7 - Pandas-Indexes

March 22, 2023

Table of Contents

1 Pandas `loc` and `iloc` for selecting data

This is a notebook for the medium article How to use `loc` and `iloc` for selecting data in Pandas

Please check out article for instructions

[1]:
```
import pandas as pd
```

[2]:
```
data = {
    'Weather': ['Sunny','Sunny','Sunny','Cloudy','Shower','Shower','Sunny'],
    'Temperature': [78,76,78,68,70,71,82],
    'Wind': [13,28,16,11,26,27,20],
    'Humidity': [30,96,20,22,79,62,10],
}
df = pd.DataFrame(data, index = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat',␣
 ↪'Sun'])
df
```

[2]:
```
    Weather  Temperature  Wind  Humidity
Mon   Sunny           78    13        30
Tue   Sunny           76    28        96
```

```
Wed    Sunny           78     16           20
Thu   Cloudy           68     11           22
Fri   Shower           70     26           79
Sat   Shower           71     27           62
Sun    Sunny           82     20           10
```

1.1 1. Differences between loc and iloc

The main distinction between `loc` and `iloc` is: * `loc` is label-based, which means that you have to specify rows and columns based on their row and column labels. * `iloc` is integer position-based, so you have to specify rows and columns by their integer position values (0-based integer position).

1.2 2. Selecting via a single value

To get Fridays' temperature

[3]:
```
# Pass label to `loc`
df.loc['Fri', 'Temperature']
```

[3]:
```
70
```

[4]:
```
# The equivalent `iloc` statement should take row number 4 and column number 1
df.iloc[4, 1]
```

[4]:
```
70
```

Use `:` to return all data

[5]:
```
# To get all rows
df.loc[:, 'Temperature']
```

[5]:
```
Mon    78
Tue    76
Wed    78
Thu    68
Fri    70
Sat    71
Sun    82
Name: Temperature, dtype: int64
```

[6]:
```
# The equivalent `iloc` statement
df.iloc[:, 1]
```

[6]:
```
Mon    78
Tue    76
Wed    78
Thu    68
Fri    70
```

```
Sat    71
Sun    82
Name: Temperature, dtype: int64
```

[7]:
```
# To get all columns
df.loc['Fri', :]
```

[7]:
```
Weather        Shower
Temperature        70
Wind               26
Humidity           79
Name: Fri, dtype: object
```

[8]:
```
# The equivalent `iloc` statement
df.iloc[4, :]
```

[8]:
```
Weather        Shower
Temperature        70
Wind               26
Humidity           79
Name: Fri, dtype: object
```

1.3 3. Selecting via a list of values

[9]:
```
# Multiple rows
df.loc[['Thu', 'Fri'], 'Temperature']
```

[9]:
```
Thu    68
Fri    70
Name: Temperature, dtype: int64
```

[10]:
```
# Multiple columns
df.loc['Fri', ['Temperature', 'Wind']]
```

[10]:
```
Temperature    70
Wind           26
Name: Fri, dtype: object
```

[11]:
```
# Multiple rows using iloc
df.iloc[[3, 4], 1]
```

[11]:
```
Thu    68
Fri    70
Name: Temperature, dtype: int64
```

[12]:
```
# Multiple columns using iloc
df.iloc[4, [1, 2]]
```

[12]:
```
Temperature    70
Wind           26
Name: Fri, dtype: object
```

[13]:
```
# Multiple rows and columns
rows = ['Thu', 'Fri']
cols=['Temperature','Wind']

df.loc[rows, cols]
```

[13]:
```
     Temperature  Wind
Thu           68    11
Fri           70    26
```

[14]:
```
# the equivalent iloc statement
rows = [3, 4]
cols = [1, 2]
df.iloc[rows, cols]
```

[14]:
```
     Temperature  Wind
Thu           68    11
Fri           70    26
```

1.4 4. Selecting a range of data via slice

For loc, we can use the syntax `A:B` to select data from label `A` to label `B` (Both `A` and `B` are included):

[15]:
```
# Slicing column labels
rows=['Thu', 'Fri']
df.loc[rows, 'Temperature':'Humidity' ]
```

[15]:
```
     Temperature  Wind  Humidity
Thu           68    11        22
Fri           70    26        79
```

[16]:
```
# Slicing row labels
cols = ['Temperature', 'Wind']
df.loc['Mon':'Thu', cols]
```

[16]:
```
     Temperature  Wind
Mon           78    13
Tue           76    28
Wed           78    16
Thu           68    11
```

We can use the syntax `A:B:S` to select data from label `A` to label `B` with step size `S` (Both `A` and `B` are included):

[17]:
```
# Slicing with step
df.loc['Mon':'Fri':2 , :]
```

[17]:
```
    Weather  Temperature  Wind  Humidity
Mon   Sunny           78    13        30
Wed   Sunny           78    16        20
Fri  Shower           70    26        79
```

With iloc, we can also use the syntax `n:m` to select data from position `n` (included) to position `m` (excluded).

