DataFrame.loc[ ]

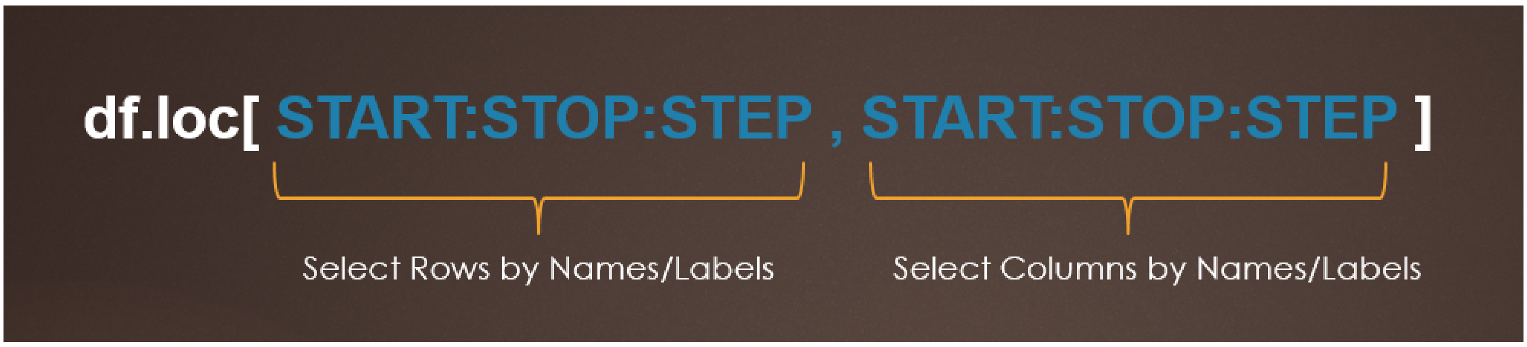
- In Pandas, the Dataframe provides a property loc[ ].
- It is used to select the subset of Dataframe.
- To access a group of rows and columns by label(s) or a boolean array.

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'.

Syntax:

> **Dataframe.loc[row_segment , column_segment]**


df.loc[START:STOP:STEP , START:STOP:STEP]
Select Rows by Names/Labels
Select Columns by Names/Labels

START is the name of the row/column label

STOP is the name of the last row/column label to take, and

STEP as the number of indices to advance after each extraction

Key points

By not providing a start row/column, loc[] selects from the beginning. By not providing stop, loc[] selects all rows/columns from the start label. Providing both start and stop, selects all rows/columns in between

