pandas

2018 年 2 月 8 日

1 pandas

第 15-20 个视频

1.1 15. 数据读取

```
In [1]: import pandas

        food_info = pandas.read_csv("food_info.csv")
        print(type(food_info))
        #print(food_info)
        #print(help(pandas.read_csv))

        print(food_info.dtypes)
```

```
<class 'pandas.core.frame.DataFrame'>
NDB_No                int64
Shrt_Desc            object
Water_(g)           float64
Energ_Kcal            int64
Protein_(g)         float64
Lipid_Tot_(g)       float64
Ash_(g)             float64
Carbohydrt_(g)      float64
Fiber_TD_(g)        float64
Sugar_Tot_(g)       float64
Calcium_(mg)        float64
Iron_(mg)           float64
Magnesium_(mg)      float64
Phosphorus_(mg)     float64
Potassium_(mg)      float64
```

```
Sodium_(mg)        float64
Zinc_(mg)          float64
Copper_(mg)        float64
Manganese_(mg)     float64
Selenium_(mcg)     float64
Vit_C_(mg)         float64
Thiamin_(mg)       float64
Riboflavin_(mg)    float64
Niacin_(mg)        float64
Vit_B6_(mg)        float64
Vit_B12_(mcg)      float64
Vit_A_IU           float64
Vit_A_RAE          float64
Vit_E_(mg)         float64
Vit_D_mcg          float64
Vit_D_IU           float64
Vit_K_(mcg)        float64
FA_Sat_(g)         float64
FA_Mono_(g)        float64
FA_Poly_(g)        float64
Cholestrl_(mg)     float64
dtype: object
```

```
In [2]: first_rows = food_info.head()
        print(first_rows)
        print(food_info.head(3))
        print(food_info.columns)
        print(food_info.shape)
```

```
   NDB_No                 Shrt_Desc  Water_(g)  Energ_Kcal  Protein_(g)  \
0    1001          BUTTER WITH SALT      15.87         717         0.85
1    1002  BUTTER WHIPPED WITH SALT      15.87         717         0.85
2    1003      BUTTER OIL ANHYDROUS       0.24         876         0.28
3    1004               CHEESE BLUE      42.41         353        21.40
4    1005              CHEESE BRICK      41.11         371        23.24

   Lipid_Tot_(g)  Ash_(g)  Carbohydrt_(g)  Fiber_TD_(g)  Sugar_Tot_(g)  \
0          81.11     2.11            0.06           0.0           0.06
1          81.11     2.11            0.06           0.0           0.06
2          99.48     0.00            0.00           0.0           0.00
```

```
3          28.74     5.11            2.34           0.0           0.50
4          29.68     3.18            2.79           0.0           0.51

        ...         Vit_A_IU  Vit_A_RAE  Vit_E_(mg)  Vit_D_mcg  Vit_D_IU  \
0       ...           2499.0      684.0        2.32        1.5      60.0
1       ...           2499.0      684.0        2.32        1.5      60.0
2       ...           3069.0      840.0        2.80        1.8      73.0
3       ...            721.0      198.0        0.25        0.5      21.0
4       ...           1080.0      292.0        0.26        0.5      22.0

   Vit_K_(mcg)  FA_Sat_(g)  FA_Mono_(g)  FA_Poly_(g)  Cholestrl_(mg)
0          7.0      51.368       21.021        3.043           215.0
1          7.0      50.489       23.426        3.012           219.0
2          8.6      61.924       28.732        3.694           256.0
3          2.4      18.669        7.778        0.800            75.0
4          2.5      18.764        8.598        0.784            94.0

[5 rows x 36 columns]
   NDB_No                 Shrt_Desc  Water_(g)  Energ_Kcal  Protein_(g)  \
0    1001          BUTTER WITH SALT      15.87         717         0.85
1    1002  BUTTER WHIPPED WITH SALT      15.87         717         0.85
2    1003      BUTTER OIL ANHYDROUS       0.24         876         0.28

   Lipid_Tot_(g)  Ash_(g)  Carbohydrt_(g)  Fiber_TD_(g)  Sugar_Tot_(g)  \
0          81.11     2.11            0.06           0.0           0.06
1          81.11     2.11            0.06           0.0           0.06
2          99.48     0.00            0.00           0.0           0.00

        ...         Vit_A_IU  Vit_A_RAE  Vit_E_(mg)  Vit_D_mcg  Vit_D_IU  \
0       ...           2499.0      684.0        2.32        1.5      60.0
1       ...           2499.0      684.0        2.32        1.5      60.0
2       ...           3069.0      840.0        2.80        1.8      73.0

   Vit_K_(mcg)  FA_Sat_(g)  FA_Mono_(g)  FA_Poly_(g)  Cholestrl_(mg)
0          7.0      51.368       21.021        3.043           215.0
1          7.0      50.489       23.426        3.012           219.0
2          8.6      61.924       28.732        3.694           256.0

[3 rows x 36 columns]
```

```
Index(['NDB_No', 'Shrt_Desc', 'Water_(g)', 'Energ_Kcal', 'Protein_(g)',
       'Lipid_Tot_(g)', 'Ash_(g)', 'Carbohydrt_(g)', 'Fiber_TD_(g)',
       'Sugar_Tot_(g)', 'Calcium_(mg)', 'Iron_(mg)', 'Magnesium_(mg)',
       'Phosphorus_(mg)', 'Potassium_(mg)', 'Sodium_(mg)', 'Zinc_(mg)',
       'Copper_(mg)', 'Manganese_(mg)', 'Selenium_(mcg)', 'Vit_C_(mg)',
       'Thiamin_(mg)', 'Riboflavin_(mg)', 'Niacin_(mg)', 'Vit_B6_(mg)',
       'Vit_B12_(mcg)', 'Vit_A_IU', 'Vit_A_RAE', 'Vit_E_(mg)', 'Vit_D_mcg',
       'Vit_D_IU', 'Vit_K_(mcg)', 'FA_Sat_(g)', 'FA_Mono_(g)', 'FA_Poly_(g)',
       'Cholestrl_(mg)'],
      dtype='object')
(8618, 36)
```

1.2 16. 索引与计算

In [3]:
```
#pandas uses zero-indexing
#Series object representing the row at index 0.
#print(food_info.loc[0])

# Series object representing the seventh row.
print(food_info.loc[6])
```

```
NDB_No                          1007
Shrt_Desc           CHEESE CAMEMBERT
Water_(g)                       51.8
Energ_Kcal                       300
Protein_(g)                     19.8
Lipid_Tot_(g)                  24.26
Ash_(g)                         3.68
Carbohydrt_(g)                  0.46
Fiber_TD_(g)                       0
Sugar_Tot_(g)                   0.46
Calcium_(mg)                     388
Iron_(mg)                       0.33
Magnesium_(mg)                    20
Phosphorus_(mg)                  347
Potassium_(mg)                   187
Sodium_(mg)                      842
Zinc_(mg)                       2.38
Copper_(mg)                    0.021
Manganese_(mg)                 0.038
```

```
Selenium_(mcg)               14.5
Vit_C_(mg)                      0
Thiamin_(mg)                0.028
Riboflavin_(mg)             0.488
Niacin_(mg)                  0.63
Vit_B6_(mg)                 0.227
Vit_B12_(mcg)                 1.3
Vit_A_IU                      820
Vit_A_RAE                     241
Vit_E_(mg)                   0.21
Vit_D_mcg                     0.4
Vit_D_IU                       18
Vit_K_(mcg)                     2
FA_Sat_(g)                 15.259
FA_Mono_(g)                 7.023
FA_Poly_(g)                 0.724
Cholestrl_(mg)                 72
Name: 6, dtype: object
```

