All-The-Snippets Pandas

Andreea Gheorghe

Dec 16, 2020

.loc and .iloc:

Chapter 1

Selecting data from a pandas DataFrame

1.1 Select Data With .loc & .iloc

The common ways of selecting rows and columns in a pandas DataFrame are the functions **.loc** (*label based selection*, which means that one needs to specify the name of the rows and columns) and **.iloc** (*index based selection*, which means that one needs to specify rows and columns by their integer index).

1.1.1 .loc selection by index/labels and columns

```
df.loc[<row selection>, <column selection>]
```

1.1.2 .loc select all rows and some columns by index/labels

```
df.loc[:, ['column1', 'column2', 'column3']]
```

1.1.3 .loc select some rows and all columns by index/labels

```
df.loc[['row1', 'row2'], :]
# or ommiting the colon for columns labels
df.loc[['row1', 'row2']]
```

1.1.4 .loc select some rows and some columns by index/labels

```
df.loc[['row1', 'row2'], ['column1', 'column2', 'column3']]
```

1.1.5 Select rows whose column values *equal* some value/scalar

```
df.loc[df['column_x'] == some_value]
```

1.1.6 Select rows whose column values *do not equal* some_value/scalar

```
df.loc[df['column_x'] != some_value]
```

1.1.7 Select rows whose column values *are in* a list

```
df.loc[df['column_x'].isin(['A', 'B', 'C'])]
```

1.1.8 Select rows whose column values *are not* in a list

```
df.loc[~df['column_x'].isin(['A', 'B', 'C'])]
```

1.1.9 Select rows that satisfy multiple *and* (&) boolean conditions

```
# note the parentheses due to python operators precedence!
df.loc[(df['column_x'] >= A) & (df['column_y'] <= B)]
```

1.1.10 Select rows that satisfy multiple *or* (|) boolean conditions

```
# note the parentheses due to python operators precedence!
df.loc[(df['column_x'] >= A) | (df['column_y'] <= B)]
```

1.1.11 Update column values based on rows condition

```
df.loc[df['column_x'] >= A, 'column_y'] = new_value
```

1.1.12 Update multiple columns values based on rows condition

```
df.loc[df['column_x'] >= A, ['column_y', 'column_z']] = [new_value_y, new_value_
↪z]
```

1.1.13 .loc selection for single label, all columns / returns a Series

```
# giving DataFrame df
>>> df
     Name  Age
a    Alex   10
b     Bob   12
c  Clarke   13

df.loc['a']

>>> df.loc['a']
Name    Alex
Age       10
Name: a, dtype: object
```

1.1.14 .loc selection for list of label, all columns / returns a DataFrame

```
# giving DataFrame df
>>> df
     Name  Age
a    Alex   10
b     Bob   12
c  Clarke   13

df.loc[['a', 'b']]

>>> df.loc[['a', 'b']]
   Name  Age
a  Alex   10
b   Bob   12
```

1.1.15 .loc selection for single label, single column / returns one value

```
# giving DataFrame df
>>> df
      Name  Age
a     Alex   10
b      Bob   12
c   Clarke   13

df.loc['c', 'Age']

>>> df.loc['c', 'Age']
13
```

1.1.16 .loc selection for all label, single column / returns a Series

```
# giving DataFrame df
>>> df
      Name  Age
a     Alex   10
b      Bob   12
c   Clarke   13

df.loc[:, 'Age']

>>> df.loc[:, 'Age']
a    10
b    12
c    13
Name: Age, dtype: int64
```

1.1.17 .loc selection for multiple labels, multiple columns / returns a DataFrame

```
# giving DataFrame df
>>> df
      Name  Age
a     Alex   10
b      Bob   12
c   Clarke   13

df.loc[['a', 'c'], ['Age', 'Name']]

>>> df.loc[['a', 'c'], ['Age', 'Name']]
```

(continues on next page)

(continued from previous page)

```
   Age    Name
a   10    Alex
c   13  Clarke
```

1.1.18 .loc selection for slice of labels for row, all columns / returns a DataFrame

```
# giving DataFrame df
>>> df
     Name  Age
a    Alex   10
b     Bob   12
c  Clarke   13

df.loc['b':'c']

# or

df.loc['b':'c', ['Name', 'Age']]

>>> df.loc['b':'c']
     Name  Age
b     Bob   12
c  Clarke   13
```

1.1.19 .loc selection for all rows, slice of columns / returns a DataFrame

```
# giving DataFrame df
>>> df
     Name  Age Occupation
a    Alex   10   Student
b     Bob   12   Teacher
c  Clarke   13  Engineer

df.loc[:, 'Age':'Occupation']

>>> df.loc[:, 'Age':'Occupation']
   Age Occupation
a   10    Student
b   12    Teacher
c   13   Engineer
```

1.1.20 .loc selection for rows and columns with boolean conditions

```
# length of boolean arrays must match dataframe size
# select rows 1 and 3 and first column
df.loc[[True, False, True], [True]]
```