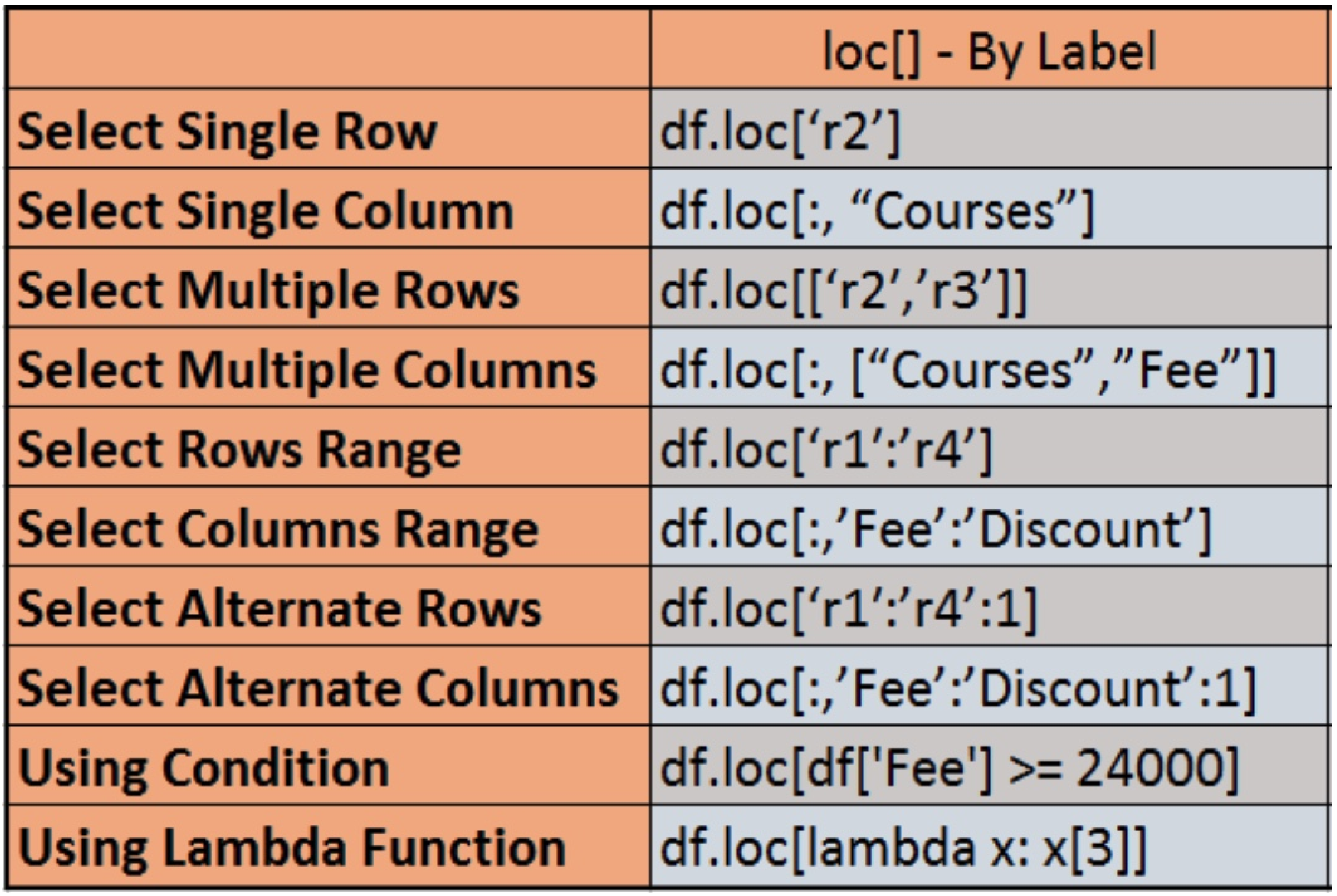
Note:

The column_segment argument is optional. Therefore, if column_segment is not provided, loc [ ] will select the subset of Dataframe based on row_segment argument only.

Let’s create a DataFrame and explore how to use pandas loc[].

```
import pandas as pd
technologies = {
    'Courses':["Spark","PySpark","Hadoop","Python","pandas"],
    'Fee' :[20000,25000,26000,22000,24000],
    'Duration':['30day','40days','35days','40days','60days'],
    'Discount':[1000,2300,1200,2500,2000]
              }
index_labels=['r1','r2','r3','r4','r5']
df = pd.DataFrame(technologies,index=index_labels)
print(df)
```

```
        Courses    Fee Duration  Discount
    r1    Spark  20000    30day      1000
    r2  PySpark  25000   40days      2300
    r3   Hadoop  26000   35days      1200
    r4   Python  22000   40days      2500
    r5   pandas  24000   60days      2000
```



	loc[] - By Label
Select Single Row	df.loc[‘r2’]
Select Single Column	df.loc[:, “Courses”]
Select Multiple Rows	df.loc[[‘r2’,’r3’]]
Select Multiple Columns	df.loc[:, [“Courses”,”Fee”]]
Select Rows Range	df.loc[‘r1’:’r4’]
Select Columns Range	df.loc[:,’Fee’:’Discount’]
Select Alternate Rows	df.loc[‘r1’:’r4’:1]
Select Alternate Columns	df.loc[:,’Fee’:’Discount’:1]
Using Condition	df.loc[df['Fee'] >= 24000]
Using Lambda Function	df.loc[lambda x: x[3]]

Select Single Row & Column By Label using loc[]

```
# Select Single Column by label
print(df.loc[:, "Courses"])
```

```
    r1      Spark
    r2    PySpark
    r3     Hadoop
    r4     Python
    r5     pandas
    Name: Courses, dtype: object
```

Select Multiple Rows & Columns

```
# Select Multiple Rows by Label
print(df.loc[['r2','r3']])
```

```
         Courses    Fee Duration  Discount
     r2  PySpark  25000   40days      2300
     r3   Hadoop  26000   35days      1200
```