1.3 查看数据类型

```
In [4]: #object - For string values
        #int - For integer values
        #float - For float values
        #datetime - For time values
        #bool - For Boolean values
        print(food_info.dtypes)
```

```
NDB_No               int64
Shrt_Desc           object
Water_(g)          float64
Energ_Kcal           int64
Protein_(g)        float64
Lipid_Tot_(g)      float64
Ash_(g)            float64
Carbohydrt_(g)     float64
Fiber_TD_(g)       float64
Sugar_Tot_(g)      float64
Calcium_(mg)       float64
Iron_(mg)          float64
```

```
Magnesium_(mg)     float64
Phosphorus_(mg)    float64
Potassium_(mg)     float64
Sodium_(mg)        float64
Zinc_(mg)          float64
Copper_(mg)        float64
Manganese_(mg)     float64
Selenium_(mcg)     float64
Vit_C_(mg)         float64
Thiamin_(mg)       float64
Riboflavin_(mg)    float64
Niacin_(mg)        float64
Vit_B6_(mg)        float64
Vit_B12_(mcg)      float64
Vit_A_IU           float64
Vit_A_RAE          float64
Vit_E_(mg)         float64
Vit_D_mcg          float64
Vit_D_IU           float64
Vit_K_(mcg)        float64
FA_Sat_(g)         float64
FA_Mono_(g)        float64
FA_Poly_(g)        float64
Cholestrl_(mg)     float64
dtype: object
```

```
In [5]: # Returns a DataFrame containing the rows at indexes 3, 4, 5, and 6.
        print(food_info.loc[3:6])
        print(food_info.loc[[2,5,10]])
```

```
   NDB_No         Shrt_Desc  Water_(g)  Energ_Kcal  Protein_(g)  \
3    1004       CHEESE BLUE      42.41         353        21.40
4    1005      CHEESE BRICK      41.11         371        23.24
5    1006       CHEESE BRIE      48.42         334        20.75
6    1007  CHEESE CAMEMBERT      51.80         300        19.80

   Lipid_Tot_(g)  Ash_(g)  Carbohydrt_(g)  Fiber_TD_(g)  Sugar_Tot_(g)  \
3          28.74     5.11            2.34           0.0           0.50
4          29.68     3.18            2.79           0.0           0.51
5          27.68     2.70            0.45           0.0           0.45
```

```
6         24.26     3.68            0.46            0.0            0.46

       ...         Vit_A_IU  Vit_A_RAE  Vit_E_(mg)  Vit_D_mcg  Vit_D_IU  \
3      ...            721.0      198.0        0.25        0.5      21.0
4      ...           1080.0      292.0        0.26        0.5      22.0
5      ...            592.0      174.0        0.24        0.5      20.0
6      ...            820.0      241.0        0.21        0.4      18.0

   Vit_K_(mcg)  FA_Sat_(g)  FA_Mono_(g)  FA_Poly_(g)  Cholestrl_(mg)
3          2.4      18.669        7.778        0.800            75.0
4          2.5      18.764        8.598        0.784            94.0
5          2.3      17.410        8.013        0.826           100.0
6          2.0      15.259        7.023        0.724            72.0

[4 rows x 36 columns]
    NDB_No            Shrt_Desc  Water_(g)  Energ_Kcal  Protein_(g)  \
2     1003  BUTTER OIL ANHYDROUS       0.24         876         0.28
5     1006          CHEESE BRIE      48.42         334        20.75
10    1011         CHEESE COLBY      38.20         394        23.76

    Lipid_Tot_(g)  Ash_(g)  Carbohydrt_(g)  Fiber_TD_(g)  Sugar_Tot_(g)  \
2           99.48     0.00            0.00           0.0           0.00
5           27.68     2.70            0.45           0.0           0.45
10          32.11     3.36            2.57           0.0           0.52

        ...         Vit_A_IU  Vit_A_RAE  Vit_E_(mg)  Vit_D_mcg  Vit_D_IU  \
2       ...           3069.0      840.0        2.80        1.8      73.0
5       ...            592.0      174.0        0.24        0.5      20.0
10      ...            994.0      264.0        0.28        0.6      24.0

    Vit_K_(mcg)  FA_Sat_(g)  FA_Mono_(g)  FA_Poly_(g)  Cholestrl_(mg)
2           8.6      61.924       28.732        3.694           256.0
5           2.3      17.410        8.013        0.826           100.0
10          2.7      20.218        9.280        0.953            95.0

[3 rows x 36 columns]
```

```
In [6]: # Series object representing the "NDB_No" column.
        #print(food_info["NDB_No"])
```

```
print(food_info[["NDB_No","Shrt_Desc"]])
```

```
      NDB_No                                         Shrt_Desc
0       1001                                  BUTTER WITH SALT
1       1002                          BUTTER WHIPPED WITH SALT
2       1003                              BUTTER OIL ANHYDROUS
3       1004                                       CHEESE BLUE
4       1005                                      CHEESE BRICK
5       1006                                       CHEESE BRIE
6       1007                                  CHEESE CAMEMBERT
7       1008                                    CHEESE CARAWAY
8       1009                                    CHEESE CHEDDAR
9       1010                                   CHEESE CHESHIRE
10      1011                                      CHEESE COLBY
11      1012              CHEESE COTTAGE CRMD LRG OR SML CURD
12      1013                      CHEESE COTTAGE CRMD W/FRUIT
13      1014   CHEESE COTTAGE NONFAT UNCRMD DRY LRG OR SML CURD
14      1015                 CHEESE COTTAGE LOWFAT 2% MILKFAT
15      1016                 CHEESE COTTAGE LOWFAT 1% MILKFAT
16      1017                                      CHEESE CREAM
17      1018                                       CHEESE EDAM
18      1019                                       CHEESE FETA
19      1020                                    CHEESE FONTINA
20      1021                                    CHEESE GJETOST
21      1022                                      CHEESE GOUDA
22      1023                                    CHEESE GRUYERE
23      1024                                  CHEESE LIMBURGER
24      1025                                   CHEESE MONTEREY
25      1026                         CHEESE MOZZARELLA WHL MILK
26      1027              CHEESE MOZZARELLA WHL MILK LO MOIST
27      1028                 CHEESE MOZZARELLA PART SKIM MILK
28      1029               CHEESE MOZZARELLA LO MOIST PART-SKIM
29      1030                                   CHEESE MUENSTER
...      ...                                               ...
8588   43544             BABYFOOD CRL RICE W/ PEARS & APPL DRY INST
8589   43546                        BABYFOOD BANANA NO TAPIOCA STR
8590   43550                        BABYFOOD BANANA APPL DSSRT STR
8591   43566         SNACKS TORTILLA CHIPS LT (BAKED W/ LESS OIL)
8592   43570  CEREALS RTE POST HONEY BUNCHES OF OATS HONEY RSTD
8593   43572                         POPCORN MICROWAVE LOFAT&NA
```

```
8594   43585                        BABYFOOD FRUIT SUPREME DSSRT
8595   43589                                CHEESE SWISS LOW FAT
8596   43595               BREAKFAST BAR CORN FLAKE CRUST W/FRUIT
8597   43597                             CHEESE MOZZARELLA LO NA
8598   43598                            MAYONNAISE DRSNG NO CHOL
8599   44005                           OIL CORN PEANUT AND OLIVE
8600   44018                     SWEETENERS TABLETOP FRUCTOSE LIQ
8601   44048                               CHEESE FOOD IMITATION
8602   44055                                 CELERY FLAKES DRIED
8603   44061             PUDDINGS CHOC FLAVOR LO CAL INST DRY MIX
8604   44074                      BABYFOOD GRAPE JUC NO SUGAR CND
8605   44110                     JELLIES RED SUGAR HOME PRESERVED
8606   44158                          PIE FILLINGS BLUEBERRY CND
8607   44203                 COCKTAIL MIX NON-ALCOHOLIC CONCD FRZ
8608   44258              PUDDINGS CHOC FLAVOR LO CAL REG DRY MIX
8609   44259  PUDDINGS ALL FLAVORS XCPT CHOC LO CAL REG DRY MIX
8610   44260  PUDDINGS ALL FLAVORS XCPT CHOC LO CAL INST DRY...
8611   48052                                  VITAL WHEAT GLUTEN
8612   80200                                       FROG LEGS RAW
8613   83110                                     MACKEREL SALTED
8614   90240                          SCALLOP (BAY&SEA) CKD STMD
8615   90480                                          SYRUP CANE
8616   90560                                           SNAIL RAW
8617   93600                                     TURTLE GREEN RAW

[8618 rows x 2 columns]
```

```
In [7]: print(food_info.columns)
```

```
Index(['NDB_No', 'Shrt_Desc', 'Water_(g)', 'Energ_Kcal', 'Protein_(g)',
       'Lipid_Tot_(g)', 'Ash_(g)', 'Carbohydrt_(g)', 'Fiber_TD_(g)',
       'Sugar_Tot_(g)', 'Calcium_(mg)', 'Iron_(mg)', 'Magnesium_(mg)',
       'Phosphorus_(mg)', 'Potassium_(mg)', 'Sodium_(mg)', 'Zinc_(mg)',
       'Copper_(mg)', 'Manganese_(mg)', 'Selenium_(mcg)', 'Vit_C_(mg)',
       'Thiamin_(mg)', 'Riboflavin_(mg)', 'Niacin_(mg)', 'Vit_B6_(mg)',
       'Vit_B12_(mcg)', 'Vit_A_IU', 'Vit_A_RAE', 'Vit_E_(mg)', 'Vit_D_mcg',
       'Vit_D_IU', 'Vit_K_(mcg)', 'FA_Sat_(g)', 'FA_Mono_(g)', 'FA_Poly_(g)',
       'Cholestrl_(mg)'],
      dtype='object')
```

