Open in Colab

Session 5 Sorting

bold text###Sorting

```
1 import numpy as np
2 import pandas as pd
```

```
1 df = pd.read_csv("FiveYearData.csv")
2 df
```

```
      country  year        pop  continent  lifeExp   gdpPercap
```

Double-click (or enter) to edit

```
1 df.describe()
2 df.info()
3 df.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1704 entries, 0 to 1703
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   country    1704 non-null   object 
 1   year       1704 non-null   int64  
 2   pop        1704 non-null   float64
 3   continent  1704 non-null   object 
 4   lifeExp    1704 non-null   float64
 5   gdpPercap  1704 non-null   float64
dtypes: float64(3), int64(1), object(2)
memory usage: 80.0+ KB
(1704, 6)
```

pandas.DataFrame.sort_values

DataFrame.sort_values(by, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last', ignore_index=False, key=None)

Sort by the values along either axis.

```
1 df.sort_values('lifeExp', inplace=True)
2 df
```

	country	year	pop	continent	lifeExp	gdpPercap
1292	Rwanda	1992	7290203.0	Africa	23.599	737.068595
0	Afghanistan	1952	8425333.0	Asia	28.801	779.445314
552	Gambia	1952	284320.0	Africa	30.000	485.230659
36	Angola	1952	4232095.0	Africa	30.015	3520.610273
1344	Sierra Leone	1952	2143249.0	Africa	30.331	879.787736
...	...	...	...	...	...	...
1487	Switzerland	2007	7554661.0	Europe	81.701	37506.419070
695	Iceland	2007	301931.0	Europe	81.757	36180.789190
802	Japan	2002	127065841.0	Asia	82.000	28604.591900
671	Hong Kong China	2007	6980412.0	Asia	82.208	39724.978670

```
1 df.sort_values('lifeExp', inplace=True, ascending=False)
2 df
```

	country	year	pop	continent	lifeExp	gdpPercap
803	Japan	2007	127467972.0	Asia	82.603	31656.068060
671	Hong Kong China	2007	6980412.0	Asia	82.208	39724.978670

pandas.DataFrame.sort_index

DataFrame.sort_index(axis=0, level=None, ascending=True, inplace=False, kind='quicksort', na_position='last', sort_remaining=True, ignore_index=False, key=None)

Sort object by labels (along an axis).

Returns a new DataFrame sorted by label if inplace argument is False, otherwise updates the original DataFrame and returns None.

0	Afghanistan	1952	8425333.0	Asia	28.801	779.445314

```
df.sort_values('lifeExp', inplace=True, na_position="first")
df
```

	country	year	pop	continent	lifeExp	gdpPercap
1292	Rwanda	1992	7290203.0	Africa	23.599	737.068595
0	Afghanistan	1952	8425333.0	Asia	28.801	779.445314

```
1 df.sort_index(inplace=True)
2 df
```

	country	year	pop	continent	lifeExp	gdpPercap
0	Afghanistan	1952	8425333.0	Asia	28.801	779.445314
1	Afghanistan	1957	9240934.0	Asia	30.332	820.853030
2	Afghanistan	1962	10267083.0	Asia	31.997	853.100710
3	Afghanistan	1967	11537966.0	Asia	34.020	836.197138
4	Afghanistan	1972	13079460.0	Asia	36.088	739.981106
...	...	...	...	...	...	...
1699	Zimbabwe	1987	9216418.0	Africa	62.351	706.157306
1700	Zimbabwe	1992	10704340.0	Africa	60.377	693.420786
1701	Zimbabwe	1997	11404948.0	Africa	46.809	792.449960
1702	Zimbabwe	2002	11926563.0	Africa	39.989	672.038623
1703	Zimbabwe	2007	12311143.0	Africa	43.487	469.709298

1704 rows × 6 columns

pandas.DataFrame.reindex

DataFrame.reindex(labels=None, index=None, columns=None, axis=None, method=None, copy=True, level=None, fill_value=nan, limit=None, tolerance=None)[source] **Conform Series/DataFrame to new index with optional filling logic.**

Places NA/NaN in locations having no value in the previous index. A new object is produced unless the new index is equivalent to the current one and copy=False.

```
1 df.reindex()
```

	country	year	pop	continent	lifeExp	gdpPercap
0	Afghanistan	1952	8425333.0	Asia	28.801	779.445314
1	Afghanistan	1957	9240934.0	Asia	30.332	820.853030
2	Afghanistan	1962	10267083.0	Asia	31.997	853.100710
3	Afghanistan	1967	11537966.0	Asia	34.020	836.197138
4	Afghanistan	1972	13079460.0	Asia	36.088	739.981106
...	...	...	...	...	...	...
1699	Zimbabwe	1987	9216418.0	Africa	62.351	706.157306
1700	Zimbabwe	1992	10704340.0	Africa	60.377	693.420786
1701	Zimbabwe	1997	11404948.0	Africa	46.809	792.449960
1702	Zimbabwe	2002	11926563.0	Africa	39.989	672.038623
1703	Zimbabwe	2007	12311143.0	Africa	43.487	469.709298

