SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_index,col\_indexer] = value instead

|  |  |
| --- | --- |
| 21down voteaccepted | As suggested in the error message, you should use loc to do this:  sve2\_all.loc[sve2\_all['Hgtot ng/l'] > 100] = np.nan  *The warning is here to stop you modifying a copy (here sve2\_all[sve2\_all[' Hgtot ng/l'] > 100]is****potentially****a copy, and if it is then any modifications would not change the original frame. It could be that it works correctly in some cases but pandas cannot guarantee it will work in all cases... use at your own risk (consider yourself warned! ;) ).* |

<https://stackoverflow.com/questions/23002762/pandas-settingwithcopywarning>

<https://stackoverflow.com/questions/10062954/valueerror-the-truth-value-of-an-array-with-more-than-one-element-is-ambiguous>

ValueError: The truth value of an array with more than one element is ambiguous. Use a.any() or a.all()

Pandas 类型与各种取值类型的相互关系

>>> import pandas as pd

>>>

dfd = pd.DataFrame({'A': [1, 2, 3],

'B': [4, 5, 6]},

index=list('abc'))

>>> dfd

A B

a 1 4

b 2 5

c 3 6

>>> dfd[dfd.A>1]

A B

b 2 5

c 3 6

>>> dfd[dfd.A>2]

A B

c 3 6

>>> dfd[dfd.A>2]['A']

c 3

Name: A, dtype: int64

>>> type(dfd[dfd.A>2]['A'])

<class 'pandas.core.series.Series'>

>>> type(dfd[dfd.A>2])

<class 'pandas.core.frame.DataFrame'>

>>> dfd[dfd.A>2]['A'] == 3

c True

Name: A, dtype: bool

>>> type(dfd[dfd.A>2]['A'] == 3)

<class 'pandas.core.series.Series'>

>>> dfd[dfd.A>2]

A B

c 3 6

>>> dfd[dfd.A>2].values

array([[3, 6]], dtype=int64)

>>> dfd[dfd.A>1].values

array([[2, 5],

[3, 6]], dtype=int64)

>>> type(dfd[dfd.A>1].values)

<type 'numpy.ndarray'>

>>> for v in type(dfd[dfd.A>1].values): print v

Traceback (most recent call last):

File "<input>", line 1, in <module>

TypeError: 'type' object is not iterable

>>> for v in dfd[dfd.A>1].values: print v

[2 5]

[3 6]

>>> for v in dfd[dfd.A>1].values: print type(v)

<type 'numpy.ndarray'>

<type 'numpy.ndarray'>

>>> dd = pd.DataFrame([[1, 2, 3],

[4, 5, 6]],

index=list('ab'))

>>> dd

0 1 2

a 1 2 3

b 4 5 6

>>> d

0 1 2

a 1 2 3

b 4 5 6

c 9 9 9

>>> pd.merge(d, dd, how='outer', indicator=True).query('\_merge == "left\_only" ').drop(['\_merge'], axis=1)

0 1 2

2 9 9 9

>>> pd.merge(d, dd, how='outer', indicator=True)

0 1 2 \_merge

0 1 2 3 both

1 4 5 6 both

2 9 9 9 left\_only

>>> pd.merge(d, dd, how='outer', indicator=True).query('\_merge == "left\_only" ')

0 1 2 \_merge

2 9 9 9 left\_only

# pandas iloc vs ix vs loc explanation

<https://stackoverflow.com/questions/31593201/pandas-iloc-vs-ix-vs-loc-explanation>

pd.loc[:n] 按照索引的值去查找， 如果索引不是顺序的，则返回的是index=n这一行之前的所有数据

pd.iloc[:n] 返回的是数据的location为n之前的所有数据。

>>> s = pd.Series(np.nan, index=[49,48,47,46,45, 1, 2, 3, 4, 5])

>>> s

49 NaN

48 NaN

47 NaN

46 NaN

45 NaN

1 NaN

2 NaN

3 NaN

4 NaN

5 NaN

>>> s.iloc[:3] # slice the first three rows

49 NaN

48 NaN

47 NaN

>>> s.loc[:3] # slice up to and including label 3

49 NaN

48 NaN

47 NaN

46 NaN

45 NaN

1 NaN

2 NaN

3 NaN

>>> s.ix[:3] # the integer is in the index so s.ix[:3] works like loc

49 NaN

48 NaN

47 NaN

46 NaN

45 NaN

1 NaN

2 NaN

3 NaN

drop\_duplicates

>>>

dfd = pd.DataFrame({ 'A': [1, 2, 3, 1],

'C': [1, 2, 3, 1],

'B': [4, 5, 6, 4]})

