□□□□

□□□□□□□□

```
import panda as pd
## □□csv□tsv□txt□□
csv = pd.read_csv('path')

## □□excel□□
excel = pd.read_excel('path')

## □□sql□□
sql = pd.read_sql('path')

## □□csv□□□index_col□0□□□□□□□□□□□□
reviews = pd.read_csv('../input/wine-reviews/winemag-data_first150k.csv',index_col =
0)
```

□□**txt**□□□□□□□□□□□□□□□

```
txt = pd.read_csv(
    'path'
    # □□□□□□□
    sep = '\t'
    # □□□□
    header = None
    # □□□□□
    names = ['pdate','pv','uv']
)
```

□□□□

```
## □□□□□□□□
csv.head()
## □□□□□□□
csv.shape
## □□□□□□□
csv.colums
## □□□□□
csv.index
## □□□□□□□□□□
csv.dtypes
```

□□□□

to_csv□□

```
## □animals□□□csv
animals = pd.DataFrame({'Cows': [12, 20], 'Goats': [22, 19]}, index=['Year 1', 'Year
```

```
2'])
animals.to_csv('cows_and_goats.csv')
```

□□□□

- `Series`:□□□□□□□□□□

```
## □□□Series□index
s1 = pd.Series([1,'a',5.2,7],index=['d','b','a','c'])
ingredients = pd.Series(['4 cups','1 cup','2 large','1 can'],index =
['Flour','Milk','Eggs','Spam'],name = 'Dinner')
## □□□□
s1.index

## □□□
s1.values

## □□index□□values
print(s1['a'])
print(type(s1['a']))
```

- `DataFrame`:□□□□□□□□□□

```
## □□□□□
df.dtypes
## □□□□□
df.index
## □□□□□
df.colums

## □□□□□□□□□Series
df['year']
## □□□□□□□□□DataFrame
df[['year','pop']]

## □□□□□
df.loc[1]
## □□□□(□□□□□□□□python□□□□□)
df.locp[1:3]
```

□□DataFrame

```
import pandas as pd
fruits = pd.DataFrame({'Apple':[30],'Bananas':[21]})
```

□□□□

□□□□

```
## □□□□□□□□□□
reviews.dtypes
## □□□□□□□
reviews.points.astype('float64')
```

□□□□

- `drop(colums=['□□'])`

```
## □□□□id□□
data = data.drop(colums=['□□id'])
```

- `drop_duplicates(subset = ['□1','□2',...],keep = 'first')`
  - □□subset□□□□□□
  - keep□□□□□□□□□□□□□□□□□□□□□□□□□□□□
    - first□□□□□□□□□□□□□□□
    - last□□□□□□□□□□□□□□□
    - False□□□□□□□□□

```
df = df.drop_duplicates(subset=['□□□id', '□□□□', '□□DeptId'], keep='first')
```

□□□□

□□□□□□

==.iloc□□□□□□□□□==

==.loc□□□□□□□□□==

```
## □□□□□□□□□□□□□
df.loc
## □□□□□□□□□□□□□
df.iloc
```

`.loc` □□□□□□□

```
## □□□□label□□□□□□
df.loc['2018-01-03','bWendu']
## □□□□□□□□□
df.loc[['2018-01-03','2018-01-04','2018-01-05'],['bWemdu','yWendu']]
## □□□□□□□□□□
df.loc['2018-01-03':'2018-01-05','bWendu']
## □□□□□□□□□
df.loc[df['yWendu']<-10,:]
```

□□□□□□□

```
## □□□□□□□
reviews.country
## □□□□□□□□□
reviews['country']
## □□□□□□
reviews['country'][0]
```

□□□□

```
## isin□□□□country□Italy□France□□
reviews.loc[reviews.country.isin(['Italy', 'France'])] # □□□□□□□

## .idxmax()□□□□□□□□□□□□□□□
(reviews.points / reviews.price).idxmax()
```

□□□□□

□□□□

```
## □□□□□
df.loc[:,'Wencha'] = df['bWendu'] - df['yWendu']
## □□apply□□
def get_wendu_type(x):
    if x['bWendu'] > 33:
        return '□□'
    if x['bWendu'] < -10:
        return '□□'
    return '□□'
df.loc[:,'wendu_type'] = df.apply(get_wendu_type,axis = 1)
## assign□□□□□□□□□□□□□□□□□□□□□□□□
df.assign(
    yWendu_huashi = lambda x : x['yWendu'] * 9 / 5 + 32.
    bWendu_huashi = lambda x : x['bWendu'] * 9 / 5 + 32
)
```

□□□

```
## □□title□
reviews.set_index('title')
```

Pandas□□□□□□

1. □□□□□

```
## □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
df.describe()
## □□□
```

```
df['bWendu'].mean()
## □□□
df['bWendu'].max()
## □□□
df['bWendu'].median()
```

2. □□□□□□□□□□

```
## □□□□□□□(□□)
df['fengxiang'].unique()
## □□□□□□□□□
df['fengxiang'].value_counts()
```

3. □□□□□□□□□

```
## □□□□□:□□□□□□□□
df.cov()
## □□□□□□□:□□□□□□□
df.corr()
## □□□□□□□□□□□□□□
df['api'].corr(df['bWendu'])
```

Pandas□□□□□□□

□□□□□□□

```
## □□□□DataFrame□□□□□
studf.isnull()
## □□□□□□□□□□□
studf['□□'].isnull()
## □isnulll□□
studf.notnull()
studf['□□'].notnull()
```