Select multiple columns from pandas DataFrame.

```
# Select Multiple Columns by labels
print(df.loc[:, ["Courses","Fee","Discount"]])
```

```
         Courses    Fee  Discount
     r1    Spark  20000      1000
     r2  PySpark  25000      2300
     r3   Hadoop  26000      1200
     r4   Python  22000      2500
     r5   pandas  24000      2000
```

Select Between Two Rows or Columns

```
# Select Rows Between two Index Labels
# Includes both r1 and r4 rows
print(df.loc['r1':'r4'])
```

```
         Courses    Fee Duration  Discount
     r1    Spark  20000    30day      1000
     r2  PySpark  25000   40days      2300
     r3   Hadoop  26000   35days      1200
     r4   Python  22000   40days      2500
```

selects all columns between Fee and Discount column labels.

```
# Select Columns between two Labels
# Includes both 'Fee' and 'Discount' columns
print(df.loc[:,'Fee':'Discount'])
```

```
           Fee Duration  Discount
     r1  20000    30day      1000
     r2  25000   40days      2300
     r3  26000   35days      1200
     r4  22000   40days      2500
     r5  24000   60days      2000
```

Select Alternate Rows

```
# Select Alternate rows By indeces
print(df.loc['r1':'r4':2])
```

```
        Courses    Fee Duration  Discount
```

```
     r1   Spark  20000    30day       1000
     r3  Hadoop  26000   35days       1200
```

Select alternate columns use

```
# Select Alternate Columns between two Labels
print(df.loc[:,'Fee':'Discount':2])
```

```
            Fee  Discount
     r1  20000      1000
     r2  25000      2300
     r3  26000      1200
     r4  22000      2500
     r5  24000      2000
```

Using Conditions with pandas loc

```
# Using Conditions
print(df.loc[df['Fee'] >= 24000])
```

```
         Courses    Fee Duration  Discount
     r2  PySpark  25000   40days      2300
     r3   Hadoop  26000   35days      1200
     r5   pandas  24000   60days      2000
```

Complete Examples of pandas DataFrame loc

```
import pandas as pd
technologies = {
    'Courses':["Spark","PySpark","Hadoop","Python","pandas"],
    'Fee' :[20000,25000,26000,22000,24000],
    'Duration':['30day','40days','35days','40days','60days'],
    'Discount':[1000,2300,1200,2500,2000]
              }
index_labels=['r1','r2','r3','r4','r5']
df = pd.DataFrame(technologies,index=index_labels)
print(df)

# Select single Row
print(df.loc['r2'])

# Select Single Column by label
print(df.loc[:, "Courses"])

# Select Multiple Rows by Label
print(df.loc[['r2','r3']])

# Select Multiple Columns by labels
print(df.loc[:, ["Courses","Fee","Discount"]])

# Select Rows Between two Index Labels
# Includes both r1 and r4 rows
print(df.loc['r1':'r4'])
```

```
# Select Columns between two Labels
# Includes both 'Fee' and 'Discount' columns
print(df.loc[:,'Fee':'Discount'])

# Select Alternate rows By indeces
print(df.loc['r1':'r4':2])

# Select Alternate Columns between two Labels
print(df.loc[:,'Fee':'Discount':2])

# Using Conditions
print(df.loc[df['Fee'] >= 24000])
```

```
        Courses    Fee Duration  Discount
    r1    Spark  20000    30day      1000
    r2  PySpark  25000   40days      2300
    r3   Hadoop  26000   35days      1200
    r4   Python  22000   40days      2500
    r5   pandas  24000   60days      2000
    Courses     PySpark
    Fee           25000
    Duration     40days
    Discount       2300
    Name: r2, dtype: object
    r1      Spark
    r2    PySpark
    r3     Hadoop
    r4     Python
    r5     pandas
    Name: Courses, dtype: object
        Courses    Fee Duration  Discount
    r2  PySpark  25000   40days      2300
    r3   Hadoop  26000   35days      1200
        Courses    Fee  Discount
    r1    Spark  20000      1000
    r2  PySpark  25000      2300
    r3   Hadoop  26000      1200
    r4   Python  22000      2500
    r5   pandas  24000      2000
        Courses    Fee Duration  Discount
    r1    Spark  20000    30day      1000
    r2  PySpark  25000   40days      2300
    r3   Hadoop  26000   35days      1200
    r4   Python  22000   40days      2500
          Fee Duration  Discount
    r1  20000    30day      1000
    r2  25000   40days      2300
    r3  26000   35days      1200
    r4  22000   40days      2500
    r5  24000   60days      2000
       Courses    Fee Duration  Discount
    r1   Spark  20000    30day      1000
    r3  Hadoop  26000   35days      1200
          Fee  Discount
    r1  20000      1000
    r2  25000      2300
    r3  26000      1200
    r4  22000      2500
    r5  24000      2000
        Courses    Fee Duration  Discount
    r2  PySpark  25000   40days      2300
```

```
r3   Hadoop  26000   35days      1200
r5   pandas  24000   60days      2000
```

