

pandas库

readme

- Pandas 库在处理excel类型数据非常好用，常见业务对接数据往往通过excel收集，学习之前请确保学习完成[正则表达式](#)

1. 导入库

```
1 import pandas as pd
```

2. 创建DataFrame

```
1 data = {'A': ['x', 'y', 'z'],  
2         'B': [1000, 2000, 3000],  
3         'C': [10, 20, 30]}  
4 df = pd.DataFrame(data, index = ['a', 'b', 'c'])  
5 df
```

- A, B, C 为列名，a, b, c 为行名

	A	B	C
a	x	1000	10
b	y	2000	20
c	z	3000	30

3. 查看DataFrame中的值 df.values

```
1 df.values
```

```
array([[ 'x', 1000, 10],  
       [ 'y', 2000, 20],  
       [ 'z', 3000, 30]], dtype=object)
```

4. 获取行列名df.index, df.columns

```
1 df.index  
2 df.columns
```

```
Index([ 'a', 'b', 'c'], dtype='object')  
Index([ 'A', 'B', 'C'], dtype='object')
```

- 创建时可以更改列的顺序，只需在 `columns` 处改变顺序

```
1 data = { 'A': [ 'x', 'y', 'z'],  
2         'B': [1000, 2000, 3000],  
3         'C': [10, 20, 30]}  
4 df = pd.DataFrame(data, columns = [ 'B', 'C', 'A'], index = [ 'row1', 'row2', 'row3'])  
5 df
```

	B	C	A
row1	1000	10	x
row2	2000	20	y
row3	3000	30	z

5. 添加、删除一列 del df['D']

```
1 df['D'] = 10 # 把D列全部赋值为10  
2 del df['D']
```

6. 横行合并两个DataFrame

- `how = 'left', 'right', 'inner', 'outer'`
- 这里和sql的合并表方式类似

```
1 df2 = pd.DataFrame(data = {'x1':['A','B','C'],'x2':[1,2,3]})
2 df3 = pd.DataFrame(data = {'x1':['A','B','D'],'x2':['T','F','T']})
3 pd.merge(df2,df3,how = 'left', on = 'x1')
```

	x1	x2_x	x2_y
0	A	1	T
1	B	2	F
2	C	3	NaN

```
1 pd.merge(df2,df3,how = 'right', on = 'x1')
```

	x1	x2_x	x2_y
0	A	1.0	T
1	B	2.0	F
2	D	NaN	T

```
1 pd.merge(df2,df3,how = 'outer', on = 'x1')
```

	x1	x2_x	x2_y
0	A	1.0	T
1	B	2.0	F
2	C	3.0	NaN
3	D	NaN	T

```
1 pd.merge(df2,df3,how = 'inner', on = 'x1')
```

	x1	x2_x	x2_y
0	A	1	T
1	B	2	F

7. 常见统计操作

```
1 import numpy as np
2 df = pd.DataFrame(np.random.randn(8,4),columns = list('ABCD'))
3 df
```

	A	B	C	D
0	0.132863	0.228572	0.874787	-0.261211
1	0.433415	0.626160	0.060306	-0.144069
2	0.337980	0.627468	0.795961	-0.081089
3	-0.652359	0.298969	-1.198896	-0.061287
4	0.690291	-0.782332	0.846143	-0.026827
5	0.788389	-0.096860	0.164575	-3.322447
6	-1.452064	-0.192849	0.190505	1.257995
7	-0.997478	-0.737240	1.736579	0.041573

```
1 df.sum()
2 df.mean()
3 df.var()
4 df.std()
5 df.cumsum() # 从上往下累加
6 df.describe()
7 df.info()
8 df.min()
9 df.max()
10 df['A'].value_counts() # 统计A列的数据分布
11 df.shape
```

8. DataFrame切片

```
1 data = {'A': ['x', 'y', 'z'],
2         'B': [1000, 2000, 3000],
3         'C': [10, 20, 30]}
4 df = pd.DataFrame(data, index = ['a', 'b', 'c'])
5
6 # 提取名为A的列
7 df['A']
8 df.A
9
10 # 提取名为a的行
11 df.loc['a']
12
13 # 提取a到c行, A和B列
14 df.loc['a': 'c', ['A', 'B']]
15
16 # 提取0-2行, 0, 1列
17 df.iloc[[0, 1, 2], 0:2]
```

- `df.loc[]` 依据行列名, `df.iloc[]` 依据行列号
- `df.iloc[]` 使用非常多, 经常搭配遍历 `for i in range(len(df)): df.iloc[i,1] = ''` 使用
- 即从上往下把每一行到第一列进行一个赋值

