pandas IO

In [1]:
```
import pandas as pd
import numpy as np
```

# 读文件

## *pd.read_csv( path, sep, header, index_col, names, skiprows, na_values, nrows, chunksize, encoding )*

> ***path*** : 文件位置
> ***sep*** : 对每行各字段进行拆分的字符序列
> ***header*** : 用作列名的行号, 默认为 0 , 没有标题行设为 None
> ***names*** : 用作设置列名的 list
> ***index_col*** : 用作索引的列标签, 可以是单个或多个列名组成的 list
> ***skiprows*** : 需要忽略的行号的 list
> ***na_values*** : 用作替换缺失值 NaN 的 list 或 dict
> ***nrows*** : 需要读取的行数
> ***chunksize*** : 返回分块读取的迭代器
> ***encoding*** : 用作 unicode 的文本编码格式

In [2]:
```
pd.read_csv('pydata-book-2nd-edition/examples/ex1.csv')
```

Out[2]:

| | a | b | c | d | message |
|---|---|---|---|---|---|
| **0** | 1 | 2 | 3 | 4 | hello |
| **1** | 5 | 6 | 7 | 8 | world |
| **2** | 9 | 10 | 11 | 12 | foo |

### *header* : 处理没有标题行的文件 :

In [3]:
```
pd.read_csv('pydata-book-2nd-edition/examples/ex2.csv')
```

Out[3]:

| | 1 | 2 | 3 | 4 | hello |
|---|---|---|---|---|---|
| **0** | 5 | 6 | 7 | 8 | world |
| **1** | 9 | 10 | 11 | 12 | foo |

header = None

In [4]:
```
pd.read_csv('pydata-book-2nd-edition/examples/ex2.csv', header=None)
```

Out[4]:

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| **0** | 1 | 2 | 3 | 4 | hello |
| **1** | 5 | 6 | 7 | 8 | world |
| **2** | 9 | 10 | 11 | 12 | foo |

names = list

In [5]:
```
pd.read_csv('pydata-book-2nd-edition/examples/ex2.csv', names=['q', 'w', 'e', 'r', 'message'])
```

Out[5]:

| | q | w | e | r | message |
|---|---|---|---|---|---|
| **0** | 1 | 2 | 3 | 4 | hello |
| **1** | 5 | 6 | 7 | 8 | world |
| **2** | 9 | 10 | 11 | 12 | foo |

### *index_col* : 设置索引 :

In [6]:
```
pd.read_csv('pydata-book-2nd-edition/examples/ex2.csv',
            names=['q', 'w', 'e', 'r', 'message'],
            index_col='message')
```

Out[6]:

| message | q | w | e | r |
|---|---|---|---|---|
| hello | 1 | 2 | 3 | 4 |
| world | 5 | 6 | 7 | 8 |
| foo | 9 | 10 | 11 | 12 |

层次化索引

In [7]:
```
pd.read_csv('pydata-book-2nd-edition/examples/csv_mindex.csv')
```

Out[7]:

| | key1 | key2 | value1 | value2 |
|---|---|---|---|---|
| 0 | one | a | 1 | 2 |
| 1 | one | b | 3 | 4 |
| 2 | one | c | 5 | 6 |
| 3 | one | d | 7 | 8 |
| 4 | two | a | 9 | 10 |
| 5 | two | b | 11 | 12 |
| 6 | two | c | 13 | 14 |
| 7 | two | d | 15 | 16 |

In [8]:
```
pd.read_csv('pydata-book-2nd-edition/examples/csv_mindex.csv', index_col=['key1', 'key2'])
```

Out[8]:

| key1 | key2 | value1 | value2 |
|---|---|---|---|
| one | a | 1 | 2 |
| | b | 3 | 4 |
| | c | 5 | 6 |
| | d | 7 | 8 |
| two | a | 9 | 10 |
| | b | 11 | 12 |
| | c | 13 | 14 |
| | d | 15 | 16 |

### *skiprows*：跳行

In [9]:
```
list(open('pydata-book-2nd-edition/examples/ex4.csv'))
```

Out[9]:
```
['# hey!\n',
 'a,b,c,d,message\n',
 '# just wanted to make things more difficult for you\n',
 '# who reads CSV files with computers, anyway?\n',
 '1,2,3,4,hello\n',
 '5,6,7,8,world\n',
 '9,10,11,12,foo']
```

In [10]:
```
pd.read_csv('pydata-book-2nd-edition/examples/ex4.csv', skiprows=[0, 2, 3])
# 跳过第1、3、4行
```