Let’s create another DataFrame and explore how to use pandas loc[].

```
import pandas as pd
# List of Tuples
students = [('jack',  34, 'Sydeny',    'Australia'),
            ('Riti',  30, 'Delhi',     'India'),
            ('Vikas', 31, 'Mumbai',    'India'),
            ('Neelu', 32, 'Bangalore', 'India'),
            ('John',  16, 'New York',   'US'),
            ('Mike',  17, 'las vegas',  'US')]

# Create a DataFrame from list of tuples
df = pd.DataFrame( students,
                   columns=['Name', 'Age', 'City', 'Country'],
                   index=['a', 'b', 'c', 'd', 'e', 'f'])
print(df)
```

```
    Name  Age       City    Country
a   jack   34     Sydeny  Australia
b   Riti   30      Delhi      India
c  Vikas   31     Mumbai      India
d  Neelu   32  Bangalore      India
e   John   16   New York         US
f   Mike   17  las vegas         US
```

Let's learn to apply loc[ ]

```
# Select row at with label name 'c'
a = df.loc['c']

print(a)
```

```
Name        Vikas
Age            31
City       Mumbai
Country     India
Name: c, dtype: object
```

```
# Select multiple rows from Dataframe by label names
subsetDf = df.loc[ ['c', 'f', 'a'] ]

print(subsetDf)
```

```
    Name  Age       City    Country
c  Vikas   31     Mumbai      India
f   Mike   17  las vegas         US
a   jack   34     Sydeny  Australia
```

```
# Select rows of Dataframe based on row label range
subsetDf = df.loc[ 'b' : 'f' ]

print(subsetDf)
```

```
    Name  Age       City Country
b   Riti   30      Delhi   India
c  Vikas   31     Mumbai   India
d  Neelu   32  Bangalore   India
e   John   16   New York      US
f   Mike   17  las vegas      US
```

```
# Select rows of Dataframe based on bool array ON ROWs
subsetDf = df.loc[ [True, False, True, False, True, False] ]

print(subsetDf)
```

```
    Name  Age      City    Country
a   jack   34    Sydeny  Australia
c  Vikas   31    Mumbai      India
e   John   16  New York         US
```

Select a few Columns from Dataframe (Slicing)

Here we will provide the (:) in the row segment argument of the Dataframe.loc[ ] . Therefore it will select all rows, but only a few columns based on the names provided in column_segement.

```
# Select single column from Dataframe by column name
column = df.loc[:, 'Age']
print(column)
```

```
a    34
b    30
c    31
d    32
e    16
f    17
Name: Age, dtype: int64
```

```
# Select multiple columns from Dataframe based on list of names
subsetDf = df.loc[:, ['Age', 'City', 'Name']]
print(subsetDf)
```

```
   Age       City   Name
a   34     Sydeny   jack
b   30      Delhi   Riti
c   31     Mumbai  Vikas
d   32  Bangalore  Neelu
e   16   New York   John
f   17  las vegas   Mike
```

```
# Select multiple columns from Dataframe by name range
subsetDf = df.loc[:, 'Name' : 'City']
print(subsetDf)
```

```
    Name  Age       City
a   jack   34     Sydeny
b   Riti   30      Delhi
c  Vikas   31     Mumbai
d  Neelu   32  Bangalore
e   John   16   New York
f   Mike   17  las vegas
```

```
# Select columns of Dataframe based on bool array
subsetDf = df.iloc[:, [True, True, False, False]]
print(subsetDf)
```

```
    Name  Age
a   jack   34
b   Riti   30
c  Vikas   31
d  Neelu   32
e   John   16
f   Mike   17
```

```
# Select a Cell value from Dataframe by row and column name
cellValue = df.loc['c','Name']
print(cellValue)
```

```
Vikas
```

```
# Select sub set of Dataframe based on row/column indices in list
subsetDf = df.loc[['b', 'd', 'f'],['Name', 'City']]
print(subsetDf)
```

```
    Name       City
b   Riti      Delhi
d  Neelu  Bangalore
f   Mike  las vegas
```

```
# Select subset of Dataframe based on row and column label name range.
subsetDf = df.loc['b':'e', 'Name':'City']
print(subsetDf)
```

```
    Name  Age       City
b   Riti   30      Delhi
c  Vikas   31     Mumbai
d  Neelu   32  Bangalore
e   John   16   New York
```

```
# Change the contents of row 'C' to 0
df.loc['c'] = 0
print(df)
```

```
    Name  Age       City    Country
a   jack   34     Sydeny  Australia
b   Riti   30      Delhi      India
c      0    0          0          0
d  Neelu   32  Bangalore      India
e   John   16   New York         US
f   Mike   17  las vegas         US
```