>>> dfd

A B C

0 1 4 1

1 2 5 2

2 3 6 3

3 1 4 1

>>> dfd.drop\_duplicates()

A B C

0 1 4 1

1 2 5 2

2 3 6 3

>>> dfd = pd.DataFrame({ 'A': [1, 2, 3, 1, 1, 1],

'C': [1, 2, 3, 1, 2, 1],

'B': [4, 5, 6, 4, 4, 5]})

>>> dfd

A B C

0 1 4 1

1 2 5 2

2 3 6 3

3 1 4 1

4 1 4 2

5 1 5 1

>>> dfd.drop\_duplicates()

A B C

0 1 4 1

1 2 5 2

2 3 6 3

4 1 4 2

5 1 5 1

>>> dfd.drop\_duplicates(keep='last')

A B C

1 2 5 2

2 3 6 3

3 1 4 1

4 1 4 2

5 1 5 1

>>> dfd.drop\_duplicates(keep='first')

A B C

0 1 4 1

1 2 5 2

2 3 6 3

4 1 4 2

5 1 5 1

>>> dfd = pd.DataFrame({ 'A': [1, 2, 3, 1, 1, 1],

'C': [1, 2, 3, 1, 2, 1],

'B': [4, 5, 6, 4, 4, 5],

'D': [1, 2, 3, 1, 1, 1],})

>>> dfd

A B C D

0 1 4 1 1

1 2 5 2 2

2 3 6 3 3

3 1 4 1 1

4 1 4 2 1

5 1 5 1 1

>>> dfd.drop\_duplicates('A', keep='first')

A B C D

0 1 4 1 1

1 2 5 2 2

2 3 6 3 3

>>> dfd.drop\_duplicates(['A','C'], keep='first')

A B C D

0 1 4 1 1

1 2 5 2 2

2 3 6 3 3

4 1 4 2 1

>>> dfd.drop\_duplicates(['A','C'], keep='last')

A B C D

1 2 5 2 2

2 3 6 3 3

4 1 4 2 1

5 1 5 1 1

>>> dfd

A B C D

0 1 4 1 1

1 2 5 2 2

2 3 6 3 3

3 1 4 1 1

4 1 4 2 1

5 1 5 1 1

>>> dfd.drop\_duplicates(['A','C'], keep='last', inplace=True)

>>> dfd

A B C D

1 2 5 2 2

2 3 6 3 3

4 1 4 2 1

5 1 5 1 1

Inplace表示是否在原数据上做修改，默认是false.

pd.to\_Datetime

pd.to\_numeric

create

创建一个3行4列，索引为给定的date series 的DataFrame.

>>> dates = pd.date\_range('20130101', periods=6)

>>>df = pd.DataFrame(np.random.randn(6,4), index=dates, columns=list('ABCD'))

<https://segmentfault.com/a/1190000008759795>

根据给定的列按值排序

df.sort\_values(by='B')

assign

<http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.assign.html>

tolist, to list

[DataFrame] df\_secid

|  |
| --- |
| secID  0 000001.XSHE  1 000002.XSHE  2 000004.XSHE  3 000005.XSHE  4 000007.XSHE |

df\_secid.secID.values.tolist()

|  |
| --- |
| [u'000001.XSHE', u'000002.XSHE', u'000004.XSHE', u'000005.XSHE', u'000007.XSHE', …] |

>>> type(list(dfd['B']))

<type 'list'>

>>> dfd = pd.DataFrame({ 'A': [1, 2, 3, 1],

'C': [1, 2, 3, 1],

'B': ['a','b','c','e']})

>>> dfd

A B C

0 1 a 1

1 2 b 2

2 3 c 3

3 1 e 1

>>> type(list(dfd['B']))

<type 'list'>

>>> ','.join(list(dfd['B']))

'a,b,c,e'

>>> ','.join(list(str(dfd['C'])))

'0, , , , ,1,\n,1, , , , ,2,\n,2, , , , ,3,\n,3, , , , ,1,\n,N,a,m,e,:, ,C,,, ,d,t,y,p,e,:, ,i,n,t,6,4'

>>> ','.join([str(x) for x in list(dfd['B'])])

'a,b,c,e'

>>> ','.join([str(x) for x in list(dfd['C'])])

'1,2,3,1'

与条件筛选行， and

df[(df.stockid > 0) & (df.report\_type == 1)]

convert to list

import pandas as pd

df = pd.DataFrame({'a':[1,3,5,7,4,5,6,4,7,8,9],

'b':[3,5,6,2,4,6,7,8,7,8,9]})

>>> df['a'].values.tolist()

[1, 3, 5, 7, 4, 5, 6, 4, 7, 8, 9]