1.4 columns.tolist() 列出列名

```
In [8]: print(food_info.columns.tolist())
```

```
['NDB_No', 'Shrt_Desc', 'Water_(g)', 'Energ_Kcal', 'Protein_(g)', 'Lipid_Tot_(g)', 'Ash_(g)', 'Car
```

```
In [9]: col_names = food_info.columns.tolist()
        gram_columns = []

        for c in col_names:
            if c.endswith("(g)"):
                gram_columns.append(c)
        gram_df = food_info[gram_columns]
        print(gram_df.head(9))
```

```
   Water_(g)  Protein_(g)  Lipid_Tot_(g)  Ash_(g)  Carbohydrt_(g)  \
0      15.87         0.85          81.11     2.11            0.06
1      15.87         0.85          81.11     2.11            0.06
2       0.24         0.28          99.48     0.00            0.00
3      42.41        21.40          28.74     5.11            2.34
4      41.11        23.24          29.68     3.18            2.79
5      48.42        20.75          27.68     2.70            0.45
6      51.80        19.80          24.26     3.68            0.46
7      39.28        25.18          29.20     3.28            3.06
8      37.10        24.04          33.82     3.71            1.33

   Fiber_TD_(g)  Sugar_Tot_(g)  FA_Sat_(g)  FA_Mono_(g)  FA_Poly_(g)
0           0.0           0.06      51.368       21.021        3.043
1           0.0           0.06      50.489       23.426        3.012
2           0.0           0.00      61.924       28.732        3.694
3           0.0           0.50      18.669        7.778        0.800
4           0.0           0.51      18.764        8.598        0.784
5           0.0           0.45      17.410        8.013        0.826
6           0.0           0.46      15.259        7.023        0.724
7           0.0            NaN      18.584        8.275        0.830
8           0.0           0.28      19.368        8.428        1.433
```

```
In [10]: print(food_info["FA_Poly_(g)"].head(6))
```

```
0    3.043
1    3.012
```

```
2    3.694
3    0.800
4    0.784
5    0.826
Name: FA_Poly_(g), dtype: float64
```

```
In [11]: print(food_info["FA_Poly_(g)"]**2)
```

```
0           9.259849
1           9.072144
2          13.645636
3           0.640000
4           0.614656
5           0.682276
6           0.524176
7           0.688900
8           2.053489
9           0.756900
10          0.908209
11          0.015129
12          0.015376
13          0.000009
14          0.006889
15          0.000961
16          2.064969
17          0.442225
18          0.349281
19          2.735716
20          0.879844
21          0.431649
22          3.003289
23          0.245025
24          0.808201
25          0.585225
26          0.605284
27          0.222784
28          0.741321
29          0.436921
              ...
8588        0.053361
```

```
8589       0.001681
8590       0.002209
8591      25.240576
8592       1.708249
8593      12.759184
8594       0.004624
8595       0.032400
8596       0.810000
8597       0.259081
8598    2073.800521
8599    1091.179089
8600       0.000000
8601      56.791296
8602       1.071225
8603       0.017161
8604       0.000000
8605       0.000064
8606       0.000000
8607       0.000081
8608       0.016900
8609       0.002500
8610       0.187489
8611       0.656100
8612       0.010404
8613      38.564100
8614       0.049284
8615       0.000000
8616       0.063504
8617       0.028900
Name: FA_Poly_(g), Length: 8618, dtype: float64
```

In [12]:
```
#It applies the arithmetic operator to the first value in both columns, the second value
water_energy = food_info["Water_(g)"] * food_info["Energ_Kcal"]
iron_grams = food_info["Iron_(mg)"] / 1000
food_info["Iron_(g)"] = iron_grams
print(food_info.shape)
```

```
(8618, 37)
```

In [13]:
```
#Score=2Œ(Protein_(g))0.75Œ(Lipid_Tot_(g))
```

```
weighted_protein = food_info["Protein_(g)"] * 2
weighted_fat = -0.75 * food_info["Lipid_Tot_(g)"]
initial_rating = weighted_protein + weighted_fat
```

In [14]:
```
# the "Vit_A_IU" column ranges from 0 to 100000, while the "Fiber_TD_(g)" column ranges f
#For certain calculations, columns like "Vit_A_IU" can have a greater effect on the resul
#due to the scale of the values
# The largest value in the "Energ_Kcal" column.
max_calories = food_info["Energ_Kcal"].max()
# Divide the values in "Energ_Kcal" by the largest value.
normalized_calories = food_info["Energ_Kcal"] / max_calories
normalized_protein = food_info["Protein_(g)"] / food_info["Protein_(g)"].max()
normalized_fat = food_info["Lipid_Tot_(g)"] / food_info["Lipid_Tot_(g)"].max()
food_info["Normalized_Protein"] = normalized_protein
food_info["Normalized_Fat"] = normalized_fat
```

1.5 17. 数据预处理

1.6 ascending=true/false 排序

In [15]:
```
#By default, pandas will sort the data by the column we specify in ascending order and re
# Sorts the DataFrame in-place, rather than returning a new DataFrame.
#print food_info["Sodium_(mg)"]
food_info.sort_values("Sodium_(mg)", inplace=True)
print (food_info["Sodium_(mg)"])
#Sorts by descending order, rather than ascending.
food_info.sort_values("Sodium_(mg)", inplace=True, ascending=False)
print (food_info["Sodium_(mg)"])
```

```
760     0.0
758     0.0
405     0.0
761     0.0
2269    0.0
763     0.0
764     0.0
770     0.0
774     0.0
396     0.0
395     0.0
6827    0.0
```

```
394     0.0
393     0.0
391     0.0
390     0.0
787     0.0
788     0.0
2270    0.0
2231    0.0
407     0.0
748     0.0
409     0.0
747     0.0
702     0.0
703     0.0
704     0.0
705     0.0
706     0.0
707     0.0
       ...
8153    NaN
8155    NaN
8156    NaN
8157    NaN
8158    NaN
8159    NaN
8160    NaN
8161    NaN
8163    NaN
8164    NaN
8165    NaN
8167    NaN
8169    NaN
8170    NaN
8172    NaN
8173    NaN
8174    NaN
8175    NaN
8176    NaN
8177    NaN
```

```
8178    NaN
8179    NaN
8180    NaN
8181    NaN
8183    NaN
8184    NaN
8185    NaN
8195    NaN
8251    NaN
8267    NaN
Name: Sodium_(mg), Length: 8618, dtype: float64
276     38758.0
5814    27360.0
6192    26050.0
1242    26000.0
1245    24000.0
1243    24000.0
1244    23875.0
292     17000.0
1254    11588.0
5811    10600.0
8575     9690.0
291      8068.0
1249     8031.0
5812     7893.0
1292     7851.0
293      7203.0
4472     7027.0
4836     6820.0
1261     6580.0
3747     6008.0
1266     5730.0
4835     5586.0
4834     5493.0
1263     5356.0
1553     5203.0
1552     5053.0
1251     4957.0
1257     4843.0
```

```
294       4616.0
8613      4450.0
            ...
8153         NaN
8155         NaN
8156         NaN
8157         NaN
8158         NaN
8159         NaN
8160         NaN
8161         NaN
8163         NaN
8164         NaN
8165         NaN
8167         NaN
8169         NaN
8170         NaN
8172         NaN
8173         NaN
8174         NaN
8175         NaN
8176         NaN
8177         NaN
8178         NaN
8179         NaN
8180         NaN
8181         NaN
8183         NaN
8184         NaN
8185         NaN
8195         NaN
8251         NaN
8267         NaN
Name: Sodium_(mg), Length: 8618, dtype: float64
```

泰坦尼克数据训练

```
In [16]: import pandas as pd
         import numpy as np
         titanic_survival = pd.read_csv("titanic_train.csv")
```

```
titanic_survival.head()
```

Out[16]:
```
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

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

In [17]:
```
#The Pandas library uses NaN, which stands for "not a number", to indicate a missing valu
#we can use the pandas.isnull() function which takes a pandas series and returns a series
age = titanic_survival["Age"]
#print(titanic_survival.loc[0:10])
age_is_null = pd.isnull(age)
#print(age_is_null)
age_null_true = age[age_is_null]
#print(age_null_true)
age_null_count = len(age_null_true)
print(age_null_count)
```

```
177
```

1.7 18. 数据预处理（常用方法）

In [18]:
```
#The result of this is that mean_age would be nan. This is because any calculations we do
mean_age = sum(titanic_survival["Age"]) / len(titanic_survival["Age"])
print (mean_age)
```

```
nan
```

1.8 查看缺失值

```
In [19]: #we have to filter out the missing values before we calculate the mean.
         good_ages = titanic_survival["Age"][age_is_null == False]
         print (good_ages)
         correct_mean_age = sum(good_ages) / len(good_ages)
         print (correct_mean_age)
```

```
0       22.0
1       38.0
2       26.0
3       35.0
4       35.0
6       54.0
7        2.0
8       27.0
9       14.0
10       4.0
11      58.0
12      20.0
13      39.0
14      14.0
15      55.0
16       2.0
18      31.0
20      35.0
21      34.0
22      15.0
23      28.0
24       8.0
25      38.0
27      19.0
30      40.0
33      66.0
34      28.0
35      42.0
37      21.0
38      18.0
```

```
        ...
856    45.0
857    51.0
858    24.0
860    41.0
861    21.0
862    48.0
864    24.0
865    42.0
866    27.0
867    31.0
869     4.0
870    26.0
871    47.0
872    33.0
873    47.0
874    28.0
875    15.0
876    20.0
877    19.0
879    56.0
880    25.0
881    33.0
882    22.0
883    28.0
884    25.0
885    39.0
886    27.0
887    19.0
889    26.0
890    32.0
Name: Age, Length: 714, dtype: float64
29.69911764705882
```

1.9 运算求均值

```
In [20]: # missing data is so common that many pandas methods automatically filter for it
         correct_mean_age = titanic_survival["Age"].mean()
         print (correct_mean_age)
```