1.1.21 Select rows 1 and 3, and second column with boolean conditions

```
# giving DataFrame df
>>> df
     Name  Age Occupation
a    Alex   10   Student
b     Bob   12   Teacher
c  Clarke   13  Engineer

df.loc[[True, False, True], [False, True]]

>>> df.loc[[True, False, True], [False, True]]
   Age
a   10
c   13
>>> df
```

1.1.22 Select rows whose column values equals a scalar

```
# giving DataFrame df
>>> df
     Name  Age Occupation
a    Alex   10   Student
b     Bob   12   Teacher
c  Clarke   13  Engineer

df.loc[df['Name'] == 'Alex']

>>> df.loc[df['Name'] == 'Alex']
   Name  Age Occupation
a  Alex   10    Student

# df['Name'] == 'Alex' produces a Pandas Series with a True/False value for␣
↪every row in the data
# where there are 'True' values for the rows where the Name is 'Alex'.

>>> df['Name'] == 'Alex'
a     True
b    False
```

(continues on next page)

(continued from previous page)

```
c    False
Name: Name, dtype: bool
```

1.1.23 Select rows that satisfy multiple boolean conditions; keep selected columns

```
# giving DataFrame df
>>> df
     Name  Age Occupation
a    Alex   10   Student
b     Bob   12   Teacher
c  Clarke   13  Engineer

df.loc[(df['Name'] == 'Alex') | (df['Age'] == 12), ['Name', 'Occupation']]

>>> df.loc[(df['Name'] == 'Alex') | (df['Age'] == 12), ['Name', 'Occupation']]
   Name Occupation
a  Alex    Student
b   Bob    Teacher
```

1.1.24 Update column values based on rows condition

```
# giving DataFrame df
>>> df
     Name  Age Occupation
a    Alex   10   Student
b     Bob   12   Teacher
c  Clarke   13  Engineer

# update Alex Occupation to Grad Student
df.loc[(df['Name'] == 'Alex'), ['Occupation']] = 'Grad Student'

>>> df
     Name  Age    Occupation
a    Alex   10  Grad Student
b     Bob   12       Teacher
c  Clarke   13      Engineer
```

1.1.25 Update multiple columns values based on rows condition

```
# giving DataFrame df
>>> df
     Name  Age Occupation
a    Alex   10   Student
b     Bob   12   Teacher
c  Clarke   13  Engineer

# update Alex Age and Occupation to 20 years and Grad Student
df.loc[(df['Name'] == 'Alex'), ['Age', 'Occupation']] = [20, 'Grad Student']

>>> df
     Name  Age    Occupation
a    Alex   20  Grad Student
b     Bob   12       Teacher
c  Clarke   13      Engineer
```

1.1.26 .iloc position based selection (from 0 to length-1 of the axis)

```
df.iloc[<row selection>, <column selection>]
```

1.1.27 .iloc single row selections, all columns

```
# return Series object
df.iloc[0]  # first row of dataframe
df.iloc[1]  # second row of dataframe
df.iloc[-1] # last row of data frame
```

1.1.28 .iloc single column selections, all rows

```
# return Series object
df.iloc[:, 0]  # first column of dataframe
df.iloc[:, 1]  # second column of dataframe
df.iloc[:, -1] # last column of data frame
```

1.1.29 .iloc multiple row and column selections

```
# return DataFrame object
df.iloc[0:3]              # first three rows of dataframe, all columns
df.iloc[:, 0:3]           # first three columns of dataframe, all rows
df.iloc[0:5, 4:7]         # first five rows, column 4th, 5th and 6th of␣
↪dataframe
df.iloc [[3, 4], [1, 2]] # retrieve rows 4th & 5th, columns 2nd & 3rd with␣
↪lists
```

1.1.30 .iloc selection for rows and columns with boolean conditions

```
# length of boolean arrays must match dataframe size
# select rows 1 and 3 and first column
df.iloc[[True, False, True], [True]]
```

1.1.31 Select the rows whose index label is an even number

```
df.iloc[lambda x: x.index % 2 == 0]
```

1.1.32 Select slice of rows with .iloc

```
# start_pos is the index of the first row to retrieve
# stop_pos is the index of the row to stop (last row retrieved will have the␣
↪index of `stop_pos -1`)
df.iloc[start_pos:stop_pos]
```

1.1.33 Update/set column values based on rows condition with .iloc

```
# set column 4 values to `new_value` for first two rows of data frame
df.iloc[0:2, 3] = 'new_value'

# set all column values to `new_value` for first two rows of data frame
df.iloc[0:2] = 'new_value'
```

1.2 More pandas snippets

1.2.1 Browse pandas snippets

Visit https://allthesnippets.com/browse/

1.2.2 Search pandas snippets

Visit https://allthesnippets.com/search/

1.2.3 Send feedback and connect on twitter

@andreagheorghe

1.2.4 Keep on learning!