[18]:
```
df.iloc[[1, 2], 0 : 3]
```

[18]:
```
    Weather  Temperature  Wind
Tue   Sunny           76    28
Wed   Sunny           78    16
```

[19]:
```
df.iloc[0:4:2, :]
```

[19]:
```
    Weather  Temperature  Wind  Humidity
Mon   Sunny           78    13        30
Wed   Sunny           78    16        20
```

1.5 5. Selecting via conditions and callable

1.5.1 5.2 Conditions

[20]:
```
# One condition
df.loc[df.Humidity > 50, :]
```

[20]:
```
    Weather  Temperature  Wind  Humidity
Tue   Sunny           76    28        96
Fri  Shower           70    26        79
Sat  Shower           71    27        62
```

[21]:
```
## multiple conditions
df.loc[
    (df.Humidity > 50) & (df.Weather == 'Shower'),
    ['Temperature','Wind'],
]
```

[21]:
```
     Temperature  Wind
Fri           70    26
Sat           71    27
```

[22]:
```
# Getting ValueError
#df.iloc[df.Humidity > 50, :]
```

[23]:
```
# Single condition
df.iloc[list(df.Humidity > 50)]
```

[23]:
```
    Weather  Temperature  Wind  Humidity
Tue   Sunny           76    28        96
Fri  Shower           70    26        79
Sat  Shower           71    27        62
```

[24]:
```
## multiple conditions
df.iloc[
    list((df.Humidity > 50) & (df.Weather == 'Shower')),
    :,
]
```

[24]:
```
    Weather  Temperature  Wind  Humidity
Fri  Shower           70    26        79
Sat  Shower           71    27        62
```

1.5.2 5.2 Callable

[25]:
```
# Selecting columns
df.loc[:, lambda df: ['Humidity', 'Wind']]
```

[25]:
```
     Humidity  Wind
Mon        30    13
Tue        96    28
Wed        20    16
Thu        22    11
Fri        79    26
Sat        62    27
Sun        10    20
```

[26]:
```
# With condition
df.loc[lambda df: df.Humidity > 50, :]
```

[26]:
```
    Weather  Temperature  Wind  Humidity
Tue   Sunny           76    28        96
Fri  Shower           70    26        79
Sat  Shower           71    27        62
```

[27]:
```
df.iloc[lambda df: [0,1], :]
```

[27]:
```
    Weather  Temperature  Wind  Humidity
Mon   Sunny           78    13        30
Tue   Sunny           76    28        96
```

[28]:
```
df.iloc[lambda df: list(df.Humidity > 50), :]
```

```
[28]:     Weather  Temperature  Wind  Humidity
     Tue    Sunny           76    28        96
     Fri   Shower           70    26        79
     Sat   Shower           71    27        62
```

1.6 6. `loc` and `iloc` are interchangeable when labels are 0-based integers

```
[29]: data = [
          ['Mon','Sunny',78,13,30],
          ['Tue','Sunny',76,28,96],
          ['Wed','Sunny',78,16,20],
          ['Thu','Cloudy',68,11,22],
          ['Fri','Shower',70,26,79],
          ['Sat','Shower',71,27,62],
          ['Sun','Sunny',82,20,10]]
      df = pd.DataFrame(data)
      df
```

```
[29]:      0        1   2   3   4
      0  Mon    Sunny  78  13  30
      1  Tue    Sunny  76  28  96
      2  Wed    Sunny  78  16  20
      3  Thu   Cloudy  68  11  22
      4  Fri   Shower  70  26  79
      5  Sat   Shower  71  27  62
      6  Sun    Sunny  82  20  10
```

Now, `loc`, a label-based data selector, can accept a single integer and a list of integer values.

```
[30]: df.loc[1, 2]
```

```
[30]: 76
```

```
[31]: df.loc[1, [1, 2]]
```

```
[31]: 1    Sunny
      2       76
      Name: 1, dtype: object
```

`loc` and `iloc` are interchangeable when selecting via a single value or a list of values.

```
[32]: df.loc[1, 2] == df.iloc[1, 2]
```

```
[32]: True
```

```
[33]: df.loc[1, [1, 2]] == df.iloc[1, [1, 2]]
```

```
[33]: 1    True
      2    True
      Name: 1, dtype: bool
```

```
[ ]:
```