```
29.69911764705882
```

```
In [21]: #mean fare for each class
         passenger_classes = [1, 2, 3]
         fares_by_class = {}
         for this_class in passenger_classes:
             pclass_rows = titanic_survival[titanic_survival["Pclass"] == this_class]
             pclass_fares = pclass_rows["Fare"]
             fare_for_class = pclass_fares.mean()
             fares_by_class[this_class] = fare_for_class
         print (fares_by_class)
```

```
{1: 84.15468749999992, 2: 20.66218315217391, 3: 13.675550101832997}
```

1.10 pivot_table 透视表

```
In [22]: #index tells the method which column to group by
         #values is the column that we want to apply the calculation to
         #aggfunc specifies the calculation we want to perform
         passenger_survival = titanic_survival.pivot_table(index="Pclass", values="Survived", aggf
         print (passenger_survival)
```

```
        Survived
Pclass
1       0.629630
2       0.472826
3       0.242363
```

```
In [23]: passenger_age = titanic_survival.pivot_table(index="Pclass", values="Age")
         print(passenger_age)
```

```
              Age
Pclass
1       38.233441
2       29.877630
3       25.140620
```

```
In [24]: port_stats = titanic_survival.pivot_table(index="Embarked", values=["Fare","Survived"], a
         print(port_stats)
```

```
              Fare  Survived
Embarked
C       10072.2962        93
Q        1022.2543        30
S       17439.3988       217
```

```
In [25]: #specifying axis=1 or axis='columns' will drop any columns that have null values
         drop_na_columns = titanic_survival.dropna(axis=1)
         new_titanic_survival = titanic_survival.dropna(axis=0,subset=["Age", "Sex"])
         print (new_titanic_survival)
```

```
     PassengerId  Survived  Pclass  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
6              7         0       1
7              8         0       3
8              9         1       3
9             10         1       2
10            11         1       3
11            12         1       1
12            13         0       3
13            14         0       3
14            15         0       3
15            16         1       2
16            17         0       3
18            19         0       3
20            21         0       2
21            22         1       2
22            23         1       3
23            24         1       1
24            25         0       3
25            26         1       3
27            28         0       1
30            31         0       1
33            34         0       2
34            35         0       1
35            36         0       1
```

```
37            38          0       3
38            39          0       3
..           ...        ...     ...
856          857          1       1
857          858          1       1
858          859          1       3
860          861          0       3
861          862          0       2
862          863          1       1
864          865          0       2
865          866          1       2
866          867          1       2
867          868          0       1
869          870          1       3
870          871          0       3
871          872          1       1
872          873          0       1
873          874          0       3
874          875          1       2
875          876          1       3
876          877          0       3
877          878          0       3
879          880          1       1
880          881          1       2
881          882          0       3
882          883          0       3
883          884          0       2
884          885          0       3
885          886          0       3
886          887          0       2
887          888          1       1
889          890          1       1
890          891          0       3

                                                  Name     Sex   Age  SibSp  \
0                              Braund, Mr. Owen Harris    male  22.0      1
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                               Heikkinen, Miss. Laina  female  26.0      0
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
```

4	Allen, Mr. William Henry	male	35.0	0
6	McCarthy, Mr. Timothy J	male	54.0	0
7	Palsson, Master. Gosta Leonard	male	2.0	3
8	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0
9	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1
10	Sandstrom, Miss. Marguerite Rut	female	4.0	1
11	Bonnell, Miss. Elizabeth	female	58.0	0
12	Saundercock, Mr. William Henry	male	20.0	0
13	Andersson, Mr. Anders Johan	male	39.0	1
14	Vestrom, Miss. Hulda Amanda Adolfina	female	14.0	0
15	Hewlett, Mrs. (Mary D Kingcome)	female	55.0	0
16	Rice, Master. Eugene	male	2.0	4
18	Vander Planke, Mrs. Julius (Emelia Maria Vande...	female	31.0	1
20	Fynney, Mr. Joseph J	male	35.0	0
21	Beesley, Mr. Lawrence	male	34.0	0
22	McGowan, Miss. Anna "Annie"	female	15.0	0
23	Sloper, Mr. William Thompson	male	28.0	0
24	Palsson, Miss. Torborg Danira	female	8.0	3
25	Asplund, Mrs. Carl Oscar (Selma Augusta Emilia...	female	38.0	1
27	Fortune, Mr. Charles Alexander	male	19.0	3
30	Uruchurtu, Don. Manuel E	male	40.0	0
33	Wheadon, Mr. Edward H	male	66.0	0
34	Meyer, Mr. Edgar Joseph	male	28.0	1
35	Holverson, Mr. Alexander Oskar	male	42.0	1
37	Cann, Mr. Ernest Charles	male	21.0	0
38	Vander Planke, Miss. Augusta Maria	female	18.0	2
..	...	...	...	...
856	Wick, Mrs. George Dennick (Mary Hitchcock)	female	45.0	1
857	Daly, Mr. Peter Denis	male	51.0	0
858	Baclini, Mrs. Solomon (Latifa Qurban)	female	24.0	0
860	Hansen, Mr. Claus Peter	male	41.0	2
861	Giles, Mr. Frederick Edward	male	21.0	1
862	Swift, Mrs. Frederick Joel (Margaret Welles Ba...	female	48.0	0
864	Gill, Mr. John William	male	24.0	0
865	Bystrom, Mrs. (Karolina)	female	42.0	0
866	Duran y More, Miss. Asuncion	female	27.0	1
867	Roebling, Mr. Washington Augustus II	male	31.0	0
869	Johnson, Master. Harold Theodor	male	4.0	1
870	Balkic, Mr. Cerin	male	26.0	0

```
871  Beckwith, Mrs. Richard Leonard (Sallie Monypeny)  female  47.0      1
872                            Carlsson, Mr. Frans Olof    male  33.0      0
873                         Vander Cruyssen, Mr. Victor    male  47.0      0
874              Abelson, Mrs. Samuel (Hannah Wizosky)  female  28.0      1
875                   Najib, Miss. Adele Kiamie "Jane"  female  15.0      0
876                      Gustafsson, Mr. Alfred Ossian    male  20.0      0
877                               Petroff, Mr. Nedelio    male  19.0      0
879       Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)  female  56.0      0
880       Shelley, Mrs. William (Imanita Parrish Hall)  female  25.0      0
881                                 Markun, Mr. Johann    male  33.0      0
882                       Dahlberg, Miss. Gerda Ulrika  female  22.0      0
883                      Banfield, Mr. Frederick James    male  28.0      0
884                              Sutehall, Mr. Henry Jr    male  25.0      0
885             Rice, Mrs. William (Margaret Norton)  female  39.0      0
886                             Montvila, Rev. Juozas    male  27.0      0
887                       Graham, Miss. Margaret Edith  female  19.0      0
889                              Behr, Mr. Karl Howell    male  26.0      0
890                                Dooley, Mr. Patrick    male  32.0      0

     Parch            Ticket     Fare        Cabin Embarked
0        0         A/5 21171   7.2500          NaN        S
1        0          PC 17599  71.2833          C85        C
2        0  STON/O2. 3101282   7.9250          NaN        S
3        0            113803  53.1000         C123        S
4        0            373450   8.0500          NaN        S
6        0             17463  51.8625          E46        S
7        1            349909  21.0750          NaN        S
8        2            347742  11.1333          NaN        S
9        0            237736  30.0708          NaN        C
10       1           PP 9549  16.7000           G6        S
11       0            113783  26.5500         C103        S
12       0         A/5. 2151   8.0500          NaN        S
13       5            347082  31.2750          NaN        S
14       0            350406   7.8542          NaN        S
15       0            248706  16.0000          NaN        S
16       1            382652  29.1250          NaN        Q
18       0            345763  18.0000          NaN        S
20       0            239865  26.0000          NaN        S
21       0            248698  13.0000          D56        S
```

```
22       0              330923    8.0292           NaN        Q
23       0              113788   35.5000            A6        S
24       1              349909   21.0750           NaN        S
25       5              347077   31.3875           NaN        S
27       2               19950  263.0000   C23 C25 C27        S
30       0            PC 17601   27.7208           NaN        C
33       0         C.A. 24579   10.5000           NaN        S
34       0            PC 17604   82.1708           NaN        C
35       0              113789   52.0000           NaN        S
37       0         A./5. 2152    8.0500           NaN        S
38       0              345764   18.0000           NaN        S
..     ...                 ...       ...           ...      ...
856      1               36928  164.8667           NaN        S
857      0              113055   26.5500           E17        S
858      3                2666   19.2583           NaN        C
860      0              350026   14.1083           NaN        S
861      0               28134   11.5000           NaN        S
862      0               17466   25.9292           D17        S
864      0              233866   13.0000           NaN        S
865      0              236852   13.0000           NaN        S
866      0       SC/PARIS 2149   13.8583           NaN        C
867      0            PC 17590   50.4958           A24        S
869      1              347742   11.1333           NaN        S
870      0              349248    7.8958           NaN        S
871      1               11751   52.5542           D35        S
872      0                 695    5.0000   B51 B53 B55        S
873      0              345765    9.0000           NaN        S
874      0           P/PP 3381   24.0000           NaN        C
875      0                2667    7.2250           NaN        C
876      0                7534    9.8458           NaN        S
877      0              349212    7.8958           NaN        S
879      1               11767   83.1583           C50        C
880      1              230433   26.0000           NaN        S
881      0              349257    7.8958           NaN        S
882      0                7552   10.5167           NaN        S
883      0  C.A./SOTON 34068   10.5000           NaN        S
884      0    SOTON/OQ 392076    7.0500           NaN        S
885      5              382652   29.1250           NaN        Q
886      0              211536   13.0000           NaN        S
```