Additional Points:

- **row_segement:**
  - It contains information about the rows to be selected. Its value can be,
    - A single label like 'A' or 7 etc.
      - In this case, it selects the single row with given label name.
      - For example, if 'B' only is given, then only the row with label 'B' is selected from Dataframe.
    - A list/array of label names like, ['B', 'E', 'H']
      - In this case, multiple rows will be selected based on row labels given in the list.
      - For example, if ['B', 'E', 'H'] is given as argument in row segment, then the rows with label name 'B', 'E' and 'H' will be selected.
    - A slice object with ints like -> a:e .
      - This case will select multiple rows i.e. from row with label a to one before the row with label e.
      - For example, if 'B':'E' is provided in the row segment of loc[], it will select a range of rows from label 'B' to one before label 'E'
      - For selecting all rows, provide the value (:)
    - A boolean sequence of same size as number of rows.
      - In this case, it will select only those rows for which the corresponding value in boolean array/list is True.
    - A callable function :
      - It can be a lambda function or general function, which accepts the calling dataframe as an argument and returns valid label names in any one of the formats mentioned above.

- **column_segement:**
  - It is optional.
  - It contains information about the columns to be selected. Its value can be,
    - A single label like 'A' or 7 etc.
      - In this case, it selects the single column with given label name.
      - For example, if 'Age' only is given, then only the column with label 'Age' is selected from Dataframe.
    - A list/array of label names like, ['Name', 'Age', 'City']
      - In this case, multiple columns will be selected based on column labels given in the list.
      - For example, if ['Name', 'Age', 'City'] is given as argument in column segment, then the columns with label names 'Name', 'Age', and 'City' will be selected.
    - A slice object with ints like -> a:e .
      - This case will select multiple columns i.e. from column with label a to one before the column with label e.
      - For example, if 'Name':'City' is provided in the column segment of loc[], it will select a range of columns from label 'Name' to one before label 'City'
      - For selecting all columns, provide the value (:)
    - A boolean sequence of same size as number of columns.
      - In this case, it will select only those columns for which the corresponding value in boolean array/list is True.
    - A callable function :
      - It can be a lambda function or general function that accepts the calling dataframe as an argument and returns valid label names in any one of the formats mentioned above.

Questions related to Dataframe.loc[ ] can be:

These are categorized into three parts i.e.

Select a few rows from Dataframe, but include all column values

- Select a single row of Dataframe
- Select rows of Dataframe based on row label names in list
- Select rows of Dataframe based on row label name range
- Select rows of Dataframe based on bool array
- Select rows of Dataframe based on callable function

Select a few columns from Dataframe, but include all row values for those columns.

- Select a single column of Dataframe
- Select columns of Dataframe based on column names in list
- Select columns of Dataframe based on column name range
- Select columns of Dataframe based on bool array
- Select a subset of Dataframe with few rows and columns
- Select a Cell value from Dataframe
- Select subset of Dataframe based on row/column names in list
- Select subset of Dataframe based on row and column name range.

Change values of Dataframe by loc[]