1704 rows × 6 columns

```
1 tr=df.sort_values(['lifeExp', 'gdpPercap'], ascending = [True,False])
2 tr
```

	country	year	pop	continent	lifeExp	gdpPercap
1292	Rwanda	1992	7290203.0	Africa	23.599	737.068595
0	Afghanistan	1952	8425333.0	Asia	28.801	779.445314
552	Gambia	1952	284320.0	Africa	30.000	485.230659
36	Angola	1952	4232095.0	Africa	30.015	3520.610273
1344	Sierra Leone	1952	2143249.0	Africa	30.331	879.787736
...	...	...	...	...	...	...
1487	Switzerland	2007	7554661.0	Europe	81.701	37506.419070
695	Iceland	2007	301931.0	Europe	81.757	36180.789190
802	Japan	2002	127065841.0	Asia	82.000	28604.591900

```
1 tr = df.round(2)
2 tr = df.round({'lifeExp': 2, 'gdpPercap': 1})
3 tr
```

	country	year	pop	continent	lifeExp	gdpPercap
0	Afghanistan	1952	8425333.0	Asia	28.80	779.4
1	Afghanistan	1957	9240934.0	Asia	30.33	820.9
3	Afghanistan	1967	11537966.0	Asia	34.02	836.2

N-Largest

Pandas **nlargest()** method is used to get n largest values from a data frame or a series.

```
df = pd.DataFrame({'population': [58812491, 37522972, 339757,
                                  339757, 339757, 38964, 11646, 11646, 11646],
                   'GDP': [2033594, 1833594 , 153594, 14520, 10128,
                           836, 139, 39, 539],
                   'alpha-2': ["SA", "CA", "AU", "MV", "IS",
                               "MC", "NU", "TV", "PW"]},
                  index=["South Africa", "Canada", "Australia",
                         "Maldives", "Iceland", "Monaco", "Niue",
                         "Tuvalu", "Palau"])
df
```

```
1 df.nlargest(3, 'population', keep='last')
```

	population	GDP	alpha-2
South Africa	58812491	2033594	SA
Canada	37522972	1833594	CA
Iceland	339757	10128	IS
Monaco	38964	836	MC

```
1 df.nlargest(3, 'population', keep='all')
```

	population	GDP	alpha-2
South Africa	58812491	2033594	SA
Canada	37522972	1833594	CA
Australia	339757	153594	AU
Maldives	339757	14520	MV
Iceland	339757	10128	IS

Pandas nsmallest() method is used to get n least values from a data frame or a series.

```
1 df.nsmallest(5, 'population')
```

	population	GDP	alpha-2

```
df.nsmallest(5, 'population', keep='last')
```

	population	GDP	alpha-2
Palau	11646	539	PW
Tuvalu	11646	39	TV
Niue	11646	139	NU
Monaco	38964	836	MC
Iceland	339757	10128	IS

```
df.nsmallest(5, 'population', keep='all')
```

	population	GDP	alpha-2
Niue	11646	139	NU
Tuvalu	11646	39	TV
Palau	11646	539	PW
Monaco	38964	836	MC
Australia	339757	153594	AU
Maldives	339757	14520	MV
Iceland	339757	10128	IS

```
df = pd.read_csv("winequality-red.csv")
df
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.9978
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.9968
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.9970
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.9980
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.9978
...	...	...	...	...	...	...	...	..
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.9949
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.9951
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.9957
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.9954
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.9954

1599 rows × 12 columns

```
1 df.iloc[:,[3,4,6]].head()
```

	residual sugar	chlorides	total sulfur dioxide
0	1.9	0.076	34.0
1	2.6	0.098	67.0
2	2.3	0.092	54.0
3	1.9	0.075	60.0
4	1.9	0.076	34.0

```
1 df.iloc[[5,9],[7,9]]
```

	density	sulphates
5	0.9978	0.56
9	0.9978	0.80

```
1 df[df['density'] == 0.998].info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 29 entries, 3 to 905
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   fixed acidity         29 non-null     float64
 1   volatile acidity      29 non-null     float64
 2   citric acid           29 non-null     float64
 3   residual sugar        29 non-null     float64
 4   chlorides             29 non-null     float64
 5   free sulfur dioxide   29 non-null     float64
 6   total sulfur dioxide  29 non-null     float64
 7   density               29 non-null     float64
 8   pH                    29 non-null     float64
 9   sulphates             29 non-null     float64
 10  alcohol               29 non-null     float64
 11  quality               29 non-null     int64  
dtypes: float64(11), int64(1)
memory usage: 2.9 KB
```

```
1 df[df['fixed acidity']>5 | (df['residual sugar']<2)]['density'].unique().sum()
```

```
434.33755
```

```
1
```

✓ 0s completed at 12:08 PM