Dicin'

Bookmark this page

Slicin' and Dicin', Part Two

Boolean Indexing

Your dataframes and series can also be indexed with a *boolean operation*—a dataframe or series with the same dimensions as the one you are selecting from, but with every value either being set to True or False. You can create a new boolean series either by manually specifying the values, or by using a conditional:

```
>>> df.recency < 7

0        False
1         True
2        False
3        False
4         True
5         True

Name: recency, dtype: bool
```

To index with your boolean series, simply feed it back into your regular series with using the [] bracket-selection syntax. The result is a new series that once again has the same dimensions, however only values corresponding to True values in the boolean series get returned:

```
>>> df[ df.recency < 7 ]

       recency   history_segment  history  mens  womens   zip_code  newbie
1            6    3) $200 - $350   329.08     1       1      Rural       1
4            2      1) $0 - $100    45.34     1       0      Urban       0
5            6    2) $100 - $200   134.83     0       1  Surburban       0
```

If you need even finer grain control of what gets selected, you can further combine multiple boolean indexing conditionals together using the bit-wise logical operators | and &:

```
>>> df[ (df.recency < 7) & (df.newbie == 0) ]

       recency history_segment  history  mens  womens   zip_code  newbie
4            2    1) $0 - $100    45.34     1       0      Urban       0
5            6  2) $100 - $200   134.83     0       1  Surburban       0
```

This is a bit counter-intuitive, as most people initially assume Pandas would support the regular, Python boolean operators or and and. The reason regular Python boolean operators cannot be used to combine Pandas boolean conditionals is because doing so causes ambiguity. There are two ways the following incorrect statement can be interpreted (df.recency<7) or (df.newbie==0):

1. If the evaluation the statement (df.recency<7) or the evaluation the statement (df.newbie==0) results in anything besides the False, then select all records in the dataset.
2. Select all columns belonging to rows in the dataset where either of the following statements are true: (df.recency<7) or (df.newbie==0).

Option 2 is the desired functionality, but to avoid this ambiguity entirely, Pandas overloads bit-wise operators on its dataframe and series objects. Be **sure** to encapsulate each conditional in parenthesis to make this work.

Writing to a Slice

Something handy that you can do with a dataframe or series is write into a slice:

```
>>> df[df.recency < 7] = -100
>>> df

       recency   history_segment  history  mens  womens   zip_code  newbie
0           10    2) $100 - $200   142.44     1       0  Surburban       0
1         -100              -100  -100.00  -100    -100       -100    -100
2            7    2) $100 - $200   180.65     0       1  Surburban       1
3            9    5) $500 - $750   675.83     1       0      Rural       1
4         -100              -100  -100.00  -100    -100       -100    -100
5         -100              -100  -100.00  -100    -100       -100    -100
```

Take precaution while doing this, as you may encounter issues with non-homogeneous dataframes. It is far safer, and generally makes more sense, to do this sort of operation on a per column basis rather than across your entire dataframe. In the above example, -100 is rendered as an integer in the recency column, a string in the history_segment column, and a float in the history column.

English