Out[10]:

| | a | b | c | d | message |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | hello |
| 1 | 5 | 6 | 7 | 8 | world |
| 2 | 9 | 10 | 11 | 12 | foo |

### *na_values*：处理缺失值

In [11]:
```
list(open('pydata-book-2nd-edition/examples/ex5.csv'))
```

Out[11]:
```
['something,a,b,c,d,message\n',
 'one,1,2,3,4,NA\n',
 'two,5,6,,8,world\n',
 'three,9,10,11,12,foo']
```

In [12]:
```
pd.read_csv('pydata-book-2nd-edition/examples/ex5.csv')
```

Out[12]:

| | something | a | b | c | d | message |
|---|---|---|---|---|---|---|
| 0 | one | 1 | 2 | 3.0 | 4 | NaN |
| 1 | two | 5 | 6 | NaN | 8 | world |
| 2 | three | 9 | 10 | 11.0 | 12 | foo |

na_value = dic/list

In [13]:
```
pd.read_csv('pydata-book-2nd-edition/examples/ex5.csv', na_values=['one', 'two', 'three'])
```

Out[13]:

| | something | a | b | c | d | message |
|---|---|---|---|---|---|---|
| 0 | NaN | 1 | 2 | 3.0 | 4 | NaN |
| 1 | NaN | 5 | 6 | NaN | 8 | world |
| 2 | NaN | 9 | 10 | 11.0 | 12 | foo |

In [14]:
```
# 字典的各列可以使用不同的 NaN 标记值
sentinels = {'message': ['foo', 'NA'], 'something': ['two']}
pd.read_csv('pydata-book-2nd-edition/examples/ex5.csv', na_values=sentinels)
```

Out[14]:

| | something | a | b | c | d | message |
|---|---|---|---|---|---|---|
| 0 | one | 1 | 2 | 3.0 | 4 | NaN |
| 1 | NaN | 5 | 6 | NaN | 8 | world |
| 2 | three | 9 | 10 | 11.0 | 12 | NaN |

## *pd.read_table( path, sep, ... )*

### *sep*：分隔符

In [15]:
```
pd.read_table('pydata-book-2nd-edition/examples/ex1.csv', sep=',')
```

Out[15]:

| | a | b | c | d | message |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | hello |
| 1 | 5 | 6 | 7 | 8 | world |
| 2 | 9 | 10 | 11 | 12 | foo |

In [16]:
```
list(open('pydata-book-2nd-edition/examples/ex3.txt'))
```

Out[16]:
```
['            A         B         C\n',
 'aaa -0.264438 -1.026059 -0.619500\n',
 'bbb  0.927272  0.302904 -0.032399\n',
 'ccc -0.264273 -0.386314 -0.217601\n',
 'ddd -0.871858 -0.348382  1.100491\n']
```

In [17]:
```
pd.read_table('pydata-book-2nd-edition/examples/ex3.txt', sep='\s+')
# 由于列名比数据行的数量少，所以 read_table 推断第一列应该是 DataFrame 的索引
```

Out[17]:

| | A | B | C |
|---|---|---|---|
| aaa | -0.264438 | -1.026059 | -0.619500 |
| bbb | 0.927272 | 0.302904 | -0.032399 |
| ccc | -0.264273 | -0.386314 | -0.217601 |
| ddd | -0.871858 | -0.348382 | 1.100491 |

# 逐块读取文件

### *pd.read_csv( path, nrows )*

In [18]:
```
pd.read_csv('pydata-book-2nd-edition/examples/ex6.csv', nrows=5)
```

Out[18]:

| | one | two | three | four | key |
|---|---|---|---|---|---|
| **0** | 0.467976 | -0.038649 | -0.295344 | -1.824726 | L |
| **1** | -0.358893 | 1.404453 | 0.704965 | -0.200638 | B |
| **2** | -0.501840 | 0.659254 | -0.421691 | -0.057688 | G |
| **3** | 0.204886 | 1.074134 | 1.388361 | -0.982404 | R |
| **4** | 0.354628 | -0.133116 | 0.283763 | -0.837063 | Q |

### *pd.read_csv( path, chunksize )*：迭代器

In [19]:
```
chunker = pd.read_csv('pydata-book-2nd-edition/examples/ex6.csv', chunksize=1000)
# 设置分块大小为1000行
tot = pd.Series([])
for piece in chunker:
    # 1000行分块
    tot = tot.add(piece['key'].value_counts(), fill_value=0)
    # piece['key'].value_counts() : 返回一个 series
    # 计算 key 中各值出现的次数

tot = tot.sort_values(ascending=False)
tot[:10]
```

```
<ipython-input-19-c6c0008a71bc>:3: DeprecationWarning: The default dtype for empty Series will be 'object' instead of
'float64' in a future version. Specify a dtype explicitly to silence this warning.
  tot = pd.Series([])
```