□□□□□□□□□

`dropna` □□□□□□□□□□

- axis□□□□□□□
- how□□any□□□□□□□□□□□□□all□□□□□□□□□□□
- inplace□□True□□□□□df□□□□□□□df

```
## □□□□□□□□□
studf.dropna(axis = 1,how = 'all',inplace = True)
## □□□□□□□□□□□
studf.dropna(axis = 0,how = 'any',inplace = True)
```

□□□□

`fillna`:□□□□

- value□□□□□□□□□□□□□□□□□□□□□□
- method□□□□□□,ffill□□□□□□□□□□□□□□□□□□bfill□□□□□□□□□□□□□□□□
- axis□□□□□□□□□
- inplace□□True□□□□df□□□□□□□□df

```
## □□□□□□□□□□□□0□
studf.fillna({'□□':0})
## □□□
studf.loc[:,'□□'] = studf.fillna(0)
## □□□□□□□□□□
studf.loc[:,'□□'] = studf['□□'].fillna(method = 'ffill')
```

□□

```
## □replace□□□taster_twitter_handle□□@kerinokeefe□□□@kerino
reviews.taster_twitter_handle.replace("@kerinokeefe", "@kerino")
```

□□□□□□

□□□

rename

- index□columns□□□□□□□
- □□□□□□□
- inplace□□True□□□□DataFrame□□□□□

```
## □□points□□score
reviews.rename(columns={'points': 'score'})
```

rename_axis

- □□□□□□□name□□
- name□name□□□□
- rows□columns□□□□□□□□□□□□□□□

```
## □□□□□□name□□'wines'□□□□□□□name□□'fields'
reviews.rename_axis('wines',axis = 'rows').rename_axis('fields',axis = 'columns')
```

□□

`concat` □□: `pandas.concat(objs,axis = 0, join = 'outer',ignore_index = False)`

- `objs` □□□□□□□□□□□□ `DataFrame` □□ `Series`
- `axis` □0□□□□□□1□□□□□
- `join` □□□□□□□□□□□□□□□outer,inner□□□□□□□□□

- `ignore_index` □□□□□□□□□□□□□□□\
- concat□□□□□□□□□□□□□□□□□□□□□□□□□□□□□NaN

```
## □□
pd.concat([df1,df2])
## □□□□
pd.concat([df1,df2],axis = 1)
```

join□□

- □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
- ** `other` **□□□□□□□□□ DataFrame□
- ** `how` **□□□□□□□□□□□□□□□□□□□
  - `'left'` □□□□□□□ DataFrame □□□□□□□□ DataFrame □□□ NaN□
  - `'right'` □□□□□□□ DataFrame □□□□□□□□ DataFrame □□□ NaN□
  - `'outer'` □□□□□ DataFrame □□□□□□□□□□□□□□□□ NaN□
  - `'inner'` □□□□□□□□□□□□□
- ** `on` **□□□□□□□□□□□□□□□□□□□□□□□□□□□
- ** `lsuffix` **□□□□ DataFrame □□□□□□□□□□□□ DataFrame □□□□□□□□
- ** `rsuffix` **□□□□ DataFrame □□□□□□□□□□□□ DataFrame □□□□□□□□
- ** `sort` **□□□□□□□□□□□□□□□□□□□□□ False□

```
DataFrame.join(other, how='left', on=None, lsuffix='', rsuffix='', sort=False)
## □□□□DataFrame□MeetID□□□□
powerlifting_combined =
owerlifting_meets.set_index('MeetID').join(powerlifting_competitors.set_index('MeetID'
```

□□

map

```
## □□points□□□□□□□□map□□lambda□□□□□p□□□reviews□□
review_points_mean = reviews.points.mean()
reviews.points.map(lambda p: p - review_points_mean)
## □□description□□tropical□fruity□□□□□
descriptor_counts = pd.Series([reviews.description.map(lambda x : 'tropical' in
x).sum(),reviews.description.map(lambda y: 'fruity' in y).sum()],index =
['tropical','fruity'])
```

apply

- axis□colums□□

```
## □□□□□
def remean_points(row):
    row.points = row.points - review_points_mean
    return row

reviews.apply(remean_points, axis='columns')
```

```
## □□□□□□□row□□country□Canada□□□□□3□□□□□□□□□□□□□□□
def method(row):
    if row.country == 'Canada':
        return 3
    elif row.points >= 95:
        return 3
    elif row.points >= 85:
        return 2
    return 1

star_ratings = reviews.apply(method,axis = 'columns')

## Check your answer
q7.check()
```

□□□□□□□□

□□

groupby□□

```
## □□points□□□□□□□points□□□□□□□□□□□□points□□□□□□□□□□
reviews.groupby('points').points.sum()
## □□□□□□□□□□□□.apply()□□
reviews.groupby(['country', 'province']).apply(lambda df:
df.loc[df.points.idxmax()])


## □□price□□□□□□□□□□points□□□points□□□
best_rating_per_price = reviews.groupby('price')['points'].max().sort_index()
```

agg□□

- □□□□□□□□□□□□□□□□□□□□□□□□□□□□

```
## □□□□□□□□□□□□□□□□□□□□□□□□□□
reviews.groupby(['country']).price.agg([len, min, max])
```

□□□□□

```
## □□□□□□□□□□□□□□□□□□□□□□□□□□
countries_reviewed.reset_index()
```

□□

- ascending□□□□True□□□□□□False□□□□
- inplace□□□□□□□Series
- by□□□□□□□□□□□□□□□□□□□□□□□□

```
## □□□□□□□
df['tianqi'].sort_values()
## □□len□□□□□
countries_reviewed.sort_values(by='len')
## □□□□□□□□□□□
countries_reviewed.sort_values(by=['country', 'len'])
## □□□□□□□□
countries_reviewed.sort_index()
```