```
887       0            112053   30.0000          B42         S
889       0            111369   30.0000         C148         C
890       0            370376    7.7500          NaN         Q

[714 rows x 12 columns]
```

```
In [26]: row_index_83_age = titanic_survival.loc[83,"Age"]
         row_index_1000_pclass = titanic_survival.loc[766,"Pclass"]
         print (row_index_83_age)
         print (row_index_1000_pclass)
```

```
28.0
1
```

1.11 19. 自定义函数

```
In [27]: new_titanic_survival = titanic_survival.sort_values("Age",ascending=False)
         print (new_titanic_survival[0:10])
         print("————————————————————————————————排序后索引值也重新建立————————————————————————————————
         titanic_reindexed = new_titanic_survival.reset_index(drop=True)
         print(titanic_reindexed.iloc[0:10])
```

```
     PassengerId  Survived  Pclass                                 Name  \
630          631         1       1  Barkworth, Mr. Algernon Henry Wilson
851          852         0       3                  Svensson, Mr. Johan
493          494         0       1              Artagaveytia, Mr. Ramon
96            97         0       1             Goldschmidt, Mr. George B
116          117         0       3                  Connors, Mr. Patrick
672          673         0       2           Mitchell, Mr. Henry Michael
745          746         0       1          Crosby, Capt. Edward Gifford
33            34         0       2                Wheadon, Mr. Edward H
54            55         0       1        Ostby, Mr. Engelhart Cornelius
280          281         0       3                     Duane, Mr. Frank

      Sex   Age  SibSp  Parch      Ticket     Fare Cabin Embarked
630  male  80.0      0      0       27042  30.0000   A23        S
851  male  74.0      0      0      347060   7.7750   NaN        S
493  male  71.0      0      0    PC 17609  49.5042   NaN        C
96   male  71.0      0      0    PC 17754  34.6542    A5        C
116  male  70.5      0      0      370369   7.7500   NaN        Q
```

```
672  male  70.0      0      0  C.A. 24580  10.5000   NaN        S
745  male  70.0      1      1   WE/P 5735  71.0000   B22        S
33   male  66.0      0      0  C.A. 24579  10.5000   NaN        S
54   male  65.0      0      1      113509  61.9792   B30        C
280  male  65.0      0      0      336439   7.7500   NaN        Q
```

————————————————排序后索引值也重新建立————————————————

```
   PassengerId  Survived  Pclass                                 Name   Sex  \
0          631         1       1  Barkworth, Mr. Algernon Henry Wilson  male
1          852         0       3                   Svensson, Mr. Johan  male
2          494         0       1               Artagaveytia, Mr. Ramon  male
3           97         0       1             Goldschmidt, Mr. George B  male
4          117         0       3                  Connors, Mr. Patrick  male
5          673         0       2           Mitchell, Mr. Henry Michael  male
6          746         0       1          Crosby, Capt. Edward Gifford  male
7           34         0       2                 Wheadon, Mr. Edward H  male
8           55         0       1        Ostby, Mr. Engelhart Cornelius  male
9          281         0       3                      Duane, Mr. Frank  male

    Age  SibSp  Parch      Ticket     Fare Cabin Embarked
0  80.0      0      0       27042  30.0000   A23        S
1  74.0      0      0      347060   7.7750   NaN        S
2  71.0      0      0    PC 17609  49.5042   NaN        C
3  71.0      0      0    PC 17754  34.6542    A5        C
4  70.5      0      0      370369   7.7500   NaN        Q
5  70.0      0      0  C.A. 24580  10.5000   NaN        S
6  70.0      1      1   WE/P 5735  71.0000   B22        S
7  66.0      0      0  C.A. 24579  10.5000   NaN        S
8  65.0      0      1      113509  61.9792   B30        C
9  65.0      0      0      336439   7.7500   NaN        Q
```

In [28]:
```
# This function returns the hundredth item from a series
def hundredth_row(column):
    # Extract the hundredth item
    hundredth_item = column.iloc[99]
    return hundredth_item

# Return the hundredth item from each column
hundredth_row = titanic_survival.apply(hundredth_row)
print (hundredth_row)
```

```
PassengerId                  100
Survived                       0
Pclass                         2
Name              Kantor, Mr. Sinai
Sex                         male
Age                           34
SibSp                          1
Parch                          0
Ticket                    244367
Fare                          26
Cabin                        NaN
Embarked                       S
dtype: object
```

```
In [29]: def not_null_count(column):
             column_null = pd.isnull(column)
             null = column[column_null]
             return len(null)

         column_null_count = titanic_survival.apply(not_null_count)
         print (column_null_count)
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

1.12 which_class 加标签

In [30]:
```
#By passing in the axis=1 argument, we can use the DataFrame.apply() method to iterate ov
def which_class(row):
    pclass = row['Pclass']
    if pd.isnull(pclass):
        return "Unknown"
    elif pclass == 1:
        return "First Class"
    elif pclass == 2:
        return "Second Class"
    elif pclass == 3:
        return "Third Class"

classes = titanic_survival.apply(which_class, axis=1)
print (classes)
```

```
0        Third Class
1        First Class
2        Third Class
3        First Class
4        Third Class
5        Third Class
6        First Class
7        Third Class
8        Third Class
9       Second Class
10       Third Class
11       First Class
12       Third Class
13       Third Class
14       Third Class
15      Second Class
16       Third Class
17      Second Class
18       Third Class
19       Third Class
20      Second Class
21      Second Class
22       Third Class
23       First Class
```

```
24       Third Class
25       Third Class
26       Third Class
27       First Class
28       Third Class
29       Third Class
            ...
861     Second Class
862      First Class
863      Third Class
864     Second Class
865     Second Class
866     Second Class
867      First Class
868      Third Class
869      Third Class
870      Third Class
871      First Class
872      First Class
873      Third Class
874     Second Class
875      Third Class
876      Third Class
877      Third Class
878      Third Class
879      First Class
880     Second Class
881      Third Class
882      Third Class
883     Second Class
884      Third Class
885      Third Class
886     Second Class
887      First Class
888      Third Class
889      First Class
890      Third Class
Length: 891, dtype: object
```

In [31]:
```python
def is_minor(row):
    if row["Age"] < 18:
        return True
    else:
        return False

minors = titanic_survival.apply(is_minor, axis=1)
#print (minors)

def generate_age_label(row):
    age = row["Age"]
    if pd.isnull(age):
        return "unknown"
    elif age < 18:
        return "minor"
    else:
        return "adult"

age_labels = titanic_survival.apply(generate_age_label, axis=1)
print (age_labels)
```

```
0         adult
1         adult
2         adult
3         adult
4         adult
5       unknown
6         adult
7         minor
8         adult
9         minor
10        minor
11        adult
12        adult
13        adult
14        minor
15        adult
16        minor
17      unknown
18        adult
```

```
19     unknown
20       adult
21       adult
22       minor
23       adult
24       minor
25       adult
26     unknown
27       adult
28     unknown
29     unknown
        ...
861      adult
862      adult
863    unknown
864      adult
865      adult
866      adult
867      adult
868    unknown
869      minor
870      adult
871      adult
872      adult
873      adult
874      adult
875      minor
876      adult
877      adult
878    unknown
879      adult
880      adult
881      adult
882      adult
883      adult
884      adult
885      adult
886      adult
887      adult
```

```
888    unknown
889      adult
890      adult
Length: 891, dtype: object
```

```
In [32]: titanic_survival['age_labels'] = age_labels
         age_group_survival = titanic_survival.pivot_table(index="age_labels", values="Survived")
         print (age_group_survival)
```

```
            Survived
age_labels
adult       0.381032
minor       0.539823
unknown     0.293785
```