Out[19]:
```
E    368.0
X    364.0
L    346.0
O    343.0
Q    340.0
M    338.0
J    337.0
F    335.0
K    334.0
H    330.0
dtype: float64
```

## 写文件

### *frames.to_csv( path, sep, na_rep, index, header, columns )*：

In [20]:
```
data = pd.read_csv('pydata-book-2nd-edition/examples/ex5.csv')
data
```

Out[20]:

| | something | a | b | c | d | message |
|---|---|---|---|---|---|---|
| **0** | one | 1 | 2 | 3.0 | 4 | NaN |
| **1** | two | 5 | 6 | NaN | 8 | world |
| **2** | three | 9 | 10 | 11.0 | 12 | foo |

In [21]:
```
data.to_csv('pydata-book-2nd-edition/examples/out.csv')
```

#### *sep*：分隔符

In [22]:
```
data.to_csv(sys.stdout, sep='|' )
# 直接写出到 sys.stdout , 所以仅仅是打印出文本结果而已
```

```
|something|a|b|c|d|message
0|one|1|2|3.0|4|
1|two|5|6||8|world
2|three|9|10|11.0|12|foo
```

### *na_rep* : 缺失值

In [23]:
```
data.to_csv(sys.stdout, na_rep='-9999')
```

```
,something,a,b,c,d,message
0,one,1,2,3.0,4,-9999
1,two,5,6,-9999,8,world
2,three,9,10,11.0,12,foo
```

### *index, header* : 索引, 列标题的控制

In [24]:
```
data.to_csv(sys.stdout, index=False, header=False)
```

```
one,1,2,3.0,4,
two,5,6,,8,world
three,9,10,11.0,12,foo
```

### *columns* : 输出指定的行或列

In [25]:
```
data.to_csv(sys.stdout, index=False, columns=['a', 'b', 'c'])
```

```
a,b,c
1,2,3.0
5,6,
9,10,11.0
```

### *series.to_csv( path )* :

In [26]:
```
dates = pd.date_range('1/1/2000', periods=7)
obj = pd.Series(np.arange(7), index=dates)
obj.to_csv(sys.stdout)
```

```
,0
2000-01-01,0
2000-01-02,1
2000-01-03,2
2000-01-04,3
2000-01-05,4
2000-01-06,5
2000-01-07,6
```

# 读写 Excel 文件

### *pd.ExcelFile( path ) + pd.read_excel( ExcelFile, sheet_name )* :

> 读取一个文件的多个 sheet 时, 先创建 ExcelFile 传递路径, 然后使用 read_excel 读取, 速度更快

In [27]:
```
xlsx = pd.ExcelFile('pydata-book-2nd-edition/examples/ex1.xlsx')
pd.read_excel(xlsx, 'Sheet1')
```

Out[27]:

| | Unnamed: 0 | a | b | c | d | message |
|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 2 | 3 | 4 | hello |
| **1** | 1 | 5 | 6 | 7 | 8 | world |
| **2** | 2 | 9 | 10 | 11 | 12 | foo |

### *pd.read_excel( path, sheet_name )* :

> 直接传递文件路径到 pd.read_excel 中

In [28]:
```
pd.read_excel('pydata-book-2nd-edition/examples/ex1.xlsx', 'Sheet1')
```

Out[28]:

| | Unnamed: 0 | a | b | c | d | message |
|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 2 | 3 | 4 | hello |
| **1** | 1 | 5 | 6 | 7 | 8 | world |
| **2** | 2 | 9 | 10 | 11 | 12 | foo |

## *pd.ExcelWriter( path ) + frame.to_excel( ExcelWriter, sheet_name ) + ExcelWriter.save( )* :

> 首先创建 ExcelWriter，然后使用 to_excel 方法将数据写入到其中

In [29]:
```
frame = pd.read_excel('pydata-book-2nd-edition/examples/ex1.xlsx', 'Sheet1')
writer = pd.ExcelWriter('pydata-book-2nd-edition/examples/ex2.xlsx')
frame.to_excel(writer, 'Sheet1')
writer.save()
```

## *frame.to_excel( path, sheet_name )* :

> 直接传递文件路径到 to_excel 中

In [30]:
```
frame.to_excel('pydata-book-2nd-edition/examples/ex2.xlsx', 'Sheet1')
```