9. 通过筛选条件切片

```
1 df.C > 10
2 # output:
3 # a False
4 # b True
5 # c True
```

10. 改变一列的值

```
1 df.loc[:, 'D'] = 1 # 把D列全赋值为1
2
3 df.loc[:, 'D'] = np.array([1,2,3])
```

11. 读取、存储excel

- 对excel的数据实际落地项目中使用较多, 因此读取、存储非常关键

```
1 df = pd.read_csv('name4excel.csv')
2 df = pd.read_excel('name4excel.xlsx')
3
4 df.to_csv('output.csv', encoding = 'utf-8-sig', index = False)
```

12. 删除重复项

```

1 data = {'A': ['x', 'y', 'z', 't'],
2         'B': [1000, 1000, 1000, 2000],
3         'C': [10, 10, 30, 30]}
4 df = pd.DataFrame(data, index = ['a', 'b', 'c', 'd'])
5 print(df)
6 df.drop_duplicates(subset = ['B'])

```

	A	B	C
a	x	1000	10
b	y	1000	10
c	z	1000	30
d	t	2000	30

	A	B	C
a	x	1000	10
d	t	2000	30

```

1 data = {'A': ['x', 'y', 'z', 't'],
2         'B': [1000, 1000, 1000, 2000],
3         'C': [10, 10, 30, 30]}
4 df = pd.DataFrame(data, index = ['a', 'b', 'c', 'd'])
5 print(df)
6 df.drop_duplicates(subset = ['B', 'C'])

```

	A	B	C
a	x	1000	10
b	y	1000	10
c	z	1000	30
d	t	2000	30

	A	B	C
a	x	1000	10
c	z	1000	30
d	t	2000	30

13. 删除存在空值的行

```

1 data = {'A': ['x', 'y', 'z', 't'],
2         'B': [1000, 1000, 1000, None],
3         'C': [10, 10, 30, 30]}
4 df = pd.DataFrame(data, index = ['a', 'b', 'c', 'd'])
5 print(df)
6 df.loc[~df['B'].isna()]

```

	A	B	C
a	x	1000.0	10
b	y	1000.0	10
c	z	1000.0	30
d	t	NaN	30

	A	B	C
a	x	1000.0	10
b	y	1000.0	10
c	z	1000.0	30

14. 列相同的值合并分组操作

- B列相同的值分为一组，对C列求和

```

1 data = {'A': ['x', 'y', 'z', 't'],
2         'B': [1000, 2000, 3000, 1000],
3         'C': [10, 20, 30, 40]}
4 df = pd.DataFrame(data, index = ['a', 'b', 'c', 'd'])
5 df.groupby(['B'])[['C']].mean()

```

	C
B	
1000	25.0
2000	20.0
3000	30.0

15. 排序

```
1 data = {'A': ['x', 'y', 'z', 't'],
2         'B': [1000, 2000, 1000, 1000],
3         'C': [40, 20, 30, 10]}
4 df.sort_values('B')
5 df.sort_values(['B', 'C'])
```

16. 映射到二维平面

- 个人从来没使用过

```
1 df = pd.DataFrame({'client': ['John', 'John', 'Silvia', 'Silvia'], 'product':
2   ['bananas', 'oranges', 'bananas', 'oranges'], 'quantity': [5, 3, 4, 2]})
3 print(df)
4 df.pivot(index = 'client', columns = 'product', values = 'quantity')
```

	client	product	quantity
0	John	bananas	5
1	John	oranges	3
2	Silvia	bananas	4
3	Silvia	oranges	2

	product	bananas	oranges
client			
John		5	3
Silvia		4	2

17. 提取某列带特定字符的列表

```
1 df = pd.DataFrame({'A': ['a', 'b', 'c'], 'B': ['dc', 'ec', 'fc']})
2 df[df['B'].str.contains('d')]
```

- 思考题：是否可以如法炮制使用其他的字符串函数，比如lower, upper那些

18. 将字符'['转回列表

```
1 df['A'] = df['A'].apply(eval)
```

- 在读取excel时，`[]` 往往被看作是字符，不能被识别为list，需要上述操作更改
- 大家以后一定会遇到这个报错回来看这个解决方案

19. 爆炸函数

```
1 a = pd.DataFrame({'A':['a','b','c'],'B':[[1,2,3],[4,5],[6]]})  
2 a.explode('B')
```

	A	B
0	a	1
0	a	2
0	a	3
1	b	4
1	b	5
2	c	6