886                Q
887                S
889                S
891                Q

[675 rows x 11 columns]
```

[237]:
```
df[['Sex', 'Survived']].groupby('Sex').mean()
```

[237]:
```
        Survived
Sex
female  0.742038
male    0.188908
```

[166]:
```
df.groupby(['Sex', 'Pclass']).mean()
```

[166]:
```
               Survived        Age     SibSp     Parch        Fare
Sex    Pclass
female 1       0.968085  34.611765  0.553191  0.457447  106.125798
       2       0.921053  28.722973  0.486842  0.605263   21.970121
       3       0.500000  21.750000  0.895833  0.798611   16.118810
male   1       0.368852  41.281386  0.311475  0.278689   67.226127
       2       0.157407  30.740707  0.342593  0.222222   19.741782
       3       0.135447  26.507589  0.498559  0.224784   12.661633
```

[ ]: