pandas.DataFrame.loc

property DataFrame.loc [source]

Access a group of rows and columns by label(s) or a boolean array.

`.loc[]` is primarily label based, but may also be used with a boolean array.

Allowed inputs are:

- A single label, e.g. `5` or `'a'`, (note that `5` is interpreted as a *label* of the index, and **never** as an integer position along the index).
- A list or array of labels, e.g. `['a', 'b', 'c']`.
- A slice object with labels, e.g. `'a':'f'`.

> **Warning**
>
> Note that contrary to usual python slices, **both** the start and the stop are included

- A boolean array of the same length as the axis being sliced, e.g. `[True, False, True]`.
- An alignable boolean Series. The index of the key will be aligned before masking.
- An alignable Index. The Index of the returned selection will be the input.
- A `callable` function with one argument (the calling Series or DataFrame) and that returns valid output for indexing (one of the above)

See more at Selection by Label.

Raises:

KeyError

If any items are not found.

IndexingError

If an indexed key is passed and its index is unalignable to the frame index.

Skip to main content

See also

`DataFrame.at`
: Access a single value for a row/column label pair.

`DataFrame.iloc`
: Access group of rows and columns by integer position(s).

`DataFrame.xs`
: Returns a cross-section (row(s) or column(s)) from the Series/DataFrame.

`Series.loc`
: Access group of values using labels.

Examples

Getting values

```
>>> df = pd.DataFrame([[1, 2], [4, 5], [7, 8]],
...                   index=['cobra', 'viper', 'sidewinder'],
...                   columns=['max_speed', 'shield'])
>>> df
            max_speed  shield
cobra               1       2
viper               4       5
sidewinder          7       8
```

Single label. Note this returns the row as a Series.

```
>>> df.loc['viper']
max_speed    4
shield       5
Name: viper, dtype: int64
```

List of labels. Note using `[[]]` returns a DataFrame.

```
>>> df.loc[['viper', 'sidewinder']]
            max_speed  shield
viper               4       5
sidewinder          7       8
```

Single label for row and column

Skip to main content

```
>>> df.loc['cobra', 'shield']
2
```

Slice with labels for row and single label for column. As mentioned above, note that both the start and stop of the slice are included.

```
>>> df.loc['cobra':'viper', 'max_speed']
cobra    1
viper    4
Name: max_speed, dtype: int64
```

Boolean list with the same length as the row axis

```
>>> df.loc[[False, False, True]]
            max_speed  shield
sidewinder          7       8
```

Alignable boolean Series:

```
>>> df.loc[pd.Series([False, True, False],
...                  index=['viper', 'sidewinder', 'cobra'])]
                  max_speed  shield
sidewinder          7       8
```

Index (same behavior as `df.reindex`)

```
>>> df.loc[pd.Index(["cobra", "viper"], name="foo")]
       max_speed  shield
foo
cobra          1       2
viper          4       5
```

Conditional that returns a boolean Series

```
>>> df.loc[df['shield'] > 6]
            max_speed  shield
sidewinder          7       8
```

Conditional that returns a boolean Series with column labels specified

Skip to main content

```
>>> df.loc[df['shield'] > 6, ['max_speed']]
            max_speed
sidewinder          7
```

Multiple conditional using `&` that returns a boolean Series

```
>>> df.loc[(df['max_speed'] > 1) & (df['shield'] < 8)]
            max_speed  shield
viper           4       5
```

Multiple conditional using `|` that returns a boolean Series

```
>>> df.loc[(df['max_speed'] > 4) | (df['shield'] < 5)]
            max_speed  shield
cobra               1       2
sidewinder          7       8
```

Please ensure that each condition is wrapped in parentheses `()`. See the user guide for more details and explanations of Boolean indexing.

Note

If you find yourself using 3 or more conditionals in `.loc[]`, consider using advanced indexing.

See below for using `.loc[]` on MultiIndex DataFrames.

Callable that returns a boolean Series

```
>>> df.loc[lambda df: df['shield'] == 8]
            max_speed  shield
sidewinder          7       8
```

Setting values

Set value for all items matching the list of labels

```
>>> df.loc[['viper', 'sidewinder'], ['shield']] = 50
```

Skip to main content

```
cobra               1       2
viper               4      50
sidewinder          7      50
```

Set value for an entire row

```
>>> df.loc['cobra'] = 10
>>> df
            max_speed  shield
cobra              10      10
viper               4      50
sidewinder          7      50
```

Set value for an entire column

```
>>> df.loc[:, 'max_speed'] = 30
>>> df
            max_speed  shield
cobra              30      10
viper              30      50
sidewinder         30      50
```

Set value for rows matching callable condition

```
>>> df.loc[df['shield'] > 35] = 0
>>> df
            max_speed  shield
cobra              30      10
viper               0       0
sidewinder          0       0
```

Add value matching location

```
>>> df.loc["viper", "shield"] += 5
>>> df
            max_speed  shield
cobra              30      10
viper               0       5
sidewinder          0       0
```

Setting using a `Series` or a `DataFrame` sets the values matching the index labels, not the index positions.

```
>>> shuffled_df = df.loc[["viper", "cobra", "sidewinder"]]
>>> df.loc[:] += shuffled_df
>>> df
            max_speed  shield
cobra              60      20
viper               0      10
sidewinder          0       0
```

Getting values on a DataFrame with an index that has integer labels

Another example using integers for the index

```
>>> df = pd.DataFrame([[1, 2], [4, 5], [7, 8]],
...                   index=[7, 8, 9], columns=['max_speed', 'shield'])
>>> df
   max_speed  shield
7          1       2
8          4       5
9          7       8
```

Slice with integer labels for rows. As mentioned above, note that both the start and stop of the slice are included.

```
>>> df.loc[7:9]
   max_speed  shield
7          1       2
8          4       5
9          7       8
```

Getting values with a MultiIndex

A number of examples using a DataFrame with a MultiIndex

```
>>> tuples = [
...     ('cobra', 'mark i'), ('cobra', 'mark ii'),
...     ('sidewinder', 'mark i'), ('sidewinder', 'mark ii'),
...     ('viper', 'mark ii'), ('viper', 'mark iii')
... ]
>>> index = pd.MultiIndex.from_tuples(tuples)
>>> values = [[12, 2], [0, 4], [10, 20],
...           [1, 4], [7, 1], [16, 36]]
>>> df = pd.DataFrame(values, columns=['max_speed', 'shield'], index=index)
>>> df
                     max_speed  shield
```

Skip to main content

```
sidewinder mark i          10      20
           mark ii          1       4
viper      mark ii          7       1
           mark iii        16      36
```

Single label. Note this returns a DataFrame with a single index.

```
>>> df.loc['cobra']
         max_speed  shield
mark i          12       2
mark ii          0       4
```

Single index tuple. Note this returns a Series.

```
>>> df.loc[('cobra', 'mark ii')]
max_speed    0
shield       4
Name: (cobra, mark ii), dtype: int64
```

Single label for row and column. Similar to passing in a tuple, this returns a Series.

```
>>> df.loc['cobra', 'mark i']
max_speed    12
shield        2
Name: (cobra, mark i), dtype: int64
```

Single tuple. Note using `[[]]` returns a DataFrame.

```
>>> df.loc[[('cobra', 'mark ii')]]
               max_speed  shield
cobra mark ii          0       4
```

Single tuple for the index with a single label for the column

```
>>> df.loc[('cobra', 'mark i'), 'shield']
2
```

Slice from index tuple to single label

Skip to main content

```
cobra      mark i           12       2
           mark ii           0       4
sidewinder mark i           10      20
           mark ii           1       4
viper      mark ii           7       1
           mark iii         16      36
```

Slice from index tuple to index tuple

```
>>> df.loc[('cobra', 'mark i'):('viper', 'mark ii')]
                     max_speed  shield
cobra      mark i           12       2
           mark ii           0       4
sidewinder mark i           10      20
           mark ii           1       4
viper      mark ii           7       1
```

Please see the user guide for more details and explanations of advanced indexing.

Previous Next