1.13 20.Series 结果

```
In [33]: #Series (collection of values)
         #DataFrame (collection of Series objects)
         #Panel (collection of DataFrame objects)

         #A Series object can hold many data types, including
         #float - for representing float values
         #int - for representing integer values
         #bool - for representing Boolean values
         #datetime64[ns] - for representing date & time, without time-zone
         #datetime64[ns, tz] - for representing date & time, with time-zone
         #timedelta[ns] - for representing differences in dates & times (seconds, minutes, etc.)
         #category - for representing categorical values
         #object - for representing String values

         #FILM - film name
         #RottenTomatoes - Rotten Tomatoes critics average score
         #RottenTomatoes_User - Rotten Tomatoes user average score
         #RT_norm - Rotten Tomatoes critics average score (normalized to a 0 to 5 point system)
         #RT_user_norm - Rotten Tomatoes user average score (normalized to a 0 to 5 point system)
         #Metacritic - Metacritic critics average score
         #Metacritic_User - Metacritic user average score
```

```
In [34]: import pandas as pd
```

```
fandango = pd.read_csv('fandango_score_comparison.csv')
series_film = fandango['FILM']
print(series_film[0:5])
series_rt = fandango['RottenTomatoes']
print (series_rt[0:5])
```

```
0    Avengers: Age of Ultron (2015)
1                 Cinderella (2015)
2                    Ant-Man (2015)
3            Do You Believe? (2015)
4     Hot Tub Time Machine 2 (2015)
Name: FILM, dtype: object
0    74
1    85
2    80
3    18
4    14
Name: RottenTomatoes, dtype: int64
```

```
In [35]: # Import the Series object from pandas
         from pandas import Series

         film_names = series_film.values
         print (type(film_names))
         print (film_names)
         rt_scores = series_rt.values
         print (rt_scores)
         series_custom = Series(rt_scores , index=film_names)
         series_custom[['Minions (2015)', 'Leviathan (2014)']]
```

```
<class 'numpy.ndarray'>
['Avengers: Age of Ultron (2015)' 'Cinderella (2015)' 'Ant-Man (2015)'
 'Do You Believe? (2015)' 'Hot Tub Time Machine 2 (2015)'
 'The Water Diviner (2015)' 'Irrational Man (2015)' 'Top Five (2014)'
 'Shaun the Sheep Movie (2015)' 'Love & Mercy (2015)'
 'Far From The Madding Crowd (2015)' 'Black Sea (2015)' 'Leviathan (2014)'
 'Unbroken (2014)' 'The Imitation Game (2014)' 'Taken 3 (2015)'
 'Ted 2 (2015)' 'Southpaw (2015)'
 'Night at the Museum: Secret of the Tomb (2014)' 'Pixels (2015)'
 'McFarland, USA (2015)' 'Insidious: Chapter 3 (2015)'
```

```
'The Man From U.N.C.L.E. (2015)' 'Run All Night (2015)'
'Trainwreck (2015)' 'Selma (2014)' 'Ex Machina (2015)'
'Still Alice (2015)' 'Wild Tales (2014)' 'The End of the Tour (2015)'
'Red Army (2015)' 'When Marnie Was There (2015)'
'The Hunting Ground (2015)' 'The Boy Next Door (2015)' 'Aloha (2015)'
'The Loft (2015)' '5 Flights Up (2015)' 'Welcome to Me (2015)'
'Saint Laurent (2015)' 'Maps to the Stars (2015)'
"I'll See You In My Dreams (2015)" 'Timbuktu (2015)' 'About Elly (2015)'
'The Diary of a Teenage Girl (2015)'
'Kingsman: The Secret Service (2015)' 'Tomorrowland (2015)'
'The Divergent Series: Insurgent (2015)' 'Annie (2014)'
'Fantastic Four (2015)' 'Terminator Genisys (2015)'
'Pitch Perfect 2 (2015)' 'Entourage (2015)' 'The Age of Adaline (2015)'
'Hot Pursuit (2015)' 'The DUFF (2015)' 'Black or White (2015)'
'Project Almanac (2015)' 'Ricki and the Flash (2015)'
'Seventh Son (2015)' 'Mortdecai (2015)' 'Unfinished Business (2015)'
'American Ultra (2015)' 'True Story (2015)' 'Child 44 (2015)'
'Dark Places (2015)' 'Birdman (2014)' 'The Gift (2015)'
'Unfriended (2015)' 'Monkey Kingdom (2015)' 'Mr. Turner (2014)'
'Seymour: An Introduction (2015)' 'The Wrecking Crew (2015)'
'American Sniper (2015)' 'Furious 7 (2015)'
'The Hobbit: The Battle of the Five Armies (2014)' 'San Andreas (2015)'
'Straight Outta Compton (2015)' 'Vacation (2015)' 'Chappie (2015)'
'Poltergeist (2015)' 'Paper Towns (2015)' 'Big Eyes (2014)'
'Blackhat (2015)' 'Self/less (2015)' 'Sinister 2 (2015)'
'Little Boy (2015)' 'Me and Earl and The Dying Girl (2015)'
'Maggie (2015)' 'Mad Max: Fury Road (2015)' 'Spy (2015)'
'The SpongeBob Movie: Sponge Out of Water (2015)' 'Paddington (2015)'
'Dope (2015)' 'What We Do in the Shadows (2015)' 'The Overnight (2015)'
'The Salt of the Earth (2015)' 'Song of the Sea (2014)'
'Fifty Shades of Grey (2015)' 'Get Hard (2015)' 'Focus (2015)'
'Jupiter Ascending (2015)' 'The Gallows (2015)'
'The Second Best Exotic Marigold Hotel (2015)' 'Strange Magic (2015)'
'The Gunman (2015)' 'Hitman: Agent 47 (2015)' 'Cake (2015)'
'The Vatican Tapes (2015)' 'A Little Chaos (2015)'
'The 100-Year-Old Man Who Climbed Out the Window and Disappeared (2015)'
'Escobar: Paradise Lost (2015)' 'Into the Woods (2014)'
'It Follows (2015)' 'Inherent Vice (2014)' 'A Most Violent Year (2014)'
"While We're Young (2015)" 'Clouds of Sils Maria (2015)'
```

```
 'Testament of Youth (2015)' 'Infinitely Polar Bear (2015)'
 'Phoenix (2015)' 'The Wolfpack (2015)'
 'The Stanford Prison Experiment (2015)' 'Tangerine (2015)'
 'Magic Mike XXL (2015)' 'Home (2015)' 'The Wedding Ringer (2015)'
 'Woman in Gold (2015)' 'The Last Five Years (2015)'
 'Mission: Impossible â“ Rogue Nation (2015)' 'Amy (2015)'
 'Jurassic World (2015)' 'Minions (2015)' 'Max (2015)'
 'Paul Blart: Mall Cop 2 (2015)' 'The Longest Ride (2015)'
 'The Lazarus Effect (2015)' 'The Woman In Black 2 Angel of Death (2015)'
 'Danny Collins (2015)' 'Spare Parts (2015)' 'Serena (2015)'
 'Inside Out (2015)' 'Mr. Holmes (2015)' "'71 (2015)"
 'Two Days, One Night (2014)' 'Gett: The Trial of Viviane Amsalem (2015)'
 'Kumiko, The Treasure Hunter (2015)']
[ 74  85  80  18  14  63  42  86  99  89  84  82  99  51  90   9  46  59
  50  17  79  59  68  60  85  99  92  88  96  92  96  89  92  10  19  11
  52  71  51  60  94  99  97  95  75  50  30  27   9  26  67  32  54   8
  71  39  34  64  12  12  11  46  45  26  26  92  93  60  94  98 100  93
  72  81  61  50  90  27  30  31  55  72  34  20  13  20  81  54  97  93
  78  98  87  96  82  96  99  25  29  57  26  16  62  17  17   7  49  13
  40  67  52  71  96  73  90  83  89  81  80  99  84  84  95  62  45  27
  52  60  92  97  71  54  35   5  31  14  22  77  52  18  98  87  97  97
 100  87]
```

```
Out[35]: Minions (2015)      54
         Leviathan (2014)    99
         dtype: int64
```

```
In [36]: # int index is also aviable
         series_custom = Series(rt_scores , index=film_names)
         series_custom[['Minions (2015)', 'Leviathan (2014)']]
         fiveten = series_custom[5:10]
         print(fiveten)
```

```
The Water Diviner (2015)        63
Irrational Man (2015)           42
Top Five (2014)                 86
Shaun the Sheep Movie (2015)    99
Love & Mercy (2015)             89
dtype: int64
```

```
In [37]: original_index = series_custom.index.tolist()
         print (original_index)
         sorted_index = sorted(original_index)
         sorted_by_index = series_custom.reindex(sorted_index)
         print (sorted_by_index)
```

```
['Avengers: Age of Ultron (2015)', 'Cinderella (2015)', 'Ant-Man (2015)', 'Do You Believe? (2015)'
'71 (2015)                                          97
5 Flights Up (2015)                                 52
A Little Chaos (2015)                               40
A Most Violent Year (2014)                          90
About Elly (2015)                                   97
Aloha (2015)                                        19
American Sniper (2015)                              72
American Ultra (2015)                               46
Amy (2015)                                          97
Annie (2014)                                        27
Ant-Man (2015)                                      80
Avengers: Age of Ultron (2015)                      74
Big Eyes (2014)                                     72
Birdman (2014)                                      92
Black Sea (2015)                                    82
Black or White (2015)                               39
Blackhat (2015)                                     34
Cake (2015)                                         49
Chappie (2015)                                      30
Child 44 (2015)                                     26
Cinderella (2015)                                   85
Clouds of Sils Maria (2015)                         89
Danny Collins (2015)                                77
Dark Places (2015)                                  26
Do You Believe? (2015)                              18
Dope (2015)                                         87
Entourage (2015)                                    32
Escobar: Paradise Lost (2015)                       52
Ex Machina (2015)                                   92
Fantastic Four (2015)                                9
                                                    ..
The Loft (2015)                                     11
The Longest Ride (2015)                             31
```

```
The Man From U.N.C.L.E. (2015)                          68
The Overnight (2015)                                    82
The Salt of the Earth (2015)                            96
The Second Best Exotic Marigold Hotel (2015)            62
The SpongeBob Movie: Sponge Out of Water (2015)         78
The Stanford Prison Experiment (2015)                   84
The Vatican Tapes (2015)                                13
The Water Diviner (2015)                                63
The Wedding Ringer (2015)                               27
The Wolfpack (2015)                                     84
The Woman In Black 2 Angel of Death (2015)              22
The Wrecking Crew (2015)                                93
Timbuktu (2015)                                         99
Tomorrowland (2015)                                     50
Top Five (2014)                                         86
Trainwreck (2015)                                       85
True Story (2015)                                       45
Two Days, One Night (2014)                              97
Unbroken (2014)                                         51
Unfinished Business (2015)                              11
Unfriended (2015)                                       60
Vacation (2015)                                         27
Welcome to Me (2015)                                    71
What We Do in the Shadows (2015)                        96
When Marnie Was There (2015)                            89
While We're Young (2015)                                83
Wild Tales (2014)                                       96
Woman in Gold (2015)                                    52
Length: 146, dtype: int64
```

```
In [38]: sc2 = series_custom.sort_index()
         sc3 = series_custom.sort_values()
         #print(sc2[0:10])
         print(sc3[0:10])
```

```
Paul Blart: Mall Cop 2 (2015)      5
Hitman: Agent 47 (2015)            7
Hot Pursuit (2015)                 8
Fantastic Four (2015)              9
Taken 3 (2015)                     9
```

```
The Boy Next Door (2015)         10
The Loft (2015)                  11
Unfinished Business (2015)       11
Mortdecai (2015)                 12
Seventh Son (2015)               12
dtype: int64
```

```
In [39]: #The values in a Series object are treated as an ndarray, the core data type in NumPy
         import numpy as np
         # Add each value with each other
         print (np.add(series_custom, series_custom))
         # Apply sine function to each value
         np.sin(series_custom)
         # Return the highest value (will return a single value not a Series)
         np.max(series_custom)
```

```
Avengers: Age of Ultron (2015)                   148
Cinderella (2015)                                170
Ant-Man (2015)                                   160
Do You Believe? (2015)                            36
Hot Tub Time Machine 2 (2015)                     28
The Water Diviner (2015)                         126
Irrational Man (2015)                             84
Top Five (2014)                                  172
Shaun the Sheep Movie (2015)                     198
Love & Mercy (2015)                              178
Far From The Madding Crowd (2015)                168
Black Sea (2015)                                 164
Leviathan (2014)                                 198
Unbroken (2014)                                  102
The Imitation Game (2014)                        180
Taken 3 (2015)                                    18
Ted 2 (2015)                                      92
Southpaw (2015)                                  118
Night at the Museum: Secret of the Tomb (2014)   100
Pixels (2015)                                     34
McFarland, USA (2015)                            158
Insidious: Chapter 3 (2015)                      118
The Man From U.N.C.L.E. (2015)                   136
Run All Night (2015)                             120
```

```
Trainwreck (2015)                               170
Selma (2014)                                    198
Ex Machina (2015)                               184
Still Alice (2015)                              176
Wild Tales (2014)                               192
The End of the Tour (2015)                      184
                                               ...
Clouds of Sils Maria (2015)                     178
Testament of Youth (2015)                       162
Infinitely Polar Bear (2015)                    160
Phoenix (2015)                                  198
The Wolfpack (2015)                             168
The Stanford Prison Experiment (2015)           168
Tangerine (2015)                                190
Magic Mike XXL (2015)                           124
Home (2015)                                      90
The Wedding Ringer (2015)                        54
Woman in Gold (2015)                            104
The Last Five Years (2015)                      120
Mission: Impossible â“ Rogue Nation (2015)       184
Amy (2015)                                      194
Jurassic World (2015)                           142
Minions (2015)                                  108
Max (2015)                                       70
Paul Blart: Mall Cop 2 (2015)                    10
The Longest Ride (2015)                          62
The Lazarus Effect (2015)                        28
The Woman In Black 2 Angel of Death (2015)       44
Danny Collins (2015)                            154
Spare Parts (2015)                              104
Serena (2015)                                    36
Inside Out (2015)                               196
Mr. Holmes (2015)                               174
'71 (2015)                                      194
Two Days, One Night (2014)                      194
Gett: The Trial of Viviane Amsalem (2015)       200
Kumiko, The Treasure Hunter (2015)              174
Length: 146, dtype: int64
```

Out[39]: 100

In [40]:
```
#will actually return a Series object with a boolean value for each film
series_custom > 50
series_greater_than_50 = series_custom[series_custom > 50]

criteria_one = series_custom > 50
criteria_two = series_custom < 75
both_criteria = series_custom[criteria_one & criteria_two]
print (both_criteria)
```

```
Avengers: Age of Ultron (2015)                                          74
The Water Diviner (2015)                                                63
Unbroken (2014)                                                         51
Southpaw (2015)                                                         59
Insidious: Chapter 3 (2015)                                             59
The Man From U.N.C.L.E. (2015)                                          68
Run All Night (2015)                                                    60
5 Flights Up (2015)                                                     52
Welcome to Me (2015)                                                    71
Saint Laurent (2015)                                                    51
Maps to the Stars (2015)                                                60
Pitch Perfect 2 (2015)                                                  67
The Age of Adaline (2015)                                               54
The DUFF (2015)                                                         71
Ricki and the Flash (2015)                                              64
Unfriended (2015)                                                       60
American Sniper (2015)                                                  72
The Hobbit: The Battle of the Five Armies (2014)                        61
Paper Towns (2015)                                                      55
Big Eyes (2014)                                                         72
Maggie (2015)                                                           54
Focus (2015)                                                            57
The Second Best Exotic Marigold Hotel (2015)                            62
The 100-Year-Old Man Who Climbed Out the Window and Disappeared (2015)    67
Escobar: Paradise Lost (2015)                                           52
Into the Woods (2014)                                                   71
Inherent Vice (2014)                                                    73
Magic Mike XXL (2015)                                                   62
Woman in Gold (2015)                                                    52
The Last Five Years (2015)                                              60
```

```
Jurassic World (2015)                                                      71
Minions (2015)                                                             54
Spare Parts (2015)                                                         52
dtype: int64
```

```
In [41]: #data alignment same index
         rt_critics = Series(fandango['RottenTomatoes'].values, index=fandango['FILM'])
         rt_users = Series(fandango['RottenTomatoes_User'].values, index=fandango['FILM'])
         rt_mean = (rt_critics + rt_users)/2

         print(rt_mean)
```

```
FILM
Avengers: Age of Ultron (2015)                    80.0
Cinderella (2015)                                 82.5
Ant-Man (2015)                                    85.0
Do You Believe? (2015)                            51.0
Hot Tub Time Machine 2 (2015)                     21.0
The Water Diviner (2015)                          62.5
Irrational Man (2015)                             47.5
Top Five (2014)                                   75.0
Shaun the Sheep Movie (2015)                      90.5
Love & Mercy (2015)                               88.0
Far From The Madding Crowd (2015)                 80.5
Black Sea (2015)                                  71.0
Leviathan (2014)                                  89.0
Unbroken (2014)                                   60.5
The Imitation Game (2014)                         91.0
Taken 3 (2015)                                    27.5
Ted 2 (2015)                                      52.0
Southpaw (2015)                                   69.5
Night at the Museum: Secret of the Tomb (2014)    54.0
Pixels (2015)                                     35.5
McFarland, USA (2015)                             84.0
Insidious: Chapter 3 (2015)                       57.5
The Man From U.N.C.L.E. (2015)                    74.0
Run All Night (2015)                              59.5
Trainwreck (2015)                                 79.5
Selma (2014)                                      92.5
Ex Machina (2015)                                 89.0
```

```
Still Alice (2015)                                86.5
Wild Tales (2014)                                 94.0
The End of the Tour (2015)                        90.5
                                                  ...
Clouds of Sils Maria (2015)                       78.0
Testament of Youth (2015)                         80.0
Infinitely Polar Bear (2015)                      78.0
Phoenix (2015)                                    90.0
The Wolfpack (2015)                               78.5
The Stanford Prison Experiment (2015)             85.5
Tangerine (2015)                                  90.5
Magic Mike XXL (2015)                             63.0
Home (2015)                                       55.0
The Wedding Ringer (2015)                         46.5
Woman in Gold (2015)                              66.5
The Last Five Years (2015)                        60.0
Mission: Impossible â “ Rogue Nation (2015)        91.0
Amy (2015)                                        94.0
Jurassic World (2015)                             76.0
Minions (2015)                                    53.0
Max (2015)                                        54.0
Paul Blart: Mall Cop 2 (2015)                     20.5
The Longest Ride (2015)                           52.0
The Lazarus Effect (2015)                         18.5
The Woman In Black 2 Angel of Death (2015)        23.5
Danny Collins (2015)                              76.0
Spare Parts (2015)                                67.5
Serena (2015)                                     21.5
Inside Out (2015)                                 94.0
Mr. Holmes (2015)                                 82.5
'71 (2015)                                        89.5
Two Days, One Night (2014)                        87.5
Gett: The Trial of Viviane Amsalem (2015)         90.5
Kumiko, The Treasure Hunter (2015)                75.0
Length: 146, dtype: float64
```

```
In [42]: import pandas as pd

         #will return a new DataFrame that is indexed by the values in the specified column
```

```
#and will drop that column from the DataFrame
#without the FILM column dropped
fandango = pd.read_csv('fandango_score_comparison.csv')
print (type(fandango))
fandango_films = fandango.set_index('FILM', drop=False)
print(fandango_films.index)
```

```
<class 'pandas.core.frame.DataFrame'>
Index(['Avengers: Age of Ultron (2015)', 'Cinderella (2015)', 'Ant-Man (2015)',
       'Do You Believe? (2015)', 'Hot Tub Time Machine 2 (2015)',
       'The Water Diviner (2015)', 'Irrational Man (2015)', 'Top Five (2014)',
       'Shaun the Sheep Movie (2015)', 'Love & Mercy (2015)',
       ...
       'The Woman In Black 2 Angel of Death (2015)', 'Danny Collins (2015)',
       'Spare Parts (2015)', 'Serena (2015)', 'Inside Out (2015)',
       'Mr. Holmes (2015)', ''71 (2015)', 'Two Days, One Night (2014)',
       'Gett: The Trial of Viviane Amsalem (2015)',
       'Kumiko, The Treasure Hunter (2015)'],
      dtype='object', name='FILM', length=146)
```

```
In [43]: # Slice using either bracket notation or loc[]
         fandango_films["Avengers: Age of Ultron (2015)":"Hot Tub Time Machine 2 (2015)"]
         fandango_films.loc["Avengers: Age of Ultron (2015)":"Hot Tub Time Machine 2 (2015)"]

         # Specific movie
         fandango_films.loc['Kumiko, The Treasure Hunter (2015)']

         # Selecting list of movies
         movies = ['Kumiko, The Treasure Hunter (2015)', 'Do You Believe? (2015)', 'Ant-Man (2015)
         fandango_films.loc[movies]

         #When selecting multiple rows, a DataFrame is returned,
         #but when selecting an individual row, a Series object is returned instead
```

```
Out[43]:                                                                   FILM  \
         FILM
         Kumiko, The Treasure Hunter (2015)  Kumiko, The Treasure Hunter (2015)
         Do You Believe? (2015)                          Do You Believe? (2015)
         Ant-Man (2015)                                          Ant-Man (2015)
```

```
                                   RottenTomatoes  RottenTomatoes_User  \
FILM
Kumiko, The Treasure Hunter (2015)             87                   63
Do You Believe? (2015)                         18                   84
Ant-Man (2015)                                 80                   90

                                   Metacritic  Metacritic_User  IMDB  \
FILM
Kumiko, The Treasure Hunter (2015)         68              6.4   6.7
Do You Believe? (2015)                     22              4.7   5.4
Ant-Man (2015)                             64              8.1   7.8

                                   Fandango_Stars  Fandango_Ratingvalue  \
FILM
Kumiko, The Treasure Hunter (2015)            3.5                   3.5
Do You Believe? (2015)                        5.0                   4.5
Ant-Man (2015)                                5.0                   4.5

                                   RT_norm  RT_user_norm  \
FILM
Kumiko, The Treasure Hunter (2015)     4.35          3.15
Do You Believe? (2015)                 0.90          4.20
Ant-Man (2015)                         4.00          4.50

                                           ...          IMDB_norm  \
FILM                                       ...
Kumiko, The Treasure Hunter (2015)         ...               3.35
Do You Believe? (2015)                     ...               2.70
Ant-Man (2015)                             ...               3.90

                                   RT_norm_round  RT_user_norm_round  \
FILM
Kumiko, The Treasure Hunter (2015)            4.5                 3.0
Do You Believe? (2015)                        1.0                 4.0
Ant-Man (2015)                                4.0                 4.5

                                   Metacritic_norm_round  \
FILM
Kumiko, The Treasure Hunter (2015)                    3.5
```

```
Do You Believe? (2015)                                   1.0
Ant-Man (2015)                                           3.0

                                    Metacritic_user_norm_round  \
FILM
Kumiko, The Treasure Hunter (2015)                         3.0
Do You Believe? (2015)                                     2.5
Ant-Man (2015)                                             4.0

                                    IMDB_norm_round  \
FILM
Kumiko, The Treasure Hunter (2015)              3.5
Do You Believe? (2015)                          2.5
Ant-Man (2015)                                  4.0

                                    Metacritic_user_vote_count  \
FILM
Kumiko, The Treasure Hunter (2015)                          19
Do You Believe? (2015)                                      31
Ant-Man (2015)                                             627

                                    IMDB_user_vote_count  Fandango_votes  \
FILM
Kumiko, The Treasure Hunter (2015)                  5289              41
Do You Believe? (2015)                              3136            1793
Ant-Man (2015)                                    103660           12055

                                    Fandango_Difference
FILM
Kumiko, The Treasure Hunter (2015)                  0.0
Do You Believe? (2015)                              0.5
Ant-Man (2015)                                      0.5

[3 rows x 22 columns]
```

In [44]:
```
#The apply() method in Pandas allows us to specify Python logic
#The apply() method requires you to pass in a vectorized operation
#that can be applied over each Series object.
import numpy as np
```

```
# returns the data types as a Series
types = fandango_films.dtypes
#print types
# filter data types to just floats, index attributes returns just column names
float_columns = types[types.values == 'float64'].index
# use bracket notation to filter columns to just float columns
float_df = fandango_films[float_columns]
#print float_df
# `x` is a Series object representing a column
deviations = float_df.apply(lambda x: np.std(x))

print(deviations)
```

```
Metacritic_User               1.505529
IMDB                          0.955447
Fandango_Stars                0.538532
Fandango_Ratingvalue          0.501106
RT_norm                       1.503265
RT_user_norm                  0.997787
Metacritic_norm               0.972522
Metacritic_user_nom           0.752765
IMDB_norm                     0.477723
RT_norm_round                 1.509404
RT_user_norm_round            1.003559
Metacritic_norm_round         0.987561
Metacritic_user_norm_round    0.785412
IMDB_norm_round               0.501043
Fandango_Difference           0.152141
dtype: float64
```

```
In [45]: rt_mt_user = float_df[['RT_user_norm', 'Metacritic_user_nom']]
         rt_mt_user.apply(lambda x: np.std(x), axis=1)
```

```
Out[45]: FILM
         Avengers: Age of Ultron (2015)                   0.375
         Cinderella (2015)                                0.125
         Ant-Man (2015)                                   0.225
         Do You Believe? (2015)                           0.925
         Hot Tub Time Machine 2 (2015)                    0.150
         The Water Diviner (2015)                         0.150
```

```
Irrational Man (2015)                              0.575
Top Five (2014)                                    0.100
Shaun the Sheep Movie (2015)                       0.150
Love & Mercy (2015)                                0.050
Far From The Madding Crowd (2015)                  0.050
Black Sea (2015)                                   0.150
Leviathan (2014)                                   0.175
Unbroken (2014)                                    0.125
The Imitation Game (2014)                          0.250
Taken 3 (2015)                                     0.000
Ted 2 (2015)                                       0.175
Southpaw (2015)                                    0.050
Night at the Museum: Secret of the Tomb (2014)     0.000
Pixels (2015)                                      0.025
McFarland, USA (2015)                              0.425
Insidious: Chapter 3 (2015)                        0.325
The Man From U.N.C.L.E. (2015)                     0.025
Run All Night (2015)                               0.350
Trainwreck (2015)                                  0.350
Selma (2014)                                       0.375
Ex Machina (2015)                                  0.175
Still Alice (2015)                                 0.175
Wild Tales (2014)                                  0.100
The End of the Tour (2015)                         0.350
                                                    ...
Clouds of Sils Maria (2015)                        0.100
Testament of Youth (2015)                          0.000
Infinitely Polar Bear (2015)                       0.075
Phoenix (2015)                                     0.025
The Wolfpack (2015)                                0.075
The Stanford Prison Experiment (2015)              0.050
Tangerine (2015)                                   0.325
Magic Mike XXL (2015)                              0.250
Home (2015)                                        0.200
The Wedding Ringer (2015)                          0.825
Woman in Gold (2015)                               0.225
The Last Five Years (2015)                         0.225
Mission: Impossible â“ Rogue Nation (2015)         0.250
Amy (2015)                                         0.075
```

```
Jurassic World (2015)                            0.275
Minions (2015)                                   0.125
Max (2015)                                       0.350
Paul Blart: Mall Cop 2 (2015)                    0.300
The Longest Ride (2015)                          0.625
The Lazarus Effect (2015)                        0.650
The Woman In Black 2 Angel of Death (2015)       0.475
Danny Collins (2015)                             0.100
Spare Parts (2015)                               0.300
Serena (2015)                                    0.700
Inside Out (2015)                                0.025
Mr. Holmes (2015)                                0.025
'71 (2015)                                       0.175
Two Days, One Night (2014)                       0.250
Gett: The Trial of Viviane Amsalem (2015)        0.200
Kumiko, The Treasure Hunter (2015)               0.025
Length: 146, dtype: float64
```

```
In [ ]:
```